Università degli Studi di Cagliari

Department of Mathematics and Computer Science
Ph.D. Course in Mathematics and Computer Science
Cycle XXXII

Ph.D. Thesis

Knowledge Extraction from Textual Resources through Semantic Web Tools and Advanced Machine Learning Algorithms for Applications in Various Domains

S.S.D. INF/01

Candidate
Danilo Dessí

Supervisor
Prof. Diego Reforgiato Recupero

PhD Coordinator
Prof. Michele Marchesi

Final examination academic year 2018/2019
February, 2020

Abstract

Nowadays there is a tremendous amount of unstructured data, often represented by texts, which is created and stored in variety of forms in many domains such as patients' health records, social networks comments, scientific publications, and so on. This volume of data represents an invaluable source of knowledge, but unfortunately it is challenging its mining for machines. At the same time, novel tools as well as advanced methodologies have been introduced in several domains, improving the efficacy and the efficiency of data-based services.

Following this trend, this thesis shows how to parse data from text with Semantic Web based tools, feed data into Machine Learning methodologies, and produce services or resources to facilitate the execution of some tasks.

More precisely, the use of Semantic Web technologies powered by Machine Learning algorithms has been investigated in the Healthcare and E-Learning domains through not yet experimented methodologies. Furthermore, this thesis investigates the use of some state-of-the-art tools to move data from texts to graphs for representing the knowledge contained in scientific literature. Finally, the use of a Semantic Web ontology and novel heuristics to detect insights from biological data in form of graph are presented. The thesis contributes to the scientific literature in terms of results and resources. Most of the material presented in this thesis derives from research papers published in international journals or conference proceedings.

Acknowledgements

This thesis ends a long period of my life. It started when I went to the high school to study computer science and has continued so far. I tried to understand all aspects of this field, but I continue to find new things that make me ever more interested to learn and explore. During years, I have first decided to go to the university, to take a M.Sc. degree, and finally to follow a Ph.D course. I have had the opportunity to meet a lot of people from whom I learnt more than from books. They have transferred to me their passion for their work and have been (and will be) source of inspiration. Today, I feel I still have a lot to learn and I am looking forward to next steps.

Focusing to this moment, throughout the work presented in this dissertation I have received a great deal of support and assistance. I would first like to thank my supervisor, Prof. Diego Reforgiato Recupero. I would like to express my most sincere appreciation to him for his continuous support day by day during the entire experience. I am so grateful to have a supervisor who cared so much about me, my work, and who responded to my questions and troubles so promptly. I learnt a lot from you!

I would also like to express my most sincere gratitude for all supervisors I had in my abroad experiences, chronologically, Dr. Sergio Consoli, Prof. Dennis Shasha, Dr. Francesco Osborne, Prof. Enrico Motta, and Prof. Davide Buscaldi. I would also like to thank all research groups that hosted me, and all people I have met during these adventures. All you have made my abroad experiences unforgettable.

A special thanks to Mirko Marras, with whom I shared most of my days during M.Sc. and Ph.D. studies. I believe we have done great so far and we will continue to do so.

Also outside from the work, I have received important support. First of all, from my parents Carla and Gino, and my sister Sara, who I would like to thank with love and gratitude for all the support. This work is largely yours.

Finally, I would like to thank my friends who supported what I have been doing, providing happy distraction and friendships.

Funding

Danilo Dessì acknowledges Sardinia Regional Government for the financial support of his Ph.D. scholarship (P.O.R. Sardegna F.S.E. Operational Programme of the Autonomous Region of Sardinia, European Social Fund 2014-2020, Axis III "Education and Training", Specific Goal 10.5).

Publications

The research reported in this thesis has contribuited to the following publications:

- [1] Dessí, D., Dragoni, M., Fenu, G., Marras, M., & Reforgiato Recupero, D. (2019). Evaluating Neural Word Embeddings Created from Online Course Reviews for Sentiment Analysis. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (pp. 2124-2127). ACM.

- [2] Dessí, D., Fenu, G., Marras, M., & Reforgiato Recupero, D. (2019). Bridging Learning Analytics and Cognitive Computing for Big Data classification in Micro-learning Video Collections. Computers in Human Behavior, 92, (pp. 468-477).

- [3] Buscaldi, D., Dessí, D., Motta, E., Osborne, F., & Reforgiato Recupero, D. (2019). Mining Scholarly Data for Fine-Grained Knowledge Graph Construction. In CEUR Workshop Proceedings (Vol. 2377, pp. 21-30).

- [4] Dessí, D., Reforgiato Recupero, D., Fenu, G., & Consoli, S. (2019). A Recommender System of Medical Reports Leveraging Cognitive Computing and Frame Semantics. In Machine Learning Paradigms (pp. 7-30). Springer, Cham.

- [5] Bardaro G., Dessí, D., Motta, E., Osborne, F., & Reforgiato Recupero, D. (2019). Parsing Natural Language Sentences into Robot Actions. In CEUR Workshop Proceedings (Vol. 2456, pp. 93-96).

- [6] Buscaldi, D., Dessí, D., Motta, E., Osborne, F., & Reforgiato Recupero, D. (2019). Mining Scholarly Publications for Scientific Knowledge Graph Construction. ESWC 2019. Lecture Notes in Computer Science, vol 11762. Springer, Cham.

- [7] Dessí, D., Dragoni, M., Fenu, G., Marras, M., & Reforgiato Recupero, D. (2020). Deep Learning Adaptation with Word Embeddings for Sentiment Analysis on Online Course Reviews. In Deep Learning-Based Approaches for Sentiment Analysis (pp. 57-83).

- [8] Reforgiato Recupero, D., Dessí, D., & Concas, E. (2019). A Flexible and Scalable Architecture for Human-Robot Interaction. AmI 2019. Lecture Notes in Computer Science, vol 11912. Springer, Cham.
- [9] Dessí, D., Cirrone, J., Reforgiato Recupero, D., & Shasha, D. (2018). SuperNoder: a Tool to Discover Over-represented Modular Structures in Networks. BMC bioinformatics, 19(1), 318.
- [10] Dessí, D., Fenu, G., Marras, M., & Reforgiato Recupero, D. (2018). COCO: Semantic-Enriched Collection of Online Courses at Scale with Experimental Use Cases. In World Conference on Information Systems and Technologies (pp. 1386-1396). Springer, Cham.
- [11] Dessí, D., Fenu, G., Marras, M., & Reforgiato Recupero, D. (2017). Leveraging Cognitive Computing for Multi-class Classification of E-Learning videos. In European Semantic Web Conference (pp. 21-25). Springer, Cham.
- [12] Dessí, D., Reforgiato Recupero, D., Fenu, G., & Consoli, S. (2017). Exploiting Cognitive Computing and Frame Semantic Features for Biomedical Document Clustering. In SeWeBMeDA@ ESWC (pp. 20-34).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Research Contributions

Nowadays there is a tremendous amount of unstructured data, often represented by texts, which is created and stored in variety of forms in many domains such as patients' health records, social networks comments, scientific publications, and so on. Textual data is one of the simplest forms of data representation that is created by humans because it results easy to understand. Moreover, it is worth to note that this volume of data represents an invaluable source of knowledge that cannot be ignored. However, textual data is undoubtedly challenging to be understood and mined by machines, representing an actual unsolved research problem. Thus, an ever-increasing interest in methods and technologies aimed to process unstructured and textual data has gained the attention of the scientific community.

Dealing with such textual data involves techniques which are aimed to detect no trivial features, represent data in machine-readable formats, and extract the underlying knowledge. To accomplish these tasks, the adoption of Semantic Web technologies powered by Machine Learning methodologies is showing a great potential and relevant advancements, positively impacting results of the use of machines to get insights out from data.

There are two main aspects that need to be considered in the process of getting knowledge out from unstructured data: (i) data needs to be modeled by means of representations that hold data peculiarities and allow an efficient execution of algorithms, and (ii) methodologies and algorithms need to be developed according to the expected knowledge. Both aspects are relevant in the task. Erroneous data representation can lead to no results of the adopted approach, or worse to wrong results and conclusions. At the same time, methodologies to detect the data patterns and insights have the role to perform those steps that transform the structured data into useful knowledge.

Broadly, depending on the kind of input data, different approaches to model data and perform algorithms can be adopted. Two of the most exploited data rep-

resentations are the vector-based representation, where a numerical vector is usually assigned to each data item and the knowledge is embedded in its values, or graphs, where data items are represented by nodes and the knowledge is represented by patterns that can be detected by studying the connections between them. According to the type of representation, novel methods can be tested to get insights out from data.

In this thesis, the use of Semantic Web technologies powered by Machine Learning algorithms has been investigated in the Healthcare and E-Learning domains through not yet experimented methodologies. Moreover, it investigates the use of some state-of-the-art tools to move data from texts to graphs for representing the knowledge of a domain. Finally, the use of a Semantic Web ontology and novel heuristics to detect insights from biological data in form of graph are presented. The thesis contributes to the scientific literature in terms of results and resources.

The research program addressed during the Ph.D. course was dedicated to the study and development of methodologies to extract, represent, and use of knowledge from textual resources.

The main research questions addressed in the target domains are:

Q1. How to use existing Semantic Web technologies to retrieve useful information in order to model contents of texts in machine readable formats?

Q2. What are the Machine Learning algorithms more suitable to infer knowledge from the modelled information?

Q3. In literature many general purpose methods can be found to address text-based applications. May these methods be used for specific domain applications?

Q4. How different techniques and their combinations will impact the performances of specific applications for a target domain?

By answering to these research questions, the contributions provided in this research work are:

- A study for the Healthcare domain that shows how textual patients records can be exploited to recognize similar health states of unseen patients that are similar to those already known.

- A novel framework for the E-Learning domain for correctly classifying online videos of courses in pre-defined categories.

- An E-Learning dataset crawled from the web which contains data about learners, instructors, and resources.

- A Deep Learning study for predicting a sentiment score for reviews left by learners within E-Learning platforms.

- A methodology to build knowledge graphs for the Scholarly Domain to support the modelling and forecasting of ideas and technologies across research communities.

- A tool called *Supernoder* which provides heuristics aimed to discover patterns on a graph and applications on biological domain.

1.2 Dissertation Structure

The thesis is organized as follows:

- Chapter 2 introduces basic concepts that have been adopted across the various addressed research problems. Moreover, it describes which software tools have been adopted.

- Chapter 3 analyzes the background of current research on textual data for the Healthcare domain and presents a novel methodology for mining patients records. This work has been done in collaborations with the Professor Diego Reforgiato (*University of Cagliari*), Professor Gianni Fenu (*University of Cagliari*) and Dr. Sergio Consoli (who was Senior Data Scientist at the *Data Science Department* of the *Philips Research*. He is currently affiliated to the Joint Research Centre of the European Commission). The work has been published in [12, 4].

- Chapter 4 presents the supervised methodologies adopted on the E-Learning domain for resources categorization and sentiment prediction tasks. In addition, it describes a new dataset collected during the research work. These works have been done in collaborations with the Professor Diego Reforgiato (*University of Cagliari*), Professor Gianni Fenu (*University of Cagliari*), and Dr. Mirko Marras (*University of Cagliari).* The Sentiment Analysis study has been also supervised by Dr. Mauro Dragoni who is researcher at *Fondazione Bruno Kessler*. These works have been published in [11, 2, 10, 1]

- Chapter 5 presents a methodology for building a knowledge graph which represents knowledge about the Semantic Web domain. Methods to solve open issues adopting supervised and unsupervised approaches are discussed. This work has been done within the collaboration with the Professor Diego Reforgiato (*University of Cagliari*), Professor Enrico Motta, (*The Open University*), Dr. Francesco Osborne (*The Open University*), and Professor Davide Buscaldi (*University Paris 13*) during my visiting in all their institutions. Preliminary results about this work have been published in [3, 6].

- Chapter 6 describes 5 heuristics to discover patterns that appear disjointedly on a graph and shows the benefits of abstracting graphs by means of new nodes that we call *supernodes*. This work has been done within the collaboration with Professor Diego Reforgiato (*University of Cagliari*), Professor Dennis Shasha (*New York University*), and Dr. Jacopo Cirrone (*New York University*) during

my visiting at the *New York University* in fall 2017. This work has been published in [9].

- Conclusions about the use of Machine Learning algorithms and Semantic Web technologies on the studied domains have been reported in Chapter 7.

Chapter 2

Machine Learning Basics

2.1 Supervised approaches

A supervised approach is a learning method pertaining to infer a function or a classifier from a set of labeled training data in order to perform predictions on unseen data. More formally, given a set of records $R = \{r_1, ..., r_n\}$ which models our data and such that each one is labeled with one ore more labels c_i of a set $C = \{c_1, ..., c_m\}$, a supervised based approach is aimed to infer a function $\gamma : R \longrightarrow C$ that relates each record in R to one or more classes in C. The supervised approaches adopted during the research work presented in this thesis are:

- **Support Vector Machine.** Support Vector Machine (SVM) algorithm works by defining boundaries through hyperplanes in order to separate a class from the others. The aim of this algorithm is to build hyperplanes among data samples in such a way that the separation between classes is as large as possible. The algorithm takes labeled pairs (x_i, y_i) where x_i is a vector representation of input data, and y_i is a numerical label. The algorithm then applies an optimization function in order to separate classes. The regression variant of SVM, generally called SVR (Support Vector Regressor), tries to find hyperplanes that can predict the distribution of information.

- **Support Vector Machine + Stochastic Gradient Descent (SVM+SGD).** This method extends the standard SVM implementation including the SGD algorithm during training. SGD finds the best coefficients describing the decision boundaries through a classification function which minimizes a hinge loss function and allows performing training over large data while reducing the computation time.

- **Decision Trees.** Decision Trees (DT) recursively split the training data in smaller subsets trying to maximize a gain function until no more gain can be obtained with subsequent splitting. First, the root node is split into several

children according to the given criteria. Then, the process recursively continues on children until there are not other nodes to be split.

- **Random Forests.** Random Forests (RF) are based on an ensemble of decision trees, where each tree is independently trained and votes for a class for the data presented as an input [13]. Essentially, each decision tree splits data into smaller data groups based on the features of the data until there are small enough sets of data that only have data points with the same label. These splits are chosen according to a purity measure and, for each node, the algorithm tries to maximize the gain computed on it.

- **Neural Networks.** This method involves one or more levels of nodes, which are randomly joined by weighted connections in a many-to-many fashion. Neural Networks usually learn by activating nodes of a certain level and propagating the signal to other nodes. The activation of a node depends on both the input of the node and the function that is assigned to that node. More details about modern Neural Networks methods have been deeply described in Section 2.3.

2.2 Unsupervised approaches

An unsupervised approach is a learning method which tries to find hidden structures of patterns out from unlabeled data. It can be applied on any kind of data because it does not need of a training stage. One of the most common unsupervised approach is named *clustering*, which is aimed to segment a collection of records $R = \{r_1, ..., r_n\}$ into partitions $C = \{c_1, ..., c_m\}$ called *clusters* where records in the same cluster are more similar in somehow to each other than those in other clusters. Two of the most used clustering algorithms, named *Hierarchical clustering* and *K-means clustering* are described:

- **Hierarchical Clustering.** Hierarchical Clustering builds a clusters hierarchy, or in other words, a tree of clusters which is usually called *dendrogram*. Each cluster contains children that are clusters as well, unless for the leafs of the tree. Sibling clusters split documents that are contained in the common parent cluster. A hierarchical clustering algorithm can be either *agglomerative* or *divisive*. In its agglomerative version, the algorithm starts with single elements of the collection, then it merges elements together based on a chosen measure (e.g., *Euclidean distance*). The agglomerative process is iterated as long as a unique cluster that covers all documents collection is obtained. The divisive variant of the algorithm starts with one cluster that contains all elements of the input collection, and recursively splits the most appropriate clusters according to a given criteria (e.g., splitting the largest cluster in each iteration.). The method continues its execution until a stop criterion is

achieved (e.g., the process stops when a given number of clusters has been obtained). Hierarchical clustering algorithms are easily applicable on each kind of data, enable a manageable granularity of clusters, and can be applied with any type of similarity measures.

- **K-means Clustering.** The K-means Clustering is a partition method. It builds a set of clusters minimizing the sum of squared distance between elements of a cluster and its center. The result is a single partition of data without any structure and, hence, can have advantages on applications which involve large sets of data for which the construction of a hierarchical structure can be onerous. The algorithm requires the number of clusters k as an input. This number is used to allocate k random centers which will be employed to build clusters. At beginning, it assigns each element to the cluster with the nearest center. Iteratively, centers are updated based on the built clusters and elements are moved into the cluster with their nearest center.

2.3 Deep Learning

Deep Learning has emerged as a subclass of the Machine Learning area, where various neural network approaches are combined together for pattern classification and regression tasks. It usually employs multiple layers able to learn complex data representation and increasingly higher level features, and to correctly classify or measure properties held by data. The success of Deep Learning is due to the new advancements in the Machine Learning field as well as to the ever more increasing computational abilities of computers through the use of Graphical Processing Units (GPUs) [14].

Broadly, a Deep Learning model embraces information processing methods consisting of a sequence of complex non-linear models. Each model forms a layer that independently processes data. The output of a layer is fed as an input to the subsequent layer in the sequence until the final output is obtained.

Feed-forward Neural Network (FNN)

Feed-forward Neural Networks (FNN) were one of the first and simplest components applied to learn from data using the Deep Learning paradigm. One or more levels of nodes, often called perceptrons, are randomly joined by weighted connections in a many-to-many fashion. These networks were historically thought in order to simulate a brain biological model where nodes are neurons and links between them represent synapses. For this reason, they are also called Multi-Layer Perceptron (MLP) networks. On the basis of the input values fed into the network, nodes of a certain level can be activated and their signal is broadcasted to the subsequent level. In order to activate nodes of a subsequent level, the signal generated at a given level is weighted and must be greater than a given threshold. Weights are

generally initialized with random values and adjusted during training in order to minimize a predefined objective function. This family of networks has been proved to be useful for pattern classification, but less suitable for labelling sequences since it does not take into account the sequence of input data. A simple schema of a three-layer Feed-forward Neural Network model is shown in Figure 2.1.

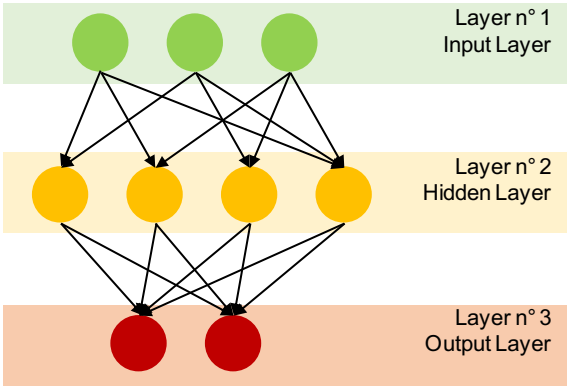


Figure 2.1: An example of a Feed-forward Network composed by three layers.

The sample FNN as it is accepts three-dimensional inputs (in green) and returns two-dimensional outputs (in red). Each node of a given layer is connected to nodes of the subsequent layer. The input data is fed into the network by means of Layer 1, which acts as Input Layer, and then sent to the first hidden layer, i.e., Layer 2. The output of this layer is finally propagated to Layer 3, which represents the Output Layer. The action to move data from a layer to another by activating or not the corresponding nodes is generally called *forward pass* of the network.

Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNN) are tailored for processing data as a sequence. In contrast to FNNs which commonly pass the input data directly from input to output nodes, RNNs have cyclic or recurrent connections among nodes of distinct levels. This makes possible to model the output of the network by taking into account the history of the received input data. Recurrent connections connect past data with the one that is currently being processed, simulating a status memory. The forward pass is similar to the one in FNNs, with the difference that the activation of a node depends on both the current input and the previous status of hidden layers.

It is worth to note that in a wide range of applications data can present patterns from the past to the future and vice versa. For instance, for classifying the sections of a given story, it could be useful to have access to both past and future sections. However, the future content of a text is ignored by common FNNs and RNNs, as they work sequentially. Bidirectional RNNs (BiRNNs) let the network, at a given

point in time, to take information from both earlier and later data in the sequence, going beyond the exposed limitation. The idea behind this kind of networks consists of presenting the training data forwards and backwards by two hidden RNNs which are then combined into a common output layer. This strategy makes possible to find patterns that can be learnt from both past and future history of data.

Long Short-Term Memory (LSTM) Network

Long Short-Term Memory (LSTM) networks are an RNN extension designed to work on sequential data and have achieved state-of-the-art results on challenging prediction tasks. LSTM networks employ recurrent connections and add memory blocks in their recurrent hidden layers. Memory blocks save the current temporal state during training and make possible to learn temporal observations hidden in the input data. The fact of using connections as a memory implies that the output of a LSTM network depends on the entire history of the training data, not only on the current input sample. Moreover, using memory blocks allows to relate the current data that is being processed with the data processed long before, solving the problem experienced by common RNNs. For this reason, LSTM networks have had a positive impact on sequence prediction tasks. As stated for RNN, a bidirectional layer using two hidden LSTMs can be leveraged to process data both forward and backward.

Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) typically perform filtering operations on the input nodes of a layer, abstracting and selecting only meaningful input features. When CNNs are trained, the weights of links between nodes acting as filters are defined. Such networks have been historically applied in the computer vision field, only recently they have been also applied on textual data.

Normalization Layer (NL)

During training, the output of a given layer is affected by parameters and processes used in previous layers. Small changes in the parameters set for a layer can hence be propagated as the network becomes deeper, resulting in large changes in the final output, i.e., the output of the last layer. Considering that the parameters are continuously changed to better fit the prediction task and that the data distribution changes across levels, the variations within the parameters can negatively influence the training, making it computationally expensive. To shape input data with a standard distribution and improve training performance, some Normalization Layers (NL) can be introduced. One of the most common normalization layers is represented by *Batch Normalization*. This layer makes possible to reduce the dependence of the optimization parameters from the input values, avoiding over-fitting and making the training process more stable.

Attention Layer (AL)

Attention Layers (ALs) are often adopted before the last fully-connected layers of a model. Attention mechanisms in neural networks serve to orient perception as well as memory access. Attention layers filter the perceptions that can be stored in memory, and filter them again on a second pass when they need to be retrieved from memory. Neural networks can allocate attention, and they can learn how to do so, by adjusting the weights they assign to various inputs. This makes possible to solve traditional limits in various Natural Language Processing (NLP) tasks. For instance, traditional word vectors presume that words meaning is relatively stable across sentences. However, there could be massive differences in meaning for a single word: e.g., lit (an adjective that describes something burning) and lit (an abbreviation for literature); or get (a verb for obtaining) and get (an animals offspring). Attention Layers can capture the shades of meaning for a given word that only emerge due to its situation in a passage and its inter-relations with other words. Moreover, Attention Layers learn how to relate an input sequence to the output of the model in order to pay selective attention on more relevant input features. For example, in order to reflect the relation between inputs and outputs, an Attention Layer may compute an arithmetic mean of results of various layers according to a certain relevance.

Other Layers

There are also other layers that can be leveraged in order to fine-tune the performance of a model. The most representative ones are described below.
Embedding Layer. An Embedding layer turns positive integers (indexes) into dense vectors of fixed size chosen from a pre-initialized matrix. For an integer-encoded text, the dense vector corresponding to those integers are selected.
Noise Layer. A Noise layer is usually employed to avoid model over-fitting. It consists in modifying a fraction of input of layers, adding and subtracting some values following a predefined distribution (e.g., Gaussian).
Dropout Layer. A Dropout layer may be seen as a particular type of noise layer. It assigns the value 0 to a randomly chosen fraction of its input data. The name *Dropout* comes from the action of dropping some units of the input.
Dense Layer. A Dense layer is a densely-connected layer that is used to map large unit inputs in a few unit results. For example, it may be used to define the number of classes that a model returns, mapping hundred and thousand nodes in a few number of classes.

2.4 Embeddings

Many Machine Learning solutions leverage word embeddings, i.e., distributed representations that model words properties in vectors of real numbers which capture

syntactic features and semantic word relationships. These resources have been useful in NLP tasks, like Part-Of-Speech (POS) tagging [15] and Word Analogy [16], and they have been also exploited for Sentiment Analysis [17, 18, 19, 20, 21]. The generation of word embeddings is based on distributional co-occurrences of adjacent words able to model words meanings that are not visible from their surface. This exploits the fact that words with a similar meaning tend to be connected by a given relation. For instance, the verbs *utilize* and *use*, which are synonyms although syntactically different, present similar sets of co-occurring words and can be considered similar, while a third verb, such as *run*, has different co-occurrences and should be considered different from both *utilize* and *use*. An overview of the most representative and recent word embedding generator algorithms is introduced in this section, highlighting their pros and cons.

Word2Vec

The *Word2Vec* word embedding generator [22] aims to detect the meaning and semantic relations between words by exploiting the co-occurrence of words in documents belonging to a given corpus. The core idea is to capture the context of words, using Machine Learning approaches such as Deep Neural Networks. In order to eliminate noise, *Word2Vec* operates on a corpus of sentences by constructing a vocabulary based on the words that appear in the corpus more often than a user-defined threshold. Then, it trains either the Continuous Bag-Of-Words (CBOW) or the Skip-gram algorithm on the input documents to learn the word vector representations.

GloVe

The *GloVe* [23] word embedding generator is an unsupervised learning algorithm developed by Stanford. It creates word embeddings by aggregating global word-word co-occurrence matrices from a corpus. The resulting embeddings show interesting linear substructures of the word in the vector space. More precisely, the algorithm consists of collecting word co-occurrence statistics in a form of word co-occurrence matrix. Each element of this matrix represents how often the word i appears in context of word j. The corpus is scanned in the following manner: for each term, it looks for context terms within some area defined by a *windowsize* before the term and a *windowsize* after the term, and assigns less weight for more distant words.

FastText

The *FastText* [24] word embedding generator is another algorithm for learning word representations. It differs form the previous ones in the sense that word vectors as the ones learned in *Word2Vec* treat every single word as the smallest unit whose vector representation is to be found, while *FastText* assumes a word to be formed by n-grams of character. This new representation of a word is helpful to find the vector

representation for rare words. Since rare words could still be broken into character n-grams, they could share these n-grams with the common words. This can help to manage vector representations for words not present in the dictionary since they can also be broken down into character n-grams. Character n-grams embeddings tend to perform superior to *Word2Vec* and *GloVe* on smaller datasets [25].

Intel

Intel proposes to improve the data structures in *Word2Vec* through the use of mini-batching and negative sample sharing, allowing to solve the neural word embedding generation problem using matrix multiply operations [26]. They explored different techniques to distribute *Word2Vec* computation across nodes in a cluster, and demonstrate strong scalability. Their algorithm is suitable for modern multi-core/many-core architectures and allows scaling up the computation near linearly across cores and nodes, and processing millions of words per second. *Intel* embeddings generally differ from *Word2Vec* embeddings since the number of updates of the model is different across these two implementations, and the convergence is not equal for the same number of epochs.

BERT

The Bidirectional Encoder Representations from Transformers (BERT) were introduced in late 2018. It is a novel method of pre-trained language representations that allows to tune the vector representation of a word to the real meaning that it has in a context, overcoming ambiguity issues of words. One of the famous examples is usually reported with the word *bank*. Consider the two sentences "*The man was accused of robbing a bank*" and "*The man went fishing by the bank of the river*". Previous introduced word embedding models describe the word *bank* with the same word embedding, i.e., they associate the same vector to express the syntactic and semantic features of the word, while BERT produces two different word embeddings, coming up with more accurate representations for the two different meanings. For doing so, BERT computes context-tuned word embeddings resulting in a more accurate representations which can lead to better models performances.

2.5 Metrics

Term Frequency-Inverse Document Frequency

This is a *bag-of-word* model where attributes are words within a collection. For assigning a value to each word w, it is common using the Term Frequency - Inverse Document Frequency (TF-IDF) technique in which (i) uncommon words are not less relevant from frequent ones, (ii) a word that occurs many times in a document is not less relevant than a single one, and (iii) the length of documents does not play a

significant role for the comparison of documents. To put it more simply, words that frequently occur within a document, but rarely in the whole collection, have more probability to be relevant in the document. The TF-IDF formula is shown in (2.1) where w_{ki} is the number of occurrences of the word w_k in the document d_i, $|d_i|$ is the size of the document expressed as number of words, N is the number of documents in the collection, and n_k is the number of documents where the word w_k occurs at least once. TF-IDF values are usually normalized in the range [0,1].

$$TF - IDF(w_k, d_i) = \frac{w_{ki}}{|d_i|} \cdot log\frac{N}{n_k} \tag{2.1}$$

Cosine Similarity

The Cosine similarity quantifies the angle between two vectors. Its formula applied on two vectors v_p and v_q can be observed in (2.2).

$$CosS(v_p, v_q) = \frac{v_p\, v_q}{\parallel v_p \parallel \parallel v_q \parallel} \tag{2.2}$$

$CosS$ values 1 when v_p and v_q are completely similar, and 0 otherwise.

Euclidean distance

The Euclidean distance $EucD$ between two vectors v_p and v_q is defined as usual in (2.3).

$$EucD(v_p, v_q) = \sqrt{\sum_i (v_p(i) - v_q(i))^2} \tag{2.3}$$

Differently from the Cosine similarity, the Euclidean distance has not a limited range of values, therefore, it needs to be scaled before used for similarity evaluation. For such reason our modules adopt the formula (2.4) for scaling Euclidean-based values.

$$EucS(v_p, v_q) = \frac{1}{1 + EucD(v_p, v_q)} \tag{2.4}$$

Precision

The precision is a measure that is often used for evaluating supervised methods. It indicates how many items have been correctly classified for a given class considering all elements that have been classified with that class. It is expressed by the following equation:

$$P = \frac{TP}{TP + FP} \tag{2.5}$$

In equation (2.5) TP is the number of items correctly classified for the target class and FP is the number of elements that have been erroneously classified for that class.

Recall

The Recall is a metric that is often computed together with the Precision. It is defined by the equation (2.6), and measures how many items that belong to a given class have been properly classified for the class.

$$R = \frac{TP}{TP + FN} \tag{2.6}$$

As for the Precison, in equation (2.6) TP is the number of items correctly classified for the target class. FN is the number of elements that belong to the target class but have not been labeled for that class.

F-measure

A measure often used to combine Precision and Recall is the F-measure. It is computed as their harmonic mean as shown in equation (2.7).

$$F = 2 \cdot \frac{P \cdot R}{P + R} \tag{2.7}$$

Mean Absolute Error

The Mean Absolute Error (MAE) is a measure of difference between two continue variables expressed by:

$$MAE(y, \hat{y}) = \frac{1}{n} \cdot \sum_{i=0}^{n-1} |y_i - \hat{y}_i| \tag{2.8}$$

where y_i is a true target value, $\hat{y}_i$ is a predicted target value, and n is the number of samples.

Mean Squared Error

The Mean Squared Error (MSE) is a measure of difference between two continue variables that shows how close they are. It is expressed by:

$$MSE(y, \hat{y}) = \frac{1}{n} \cdot \sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2 \tag{2.9}$$

where y_i is a true target value, $\hat{y}_i$ is a predicted target value, and n is the number of samples. The MAE and MSE are often adopted for evaluating supervised regression approaches.

2.6 Software Tools and Technologies

Cognitive Computing and IBM Watson

With the terms Cognitive Computing systems we refer to those smart systems that learn at scale, can learn with purpose, and recently have modules for interacting directly with humans. They are being developed to reduce costs, increase efficiency, accelerate discovery, make essential connections in large amounts of data. With the rapid growth of the availability of massive amounts of data, Cognitive Computing systems represent a new appealing model to develop applications capable of working well where traditional methods fail because they are limited by a high level of uncertainty, noise and complexity in data processing. In general, its traditional applications are based on text mining techniques while there are other ones based on Data Mining, Machine Learning and Deep Learning [27]. Cognitive Computing systems can ingest data from external resources, add functions to identify patterns and relationships in large and unstructured data, and consequently interpret massive amount of varieties of them. In this way, they attempt to mimic human behaviors, adding the ability to manage huge amount of data [28]. Cognitive Computing services can reduce the gap between the interpretation and the summarization of information, learning from a large set of interrelated concepts, providing a key for their comprehension and generalization. Embedding Cognitive Computing services in novel systems results fundamental for dealing with previous unmanageable issues.

One of the most popular Cognitive Computing system is IBM Watson. On February 2011, it was introduced as a question-answering system based on advanced NLP, information retrieval, knowledge representation, and automated reasoning. It is currently composed by 14 tools which provide analytics and interaction services. In this set, the Natural Language Understanding analytics tool[1] which provides a cloud suite of services by means of the IBM Cloud[2] platform for dealing with huge amount of data, provides a collection of natural language APIs which enables developers to explore unstructured text, detecting and inferring high-level insights. This service applies multiple technologies to enable the comprehension of vast data resources, independently from their domain. It provides, among others, the following set of functions: entity extraction, sentiment analysis, keyword extraction, concept tagging, relation extraction, language detection, text extraction, micro-formats parsing, feed detection and linked data. Processing the semantic context provided by these functions helps to infer high-level features characterizing the topic of a text,

[1] `https://www.ibm.com/watson/services/natural-language-understanding/`
[2] `https://www.ibm.com/cloud/`

without suffering from noisy data.

Frame Semantics and Framester

Frame semantics is a linguistic theory that defines a meaning as a coherent structure of related concepts [29]. To relate various concepts, knowledge-based resources are usually employed as corner stones of semantic technological approaches. In order to embed frame semantics in our modules we exploited Framester, a novel data linked resource that works as a hub between linked open data systems as FrameNet, BabelNet and WordNet. It is a new frame-based ontological resource that leverages an inter-operable predicate space formalized according to frame semantics [30] and semiotics [31].

Big Data Technologies

The Big Data movement consists of increasingly powerful and relatively inexpensive computing platforms with fault-tolerant storage and processing carried out through thousands of processors. The current volume of data managed by existing systems has surpassed the processing capacity of the traditional ones and this applies to Data Mining as well [32]. The arising of new technologies and services (e.g., Cloud Computing and Grid Computing) and the reduction in hardware price have led to an ever-growing rate of information on the Internet. Consequently, Big Data applications in various domains, such as economics and finance [33], and computer networks [34], can be efficiently moved into clouds to analyze larger amounts of data.

In last years, several platforms for large-scale processing have tried to face the problem of Big Data [35]. These platforms try to bring closer distributed technologies to standard users (e.g., engineers and data scientists) by hiding the technical nuances derived from distributed environments. On the other hand, Big Data platforms also require additional algorithms that give support to relevant tasks, like big data preprocessing and analytics. Standard algorithms for those tasks must be also re-designed if we want to learn from large-scale datasets. It is not a trivial thing and presents a big challenge for researchers. The first framework that enabled the processing of large-scale datasets has been Map Reduce. This tool has been intended to process and generate huge datasets in an automatic and distributed way. By implementing two primitives, Map and Reduce, the user is able to use a scalable and distributed tool without worrying about technical nuances such as failure recovery, data partitioning or job communication. Apache Hadoop[3] has emerged as the most popular open-source implementation of Map Reduce, maintaining the aforementioned features. In spite of its great popularity, Map Reduce is not designed to scale well when dealing with iterative and online processes typical in Machine Learning and stream analytics. Apache Spark[4] has been designed as an alternative

[3] `http://hadoop.apache.org/`

[4] `http://spark.apache.org`

to Hadoop capable of performing faster distributed computing by using in-memory primitives. Spark is built on top of a new abstraction model called Resilient Distributed Datasets (RDDs). This versatile model can control the persistence and the partition of data effectively and efficiently.

Toolkits

During the various research works the following toolkits have been employed for the development and evaluation of the proposed solutions.

Scikit-learn[5]. It is a Python library which provides implementations of supervised and unsupervised algorithms, and metrics to evaluate results.

Keras[6]. It is a high-level neural networks API, written in Python. It was developed to rapidly build Deep Learning architectures by pre-developed modules. It also has the advanatge that can be run on both CPUs and GPUs.

Stanford CoreNLP[7]. It is an integrated NLP toolkit that provides a set of language based tools to analyze textual resources. It can perform part-of-speech (PoS) analysis, words lemmatization, sentences mark up, syntactic dependencies detection, and so on. It has been mainly employed as a server that has been interfaced by using the Pyhton library *stanfordcorenlp*[8].

NLTK[9]. It is a Python framework that provides interfaces to several language-based corpus such as WordNet, and text processing tools for tokenization, stemming, parsing and so on of textual resources. It is similar to Stanford CoreNLP, with the difference that it is ready to be included within Python projects without the use of interfaces.

SpaCy[10]. It is one more Python library that has been used within the research presented in this thesis. It is an alternative of Stanford CoreNLP and NLTK.

NetworkX[11]. This is a Python library that allows the creation and manipulation of graphs, and implements many functions of complex networks. It includes efficient data structures for graphs, standard graph algorithms, and analysis measures.

Pandas[12]. It is a Python library that provides easy-to-use data structures and data analysis tools. It allows to efficiently manage great matrices of data and perform classic operations that are typically adopted on tables (e.g., *join* and *selection* of rows based on their values).

[5] `https://scikit-learn.org/stable/`
[6] `https://keras.io/`
[7] `https://stanfordnlp.github.io/CoreNLP/`
[8] `https://github.com/Lynten/stanford-corenlp`
[9] `https://www.nltk.org/`
[10] `https://spacy.io/`
[11] `https://networkx.github.io/`
[12] `https://pandas.pydata.org/`

Chapter 3

Healthcare Domain

3.1 Open Issues

During the last decades a lot of data have been collected in textual clinical datasets representing patients' health states (e.g. medical reports, treatment plans, laboratory results, clinical records, surgical transcriptions, researches results etc.). Hence, digital data available for patient-oriented decision making has extremely grown but is not often mined and analyzed. Therefore, efficient access to information becomes hard for end-users [36]. In order to overcome text data overload and transform the text into useful and understandable source of medical knowledge, automated processing methods are required. Undoubtedly, this data can be exploited for figuring out relevant insights for the healthcare industry through Data Mining and Machine Learning techniques [37]. In fact, they can work as a potential base for developing recommender systems which employ documents as items, and try to suggest diagnosis for new patients who present a clinical state similar to those that have been previously evaluated.

Technologies as Data Mining, NLP, and Machine Learning can provide novel alternatives to explore and exploit potential retrieved knowledge from historical medical records, and help doctors to prescribe medication correctly to decrease medication errors effectively. In fact, text and Data Mining approaches have been already employed in healthcare for saving time, money and life [38, 39, 40]. Knowledge based techniques and tools, if reliable, can support medical staff in diagnosis, prevention and treatment of diseases, providing suggestions based on past medical cases. This chapter shows how to build a content-based recommender system within the healthcare domain leveraging Semantic Web technologies, Cognitive Computing tools, and Machine Learning methodologies.

Moreover, tests on a real dataset are reported, showing enhancements in embedding Semantic Web and Cognitive Computing tools. We examined which features better detect distinct characteristics from texts, and result suitable to cluster medical documents in order to provide high quality recommendations.

3.2 Background

3.2.1 Biomedical Information Retrieval

It would be impossible to enumerate the numerous medical questions dealt with computational approaches for clinical enhancements. Here, we focus on an overview of the most interesting and promising Data Mining and Machine Learning methods, and their applications, to discover insightful information from textual data in order to support the development of a novel content-based recommender system.

In recent years, many retrieval tools have appeared and have been used on textual resources for extracting relevant and insightful semantics [41, 42]. These tools usually exploit statistical techniques, even though there have been recently based on open linked data and Machine Learning techniques. Medical text processing is not a new question, but extracting biomedical data into a well-defined structural storage still remains a complex task [43]. Dealing with various medical domains does not help the development of systems to support medical activity. Because biomedical information is continuously being created in textual form more than ever before, there have been a lot of efforts for coding information into databases, and developing automatic processes which aim at finding useful ways to represent and organize data [44]. Medical text processing on medical domain, in particular using NLP approaches, has been explored into many other works [39, 40]. In general, researchers have usually tried to overcome text-depending issues focusing on classic entity recognition and text disambiguation techniques to create a domain-specific semantic content for the analysis of medical reports [45, 43, 12].

To alleviate textual inherit issues, some proposals have started to adopt Semantic Web practices in medical systems development. The first competition [46] on medical text-mining was run in 2002 during the Knowledge Discovery in Databases (KDD) Challenge Cup. Participants faced with a curation problem for assessing medical documents from the FlyBase dataset in order to determine whether a document should be curated based on the presence of experimental evidence of Drosophila gene products. Exploiting Part-of-Speech (POS) tagging and semantic controls determined by examining the training documents and by focusing on figures captions, a collection of manually constructed rules obtained best results on the presence of experimental evidence for the document clustering [47]. In [48] the authors used a Support Vector Machine which was trained on MEDLINE abstracts to distinguish abstracts containing information on protein-protein interactions, prior to curate this information into their BIND database. They used a bag-of-words model with classification techniques, and discovered that classifiers could minimize the number of abstracts that practitioners employed to read by about two-thirds.

Authors in [49] have proposed a new concept-based model which exploits various text mining approaches and their combinations for improving text clustering. They propose a labeler which evaluates the semantic contribution of each word in sentences, outperforming traditional methods and discovering that the semantics

is less sensitive to noise. More recent approaches are based on semantic analysis which enables learning more accurate features defined by means of external knowledge bases. In [50] authors make able systems to face with challenges by exploiting cultural and linguistic background knowledge for better interpreting unstructured documents and reasoning on their content. In [51] an item recommender system has been provided for recommendation tasks of various resources (e.g., movies and books) exploiting Word Sense Disambiguation techniques based on WordNet lexical ontology for mapping contents by means of synsets. Similar techniques are studied today in medical domain.

3.2.2 Biomedical Classification

In this section, we present classification methods which have been adopted for dealing with unstructured clinical notes over past years.

Classification is a fundamental component in the biomedical domain due to its widespread utility in applications such as medical diagnosis and identification of genetic causes of disease. In [48] authors exploit various classification techniques as described in section 3.2.1. One more approach on MEDLINE documents was proposed by [52] where authors applied a semisupervised spectral approach technique for clustering contents over two types of constraint: must-link constraints on document pairs with high (MeSH)-semantic or global-content similarities, and cannot-link constraints on those with low similarities. The authors proved the good performance of their new method on MEDLINE documents, improving performance of linear combination methods and several well-known semisupervised clustering methods.

Authors in [53] experiment multi-label classification techniques by means of combinations of bag-of-words models, and adopt time series and dimensionality reduction approaches on the MIMIC II dataset. In [54], authors implemented a Support Vector Machine classifier on n-gram features retrieved from clinical notes of the Beth Israel Deaconess Medical Center to identify the mechanical ventilation and diagnosis of neonatal and adult patients. A Convolutional Neural Network classification approach has been proposed by [55] to build models which enable to generate context based representation of health related information at sentence level. Predefined disease labels have been adopted by [56] to classify free text clinical notes. They propose two techniques Sampled Classifier Chains (SCC) and Ensemble of Sampled Classifier Chains (ESCC), which extend their dataset with selected labels in order to obtain a relationship between disease and classification.

Performances of some classification methods applied on clinical notes have been recently evaluated in [57]. Authors focused on feature selection techniques investigating different approaches of transformation methods in order to improve the multi-label classification task. They report advantages of using filtering techniques and hybrid feature selection methods. One more recent work where classification methods have been evaluated is [58]. The best results have been obtained when a hierarchical approach to tag a document by identifying the relevant sentences for

each label has been exploited.

3.2.3 Biomedical Clustering

In this section, we describe clustering methods applied to biomedical texts, and discuss recent works.

The clustering is the unsupervised task of finding groups of similar items by segmenting a collection into partitions called clusters, where items in the same cluster are more similar to each other than those in other clusters. In our work, biomedical text clustering items are medical reports. In general, document clustering can show various insights considering different levels of granularity of texts (i.e., clusters can be composed by whole documents, paragraphs, sentences or terms). In this case, the clustering can be employed as a tool for organizing and browsing documents in order to enhance the retrieval of information [59]. In biomedical domain, it could be essential to investigate patterns of a set of medical reports on features of different stuff so that similar patients can be treated concurrently in similar way.

An interesting medical document clustering has been proposed by [60] where authors exploited an ontology-based term similarity to index terms in a set of medical documents. They used a spherical k-means clustering algorithm on PubMed documents sets in order to evaluate the proposed similarity technique.

In [61] authors employed the KNN clustering method for evaluating a new similarity measure based on the semantic connection between words of an electronic medical reports set. Authors in [62] performed cluster analysis on medical posts of online health communities for recognizing various types of content. They found that clusters can be associated to common categories as treatments, procedures, medications and so on. A framework based on clustering analysis has been developed by [63] for exploring health related topic automatically in online communities integrating data with medical domain specific knowledge.

Features as biomedical concepts and semantic relationships were identified with the help of ad-hoc ontologies for building a graph representation in order to enhance the recognition of categories by means of clustering techniques in [38].

3.2.4 Biomedical Recommendation

In literature, several systems refer to medicine for identifying active relations of new patients states with past ones, but few of them exploit natural language or text mining for accomplishing recommendation tasks. In [64], authors describe a recommendation procedure which uses similarity measures for finding relations between online users' health data and medical information of Wikipedia to increase patients' autonomy in their personal health. The task to predict future health risks by means of a recommendation technique has been proposed by [65], where authors developed an engine called CARE in order to predict the future diseases risks of patients. To provide more accurate and personalized medical recommendations, authors in [66]

mined emotions from previous users' ratings adopting a topic model technique for developing a system named iDoctor. To engender advances on health recommender systems, the ACM Conference on Recommender Systems hosted a workshop in years 2016 and 2017 where specific-purpose health recommender systems have been presented, but no one focused on textual resources as narrative medical reports. In addition, these systems deal with clinical data in order to provide specific online services which target patients as end-users, but there are not systems which exploit data for supporting diagnosis fruition and physicians' work.

3.3 Problem Statement

The problem we targeted in this work regards the finding of the best representation model and clustering algorithm that enable to recognize medical issues in short texts, in order to develop a recommender system able to support physicians in their work. The clustering problem can be defined as follows: given a set of medical documents $D = \{d_1, \ldots, d_N\}$, we want to compute an assignment $\gamma : D \to \{1, \ldots, K\}$, with K being the resulting number of clusters that minimizes the objective function. The objective function is defined in terms of distance between documents. The objective is to minimize the average distance between documents and their centroids or, equivalently, to maximize the similarity between documents and their centroids.

Documents are first mapped into a N-dimensional space depending on the occurrences of their high-level features. Then a Truncated Singular Value Decomposition is used to reduce the N-dimensional space and to obtain a model where each document is mapped to a numerical vector that represents its fingerprint. Fingerprints are fed to the clustering algorithms by testing both Euclidean and Cosine dissimilarity functions, and a certain number of clusters is thus generated. The metrics we have analyzed consider the effectiveness of the resulting clustering of fingerprints and how well the generated clusters include documents pertaining the same topics.

3.4 Data Description

The dataset used in our work was freely downloaded from the open-source iDASH repository[1]. This is characterized by a set of anonymous medical reports written in plain text. The data set is composed by 2,362 English reports and each of them is characterized by specific words or a specific health domain. On the average, each report contains 400 words (the shortest document has 138 words and the longest document has 1,048 words). Reports can be medical transcription samples including clinical notes, medical examinations, care plans, and radiology reports of individuals. Examples of transcriptions include admission and discharge notes, surgical transcriptions, outpatient clinical encounter, emergency visit notes, echo-cardiogram, nuclear

[1] https://idash-data.ucsd.edu/

medicine, allergies and so on.

The dataset consists of a collection of non labeled medical reports. Each document may be assigned to one or more categories, but there is not explicit indication in the dataset documentation of the used taxonomy or how many categories one report refers to. The category of each report lies within its file name. File names generally include the disease or/and the related body part although they do not follow precise structure or patterns. Also, it is possible that different file names have words which are synonyms and, therefore, they should be considered in the same category (e.g., there are reports concerning heart issues whose file names may have prefixes such as cardiac, heart, echo-cardiogram, cardiology etc.). The reports are various and many of them are singleton, meaning they are the only ones discussing a specific topic in the whole data set. A perfect clustering should place them as outliers.

3.5 Methodology

In this section, we describe the modules of the developed system which needs proper techniques for representing items and comparing new and old users' health states. An overview of our system is depicted in Figure 3.1. The reader can see:

- *Medical Reports Collection.* This is the set of reports on which the system can learn about past clinical historical cases.

- *Content Analyzer Module.* The module takes as an input the collection of reports and the new report that the user (e.g., a physician) wants to evaluate. It embeds various resources for mining features from textual components of medical reports.

- *Represented Medical Reports.* This is the output given back by the Content Analyzer Module. The output is formatted so that Machine Learning algorithms can be easily applied.

- *Machine Learning Module.* This module implements a set of classification and clustering algorithms that are used for building models which describe patients' profiles.

- *Clinical Patients Profiles.* They are profiles that have been built by algorithms that had been employed in the Machine Learning Module.

- *Recommender Module.* The Recommender Module matches the new medical report features with the known patients' profile in order to make a list of recommendations.

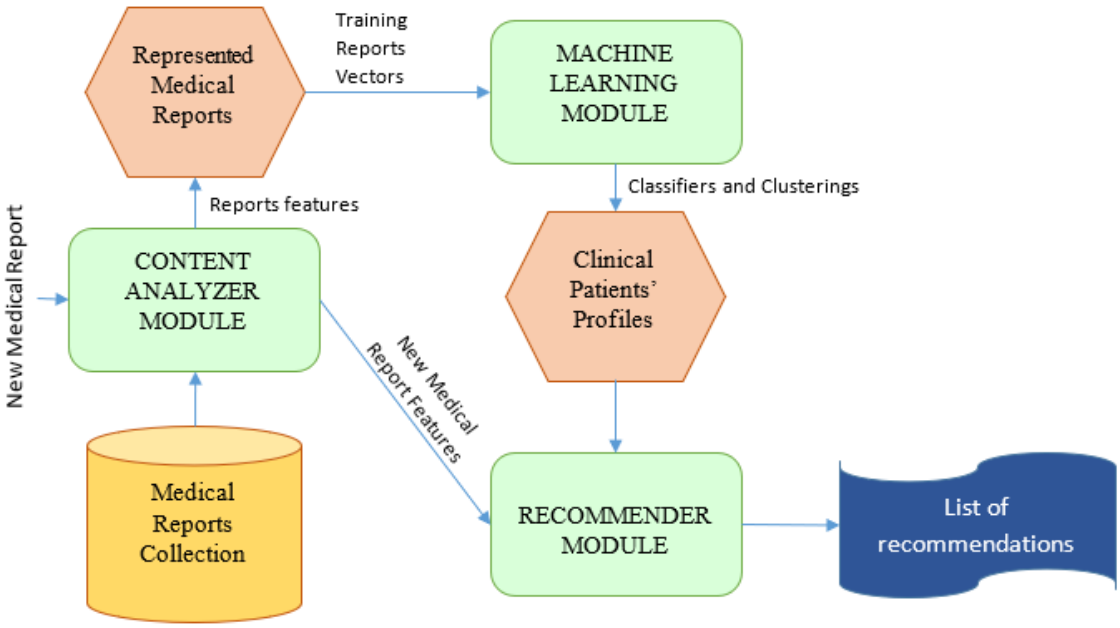


Figure 3.1: Architecture of the Content-Based Recommender System.

3.5.1 Content Analyzer Module

The Content Analyzer Module takes as an input the collection of unstructured medical reports and produces a structured documents representation which enables the automatic computation of Machine Learning techniques performed by the Machine Learning Module. In addition, it must mine new unknown medical reports. In this section, the description of the model, the features and their characteristics are described.

Item Representation

For applying Machine Learning algorithms, data must be represented by sets of features usually called attributes. For example, to recommend books, attributes adopted to describe a book can be authors, editor, genre etc. When items are described by the same set of attributes and there are known values of these attributes, they are represented in structured data that can be employed for automatic computations. In case of biomedical textual documents there are not well-defined attributes, and textual features can raise difficulties when the system learns about patients. The main problem is that traditional term-based method can fail to capture the semantics of clinical states of patients. For example, if more words can be used to indicate the same pathology (e.g., *tumors* could be indicated with the names *neoplasms, malignancies* etc.) relevant information can be lost if two clinical profiles do not contain the same word. In this context, semantic analysis of data plays a significant role and promises surprising results for solving these issues. More

specifically, we have employed words coming from IBM Watson which is a leading Cognitive Computing tool, and Framester a novel hub between semantic resources. Subsequently, in this section we show features that can be extracted from medical textual resources and discuss about advantages of each one.

Vector Space Models.

A Vector Space Model (VSM) is a spatial representation. For example, in a word-based VSM each document is represented over a N-dimensional space, where each dimension corresponds to a word that belongs to the whole set of terms of the given collection of documents. Let $D = \{d_1, d_2, ..., d_k\}$ be a collection of medical reports and $A = \{a_1, a_2, ..., a_n\}$ the set of attributes employed for representing them. A can be built by means of a natural language process or semantic content exploration pipeline which applies methods (e.g., the English stop word and stemming steps) for representing D. Each medical report d_i is represented by a vector of values $d_i = \{v_{1i}, v_{2i}, ..., v_{ni}\}$ where each value v_{ki} indicates the degree of relation between the attribute a_k and the document d_i. Attributes can have various natures such as words, n-grams, semantic features which describe contents, and so on. In our recommender system we have employed 6 different types of attributes: TF-IDF scores, Concepts, Keywords, Entities, BabelNet Synsets, and Frames.

TF-IDF

This module exploits the TF-IDF metric to weigh words. In order to prevent that longer texts have higher probability to be chosen by a recommender system, TF-IDF values are usually normalized in a range [0,1].

To avoid that frequent and no-relevant data (e.g., words that do not carry any meaning for the medical purpose as articles *the, a, an*, preposition *about, therefore, at,* etc.) appear in the TF-IDF features, the module performs some cleaning steps on the input texts. It precisely removes numeric data, punctuation, and stop-words. In fact, they are considered unnecessary and their remotion serves for (i) reducing the size of the VSM and (ii) for the subsequent efficiency of using a smaller space of features. All terms are taken in their lower case shape, avoiding to consider more times different representations of the same word (e.g. *Cardiac* and *cardiac*).

Concepts

Concepts can be defined as cognitive units which model perceived abstract subjects. They depend on the ability to process domain dependent knowledge and efficiently learn insights which become fundamental keys in the meaning of contents. Concepts can embody structures and representation of real words discovered in text, hence, they enable capturing high level abstraction reducing the complexity of the computation space. Moreover, they enable the specialization of employed attributes

for representing documents in the VSM. IBM Watson can be employed for discovering automatically concepts related to the medical domain from natural language texts. It assigns a weight to each concept we have used for building the VSM. More precisely, given a collection of medical reports we use as set of attributes a set that contains the union of almost fifty concepts returned by IBM Watson from each medical report.

Keywords

Keywords are words of texts that enable listing the content of a report, releasing information about which words result relevant for describing the content of a document. Keywords are automatically detected by IBM Watson which provides a weight for each one. The VSM model is built as in the case of concepts.

Entities

Entities are actors that make actions in a text. Specifically to the medical domain, they can be people (e.g., physicians or nurses), illnesses (e.g., tumor), medicine names and so on. By capturing entities, it is possible to find relations between different documents if they share similar actors, especially when they are specific (for example if in a subset of documents D' physicians are cardiologists and in another subset D'' they are physiotherapists, the entities are distinct and enable better separation of the document subsets in different topics). As with the previous IBM Watson features, a weight is returned for each entity and indicates its influence in a document.

BabelNet Synsets

BabelNet synsets are unique unambiguous identifiers of sets of words which share the same meaning. We have chosen this type of synsets because (i) they are the result of the integration of various linguistic and semantic resources as WordNet, Wikipedia, FrameNet, among others, and (ii) they are directly provided by Framester. Differently from IBM Watson features, we do not have weights, hence, only the presence of BabelNet synsets have been considered by means of boolean flags into the Content Analyzer Module.

Semantic Frames

A semantic frame is a coherent group of concepts such that complete knowledge about one concept within a context depends on the knowledge associated to all others in the same context. Given a text, they are activated by nouns and verbs. Each frame can have multiple hierarchical levels that indicate its abstractions. For example, in Figure 3.2, the word *cardiology* is abstracted by frames *Medical_specialties*

and *Cure*. It should be underlined that frames are different from IBM Watson concepts because they do not depend on the application domain, but on relations that words have into linguistic and knowledge resources Framester adopts. As with the BabelNet synsets, we use the frames presence in the VSM.

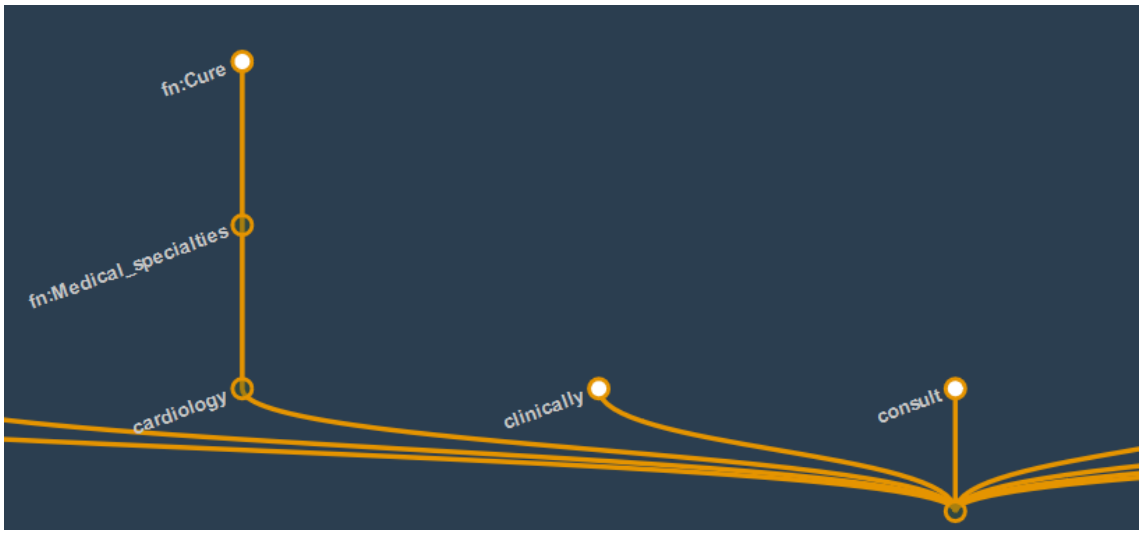


Figure 3.2: Part of Framester result on the sentence *"Consider cardiology consult and further evaluation if clinically indicated"*.

The course of dimensionality problem

The course of dimensionality problem refers to the issue that regards the great size of the number of attributes required to describe the target collection. The VSM suffers of this problem, hence, it needs to be managed into content-based applications as our recommender system. One common method often applied in order to solve the issue is the Singular Value Decomposition (SVD). In details, let $A = \{a_1, a_2, ..., a_n\}$ be the set of attributes and $D = \{d_1, d_2, ..., d_i\}$ be the collection of our documents. The VSM is usually represented by a matrix M of size $|D| \times |A|$. M can be disjointed in three components $M = USV^T$ where S is a diagonal matrix containing the largest singular values, U is a matrix where columns are left singular vectors, and V is a matrix where columns define right singular values. In order to reduce complexity of data, the module applies a truncation which consists in holding only the largest k singular values, removing others which can be considered less relevant. This technique is known in literature as Truncated-SVD (TSVD). The module adopts the matrix $M' = U \times S$ which has a number of rows equivalent to the number of considered documents with a smaller number of attributes (columns) than the original matrix M. Besides decreasing the overall computational costs, an advantage of using the TSVD is deleting noise elements that might deteriorate the list of final recommendations. We want to point out that the value of k requires a trade-off between the amount of remaining and neglecting data to avoid the loss of

information. Its value depends on the set of attributes which characterize the used collection.

3.5.2 Machine Learning Module

The Machine Learning Module receives a VSM as an input and returns a model which recognizes clinical patients' states. Its current version includes two clustering algorithms which are applied on all VSMs. The clustering techniques the module implements enable to deal with unsupervised data. They are (i) Hierarchical clustering algorithm and (ii) K-means clustering algorithm which have been described in section 2.2.

Precise clustering requires an accurate definition of the closeness between documents represented in the VSM. The closeness can be measured by either the pairwised similarity or distance. A variety of similarity or distance measures have been proposed and discussed in literature. Our Machine Learning Module adopts the Cosine and Euclidean measures.

3.5.3 Recommendation Module

This module uses the clinical patients' profiles for suggesting possible past medical cases that are similar to a new one by matching the new medical case against clinical profiles' clusterings of medical reports to be recommended. More specifically, the Recommendation Module takes a new medical report representation r and predicts whether there are clinical patients' profiles $p_1, ..., p_n$ that are interesting according to the relevance with r. It performs strategies to rank documents, and top-ranked ones are included in the list of recommendations that are provided to the final user. For doing so, the module computes the closeness between a new medical reports and clusters. In detail, given a new patients' medical report r and a clustering $C = \{c_1, ..., c_n\}$, the module finds the cluster c_i which has the closest center to r. Then elements within c_i are ranked from the most to the least similar to r. The produced ranking is used for finding the closest k medical reports as the final recommendation list.

3.5.4 Experimental Setup

Data cleaning

Data cleaning is necessary in order to provide the same valid English text to the Content Analyzer Module services. First of all, we have cleaned all medical reports from HTML tags, removed all tables and structured format styles in order to obtain simple plain texts. Then we have matched reports words against those provided by WordNet, sending the word w' and getting the word w'' which has been placed

in the text. At the end, only English text with correct grammar and punctuation composes the collection of medical reports.

Content Analyzer Module Setup

Report Name	Myocardial infarction	Heart	Atherosclerosis	Obesity	Cardiology	Cardiovascular system	Atheroma	Hypertension
heart-catheterization-ventriculography-angiography	1	1	1	0	1	1	0	0
cardiac-catheterization	1	1	1	0	1	0	1	0
cardiovascular-letter	1	0	1	1	0	0	0	1

(a)

Report Name	Myocardial infarction	Heart	Atherosclerosis	Obesity	Cardiology	Cardiovascular system	Atheroma	Hypertension
heart-catheterization-ventriculography-angiography	0.97	0.95	0.87	0	0.68	0.60	0	0
cardiac-catheterization	0.96	0.62	0.51	0	0.50	0	0.53	0
cardiovascular-letter	0.96	0	0.85	0.25	0	0	0	0.94

(b)

Report Name	Myocardial infarction	Heart	Atherosclerosis	Obesity	Cardiology	Cardiovascular system	Atheroma	Hypertension
heart-catheterization-ventriculography-angiography	4	6	4	0	4	2	0	0
cardiac-catheterization	3	3	2	0	2	2	2	0
cardiovascular-letter	4	0	3	3	0	0	0	3

(c)

Figure 3.3: Samples of VSMs built on concepts extracted from three medical reports. Samples are related to (a) binary (b) weighted and (c) counted VSM.

The Content Analyzer Module has been configured for providing more VSMs models which have been built on various features as described in section 3.5.1. More precisely, let r_i be the i-th medical report and f_j be the j-th feature of a selected type. The outcomes of the module are:

- **5 Binary VSMs**: they include a matrix representation for the Concepts, Keywords, Entities, BabelNet Synsets and Semantic Frames features. Binary means that if f_j occurs within the inferred set of features of the medical reports r_i, in the VSM model M their relation is indicated by $M[i, j] = 1$, otherwise $M[i, j] = 0$;
- **4 Weighted VSMs**: they include a matrix representation for the Concepts, Keywords, Entities, and TD-IDF features. Weighted means that $M[i, j] = weight$, where $weight$ has been calculated exploiting the Natural Language Understanding service of IBM Watson or the TF-IDF approach as described

above, and represents how strong is the relation between the medical report r_i and the feature f_j, otherwise $M[i, j] = 0$;

- **5 Counted VSMs**: they include a matrix representation for the Concepts, Keywords, Entities, BabelNet Synsets and Semantic Frames features. Counted means that $M[i, j] = count$ where $count$ is the number of times that a feature f_j occurs within the set of features of the medical reports r_i, otherwise $M[i, j] = 0$;

For more details on the three mentioned distances, the reader can look at examples of VSMs built on concepts extracted from three medical reports of the test dataset in Figure 3.3. In the first row of each VSM, there are concepts that form the N-dimensional space. In the other rows, there are the names of reports on the first column followed by values that indicate the degree of relation between the medical report and the i-th concept. The reader notices that (a) is built using the *binary* relation, (b) is built using the *weighted* relation and (c) is built using the *counted* relation.

Machine Learning Module Setup

The Machine Learning Module applied both clustering methods on all VSMs. In order to obtain high quality clusters, we set the module for exploiting the Silhouette width measure. Given a cluster c, its Silhouette width value $s(c)$ is computed as showed in Equation (3.1) where $w(c)$ is the average dissimilarity within c and $o(c)$ is the lowest average dissimilarity of c to any other cluster.

$$s(c) = \frac{o(c) - w(c)}{max\{o(c), w(c)\}} \tag{3.1}$$

Values of Silhouette width range from -1 to 1. When the value is closer to 1, it means that the clusters are well separated; when the value is closer to 0, it might be difficult to detect the decision boundary; when the value is closer to -1, it means that elements assigned to a cluster might have been erroneously assigned. In general, we can consider good clusterings those that have high average values of Silhouette width. Unsurprisingly, the value of the Silhouette width depends on the type of features of the VSM under processing.

Hierarchical clustering. After the hierarchical clustering has been computed, the resulting dendrogram has been iteratively cut starting from its head, in order to increase the number of clusters for each iteration. In doing so, various clusterings obtained with different cut values have been produced. As a reminder, in our dataset we do not know how many groups can be formed. Therefore, we have exploited the highest value of average Silhouette width values in order to cut dendrogram where the clustering showed the best separation between medical reports.

K-Means clustering. K-Means Clustering has been performed with different values of k as number of clusters. For each value of k, the average Silhouette width

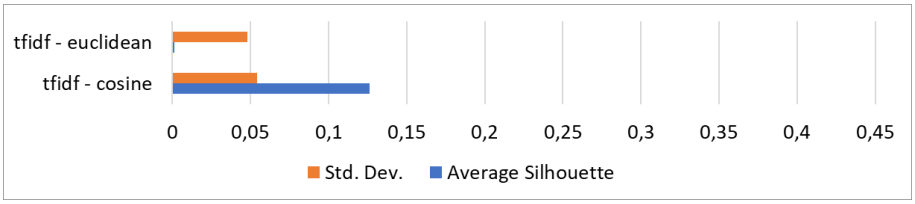
measure has been computed similarly to hierarchical clustering setup. Then the clustering with the highest average value of Silhouette width has been hold as the output of the module.

3.5.5 Recommendation Module Setup

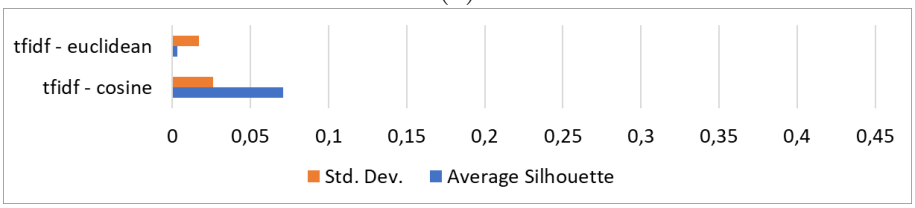
The recommendation module has been setup to receive an unknown medical report and a number k which represents the number of recommendations. In our experiments the adopted value of k is 10.

3.6 Results and Discussion

At the current state, the quality of results of our recommender system mainly depends on the Content Analyzer Module and Machine Learning Module. In fact, a good quality of clusters means that medical reports similar to a new one can be correctly detected in the test dataset. First, for obtaining good clustering the features must allow a good separation of reports, and second the clustering algorithm must recognize which the best divisions are. Therefore, in this section we discuss about the most representative features of our dataset and the clustering algorithms performance. Results of clusterings quality can be observed in Figure 3.4, 3.5, 3.6, 3.7 and 3.8.



(a)

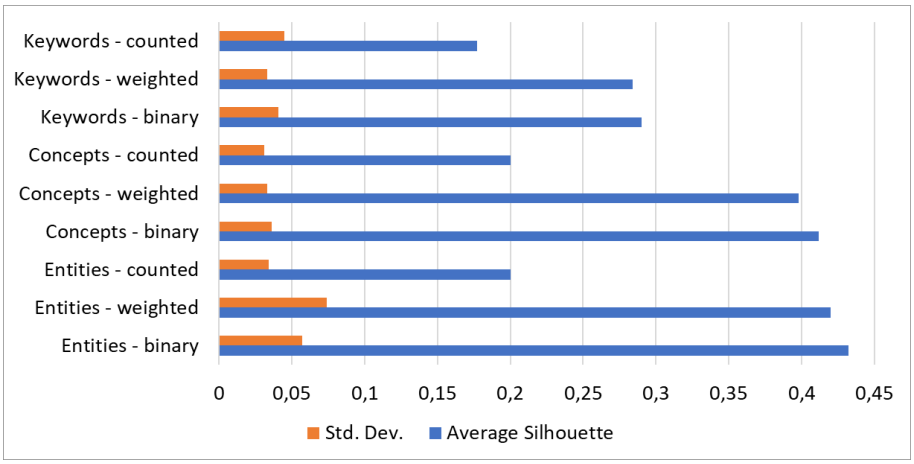


(b)

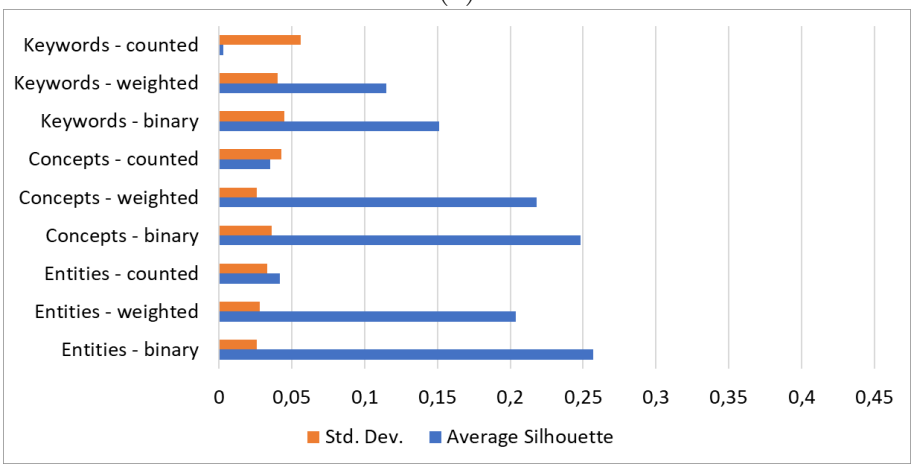
Figure 3.4: The average and standard deviation values of the silhouette width measure of the clusterings computed on the TF-IDF measure. (a) Hierarchical clustering. (b) K-means clustering.

The sets of features which have formed the best division of medical reports into clusters are those that have been computed using IBM Watson. In fact, they reach good levels of Silhouette width. In more details, concepts and entities in their

weighted and binary mappings have shown good performances in capturing medical information from medical reports of the test dataset. This fact suggests that the relevance of an entity or a concept into a medical report does not depend on the number of times that it appears. We can say that their role depends on the relations they have into reports, and more influent their actions are, stronger their relevance is. Considering the number of times that a concept or an entity appears we do not add any additional information in our representative VSM. Keywords have not shown good performances like entities and concepts, but they can be considered as good alternatives in those cases where detecting entities and concepts can be hard.



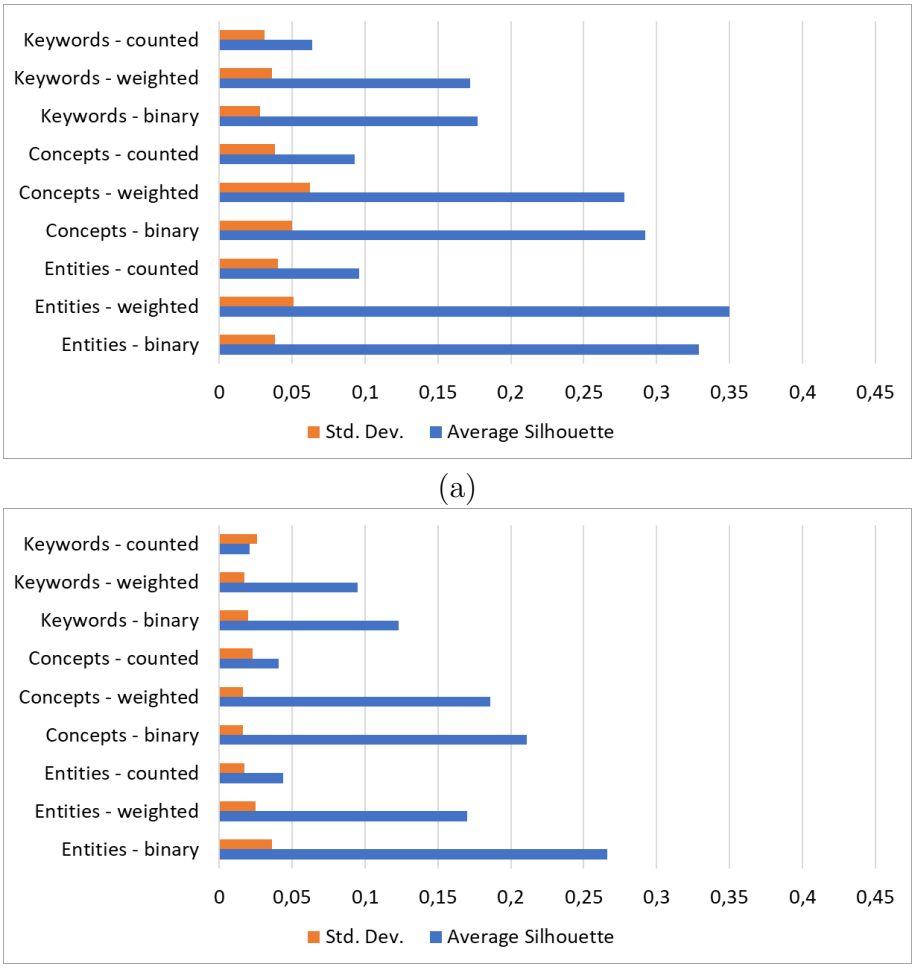
(a)



(b)

Figure 3.5: The average and standard deviation values of the silhouette width measure of the clusterings computed on IBM Watson features. (a) Hierarchical clustering on cosine distance. (b) Hierarchical clustering on Euclidean distance.

Framester features do not have reached good results in the clusterings. This can depend on the fact that they are more generic and not directly connected to the



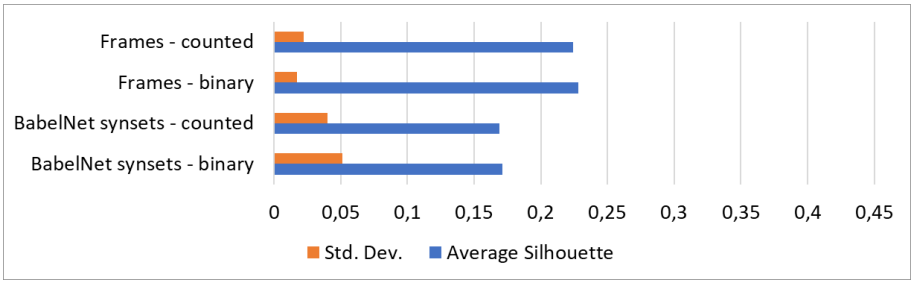
(a)

(b)

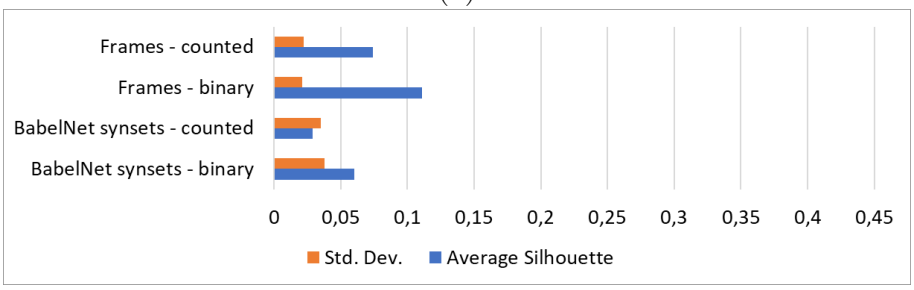
Figure 3.6: The average and standard deviation values of the silhouette width measure of the clusterings computed on IBM Watson features. (a) K-means clustering on cosine distance. (b) K-means clustering on Euclidean distance.

medical domain. Moreover, our test dataset could negatively influence this types of features since medical reports are strongly specific on patients' medical states. By contrast, they can result useful for those medical reports that describe the state of patients more in general without too clinical details (e.g., a starting examination visit). As for Framester features, the TF-IDF has not showed good performances and similar explanations can be observed.

One more point to consider is how the distance between two medical reports is computed. Results suggest that the cosine distance is more reliable than the Euclidean distance. Nevertheless, it is important underlying how they seem keeping a similar behavior on different features. To name an example, *entities-binary* and *entities-weighted* show a similar behavior both for cosine and for Euclidean distance.



(a)

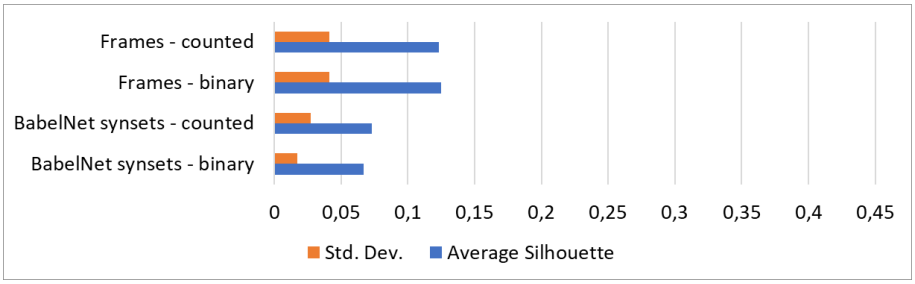


(b)

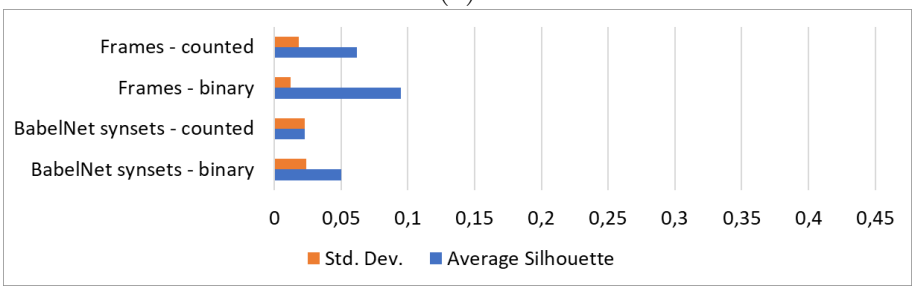
Figure 3.7: The average and standard deviation values of the silhouette width measure of the clusterings computed on Framester features. (a) Hierarchical clustering on cosine distance. (b) Hierarchical clustering on Euclidean distance.

Hierarchical clustering has outperformed the results of the K-means clustering, hence, if the recommender system would have been employed on a real medical case, the hierarchical clustering should be used. The agglomerative approach seems to be more suitable for finding medical cases similar to a new one.

Finally, to show how recommendation module has worked the reader can look at example in Figure 3.9. The Figure lists 10 medical reports of our test dataset that are returned by our recommender system when the report called *heart-catheterization-angiography-1* has been adopted for the evaluation of a new patient's clinical state. The example shows how returned recommendations are correlated to heart issues and, hence, that our approach in building a recommender system can effectively recognize medical contents in order to suggest relevant past clinical cases.



(a)



(b)

Figure 3.8: The average and standard deviation values of the silhouette width measure of the clusterings computed on Framester features. (a) K-means clustering on cosine distance. (b) K-means clustering on Euclidean distance.

```
heart-catheterization-angiography-2
coronary-ct-angiography-ccta-2
heart-catheterization-ventriculography-angiography-6
stenting
cardiac-catheterization-8
heart-cath-coronary-angiography
nuclear-cardiac-stress-report
gen-med-consult-4
heart-catheterization-ventriculography-angiography-4
cardiac-catheterization-1
```

Figure 3.9: An example of list of recommendations built by our recommender system using as new report that called *heart-catheterization-angiography-1.*

Chapter 4

E-Learning Domain

4.1 Open Issues

The ever-increasing availability of digital data brings unprecedented possibilities to analyze different educational facets. Platforms and services available to support education and emerging technologies are reshaping how people learn in their everyday life, leading the global market of off-the-shelf education to growth and evolve towards online learning at scale [67]. More and more individuals and teams are leveraging it to cultivate new skill sets and achieve personal or collective goals throughout their careers. At the same time, leading providers are offering large-scale on-demand access to their collections of online courses with varied contents, structures, requirements, objectives, instructors and prices [68]. Learners, teachers, designers and managers have been mobilized to investigate how data can be used to support learning and teaching. For instance, existing providers would automatically organize their online courses according to meaningful taxonomies which facilitate retrieval, instructors would find emerging teaching topics and develop courses on appealing subjects on the basis of the latest trends and, even more, learners would be driven along the overwhelming alternative courses. The solutions required to address such challenges in online E-Learning resources delivering at scale depend on advanced technological infrastructures for data storage and computation, and should consider users' interaction.

This intense interest has given rise to tools and techniques in the research field of Learning Analytics (LA) [69]. Notable examples include the identification of learners at risk of failing [70] and the analysis of data flows coming from the interactions among communities [71]. The field is in its dynamic youth, and there are numerous opportunities to unlock the potential of Learning Analytics, especially with the rapid development of emerging Artificial Intelligence technologies.

In the large set of educational data, micro-learning videos embedded into Massive Open Online Courses (MOOCs) represent one of the most powerful medium to deliver knowledge to learners in small chunks over time, promising to have a solid

future in delivering knowledge as proved by the recent experiences in [72]. In this context, the most popular providers are currently Coursera[1], EdX[2] and Udemy[3].

One challenge that has been raised is the need of Learning Analytics tools powered by intelligent methods for effective and efficient video categorization. Recent studies [73, 74] tend to model the problem as a text classification task which automatically assigns one category from a set of predefined ones to a video based on the transcript (i.e., a written version of the content presented by the speaker). These approaches usually map video transcripts using Term-Frequency-Inverse Document Frequencies (TF-IDF). Even if it is simple and widely-used, several limitations have emerged. In fact, text documents are modeled as a set of term frequencies, regardless the position in the document or semantic links with other words. It follows that the knowledge derived by the information behind the text is lost. Cutting-edge cognitive computing systems, such as IBM Watson, can extract more insightful information such as concepts, emotions, entities, keywords, and relations from unstructured text, and use Machine Learning algorithms to derive analytics, and generate predictions and hypothesis.

Despite the initial outcomes, different evolving problems remain to be solved. For instance, Machine Learning techniques need to combine both descriptive and content-based course features which require high-level semantic understanding. Even more, online courses at scale come with various languages, so algorithms are supposed to master cross-language capabilities. Similarly, recommendations targeted to learners should match their desired content with their requirements, goals, pedagogical, economic and temporal constraints.

One more obstacle to take these directions and provide the required technological services is the lack of suitable datasets. More in details, to build a high-level semantic understanding, we need fine-grained information about courses. To test cross-language capabilities, we require courses provided in different languages. To provide meaningful recommendations, we need stakeholder interactions within courses. However, no dataset currently meets such conditions.

Furthermore, one more interesting evolution that has innovated the field of the E-Learning is due to the advent of Social Web, which has enabled the development and the sharing of experiences among people around the world. In fact, individuals use online social platforms to express opinions about products and/or services in a wide range of domains, influencing the point of view and the behavior of their peers. Such user-generated data, which generally come in form of text (e.g., reviews, tweets, wikis, blogs), is often characterized by a positive or negative polarity according to the satisfaction of people who write the content. Online educational platforms deployed at large scale, are hence earning more and more attention as social spaces where students can discover and consume a great variety of contents

[1] `https://www.coursera.org/`
[2] `https://www.edx.org/`
[3] `https://www.udemy.com/`

about many topics, and share opinions regarding their educational experience. Such collective intelligence might be useful for various stakeholders, including peers who are planning to attend a given course, instructors who are interested in improving their teaching practices and increasing students' satisfaction, and providers who can get benefits from the feedback left by users to refine tools and services in the platform itself. With this in mind, these platforms can be envisioned as dedicated social media where discussions are limited to specific topics concerning course content quality, teachers' skills, and so on [75]. Sentiment Analysis approaches on students' opinions have recently started to receive the attention of the involved stakeholders [76] and their design and development is still an open challenge.

This chapter tackles the open challenges that have been exposed above. More precisely, the contributions within the E-Learning domain can be summarized as follows.

E-Learning Data Collection. We collected two datasets and released them to the research community. We introduce a dataset of only features that can be used to manage contents by Machine Learning algorithms. We employed this dataset for the contents categorization task. Then, we present COCO, a novel semantic-enriched Collection of Online COurses composed by more than 43K courses distributed in 35 different languages, involving over 16K instructors and 2,5M learners who provided about 4,5M ratings and 1,2M comments. We describe the collection procedure and the dataset structure together with some statistics. Furthermore, we provide two potential use cases where COCO can be handy, highlighting the issues which need to be faced.

E-Learning Contents Categorization. We describe how Cognitive Computing technologies can be adopted to support the development of Learning Analytics tools by investigating (i) how we can extract and merge features extracted from micro-learning videos to improve their representation in the eyes of Machine Learning algorithms, and (ii) which Machine Learning algorithm is best at taking advantage of such features in terms of effectiveness and efficiency in micro-learning video classification.

Sentiment Analysis of Learners Reviews. We propose a Deep Learning model, trained on Word Embedding representations coming from the E-Learning context and able to predict a sentiment score for reviews posted by learners. We also report its experimental evaluation on the large-scale dataset of online course reviews present in COCO. We show how word embeddings trained on smaller context-specific textual resources are more effective with respect to those trained on bigger general-purpose resources. Moreover, we highlight the benefits derived from the combination of word embeddings and Deep Learning instead of common Machine Learning approaches.

4.2 The COCO Dataset

4.2.1 Dataset Collection

The Udemy APIs[4] expose functionalities to help developers accessing content and building external applications. However, they are instructed to list only a subset of the over 40K courses Udemy includes. Consequently, we developed a Selenium[5] crawler in Python to access the full Udemy catalog and build a complete and comprehensive dataset. We dumped it in November 2017.

The crawler is instructed to access the course catalog sublists associated to each category of the Udemy taxonomy, so that we first extract the association between each course and the corresponding categories while getting the link to the course description page. Each course has one first-level category and one second-level category. Each second-level category belongs to only one first-level category. Unlike traditional academic taxonomies, the courses are mapped by Udemy in daily-life-oriented categories (e.g. *Lifestyle*, *Language*, *Test Preparation*). Furthermore, each course is also described with a set of fine-grained tags. We extract the association between courses and tags using the same methodology previously employed to extract categories. However, the same course can appear in the course catalog sublist of more than one tag in that case.

Then, the crawler goes inside the description page of each course. To get an idea, an example course description page is made available here[6]. Each course description page presents the course identifier, the heading with the title and the short description of the course, aggregated statistics about received ratings and enrolled students, and the language in which the course is delivered. Udemy provides courses in more than 30 different languages. Then, different HTML boxes contain a bullet-list of course objectives (e.g. *build powerful fast user-friendly web apps*, *apply for high-paid jobs or work as freelancer*), a bullet-list of both pedagogical and technical course requirements (e.g. *Javascript and HTML fundamentals, a laptop with at least 6GB RAM*), a long course description of around 500 words, and a bullet-list of possible target users (e.g. *students who want to learn how to build reactive web apps*) written by the instructors. The course description pages also include the list of lessons and their organization in chapters. Each lesson has a title, a 30/50-word description, and a format (e.g. *video* or *document*). Only a subset of lessons is freely available as preview. We collected their resource URL and, eventually, the URL of the video transcript. Furthermore, the course description pages list one or more instructors together with their id, job title and short biography. On the left-side, a HTML box depicts the current price. The crawler digests all such information.

To extract the learners reviews, the crawler uses the Udemy API method aimed to return course reviews given the course identifier. Each review includes the learner

[4] `https://www.udemy.com/developers/`

[5] `http://www.seleniumhq.org/`

[6] `https://www.udemy.com/spark-and-python-for-big-data-with-pyspark/`

id and the course id together with the timestamp, the rating ranging between 0 and 5 with step 0.5 and, optionally, a textual comment. It is worth to note that learners give their comments in their own language, but no language label is provided to keep it. The mentioned API method does not release information regarding the instructor replies to learners reviews, so the crawler digests a copy of the same course reviews from the list presented at the bottom of the course description page. Differently from the mentioned API method, the course description page does not depict the review timestamps, but shows the instructor replies to such reviews. Then, the two copies of the same course reviews are merged. Moreover, the course description page depicts the full name of the learner who has made a given review. The crawler uses it on the fly to build the URL of the public profile of the learner and access it. Each public profile shows the courses where learners are enrolled and the wish-list in the case they have given consent to publicly share them. Finally, we label all the human-made textual attributes with their own language using Lang Detect[7], a free reliable language detector.

Course attributes and interactions with them embrace a wide range of human-made-based texts such as comments, requirements, objectives, descriptions, and video transcripts. Manipulating them in Machine Learning methods tailored for online courses requires high-level semantic understanding, going beyond traditional item-frequency features. To facilitate future experiments and comparisons, we first enriched such attributes with TF-IDF features extracted by Scikit-Learn v0.19. Then, to stimulate research in high-level semantic understanding algorithms, we collected the features extracted by state-of-the-art cognitive tools. More precisely, we enriched the textual attributes with the following additional feature sets.

- Part-of-Speech (PoS) tags computed by the NLP tools of the toolkit NLTK.
- Keywords and concepts computed by the state-of-the-art Cognitive Computing tools included into the IBM Watson Natural Language Understanding APIs. Each keyword is a set of one or more words relevant in the text, while each concept captures cross-domain content not explicitly cited.

4.2.2 Dataset Description

Structure of the Dataset

COCO is a JSON-based collection whose structure in terms of entities and associations is depicted in Figure 4.1. Text attributes have Unicode coding, while languages and timestamps hold ISO639-1 and ISO8601 standards, respectively.

In COCO, *Course* is the most informative entity. First, *id* and *course URL* provide unique identification attributes. Then, the course is described by *short* and *long descriptions*. *Requirements* and *objectives* list technical and pedagogical needs

[7] `https://pypi.python.org/pypi/langdetect?`

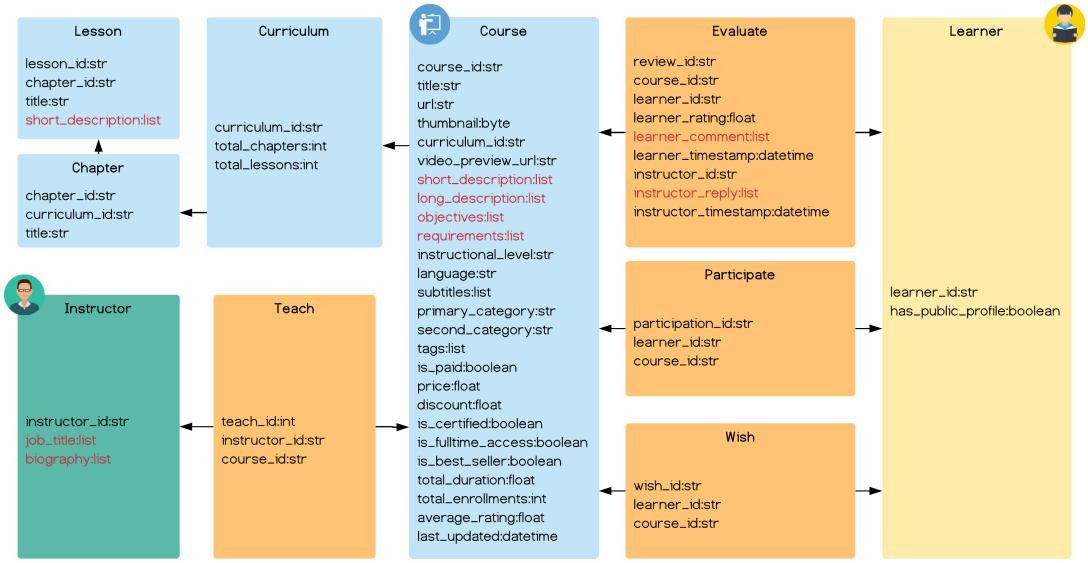


Figure 4.1: *COCO Structure.* Boxes in green, blue and yellow show primitive entities. Orange boxes depict associations. The attributes in red are enriched with semantics.

at the beginning and expected learner skills at the end, respectively. The *language*, the *instructional level* (*beginner*, *intermediate*, *expert*), *first/second-level categories*, and *tags* are listed. Each course has only one first-level category and one second-level category, while tags can be more than two for the same course. Other course fields identify the current *price* and the *discount*. Statistical attributes list the *estimated course duration* in hours. Finally, some boolean flags indicate *certification release* and *lifetime access availability*. The *Curriculum* entity includes a hierarchical list depicting the *chapters* and their *lessons*.

Due to privacy constraints, the *Instructor* and *Learner* entities only include information available on the corresponding public profiles. Each entity instance is uniquely identified by a fake *id*, so that the id stored into the dataset does not correspond to the real id of the user. Each instructor is described by the job title and biography. Each learner has a flag indicating whether the profile is public.

The COCO strength is the large amount of relationships among primitive entities. In *Teach*, the pairs of *instructor id* and *course id* model the association among instructors and the courses they teach. One instructor can teach more than one course and the same course can have one or more instructors. Then, each pair of *course id* and *learner id* in *Participate* defines the courses that the learner has been attending. In *Wish*, the id pairs set the courses each learner has inserted into the wish-list. Finally, *Evaluate* contains *learner id* and *course id* together with the [0-5] *rating* with step 0.5, the *comment* and the *timestamp*.

Statistics

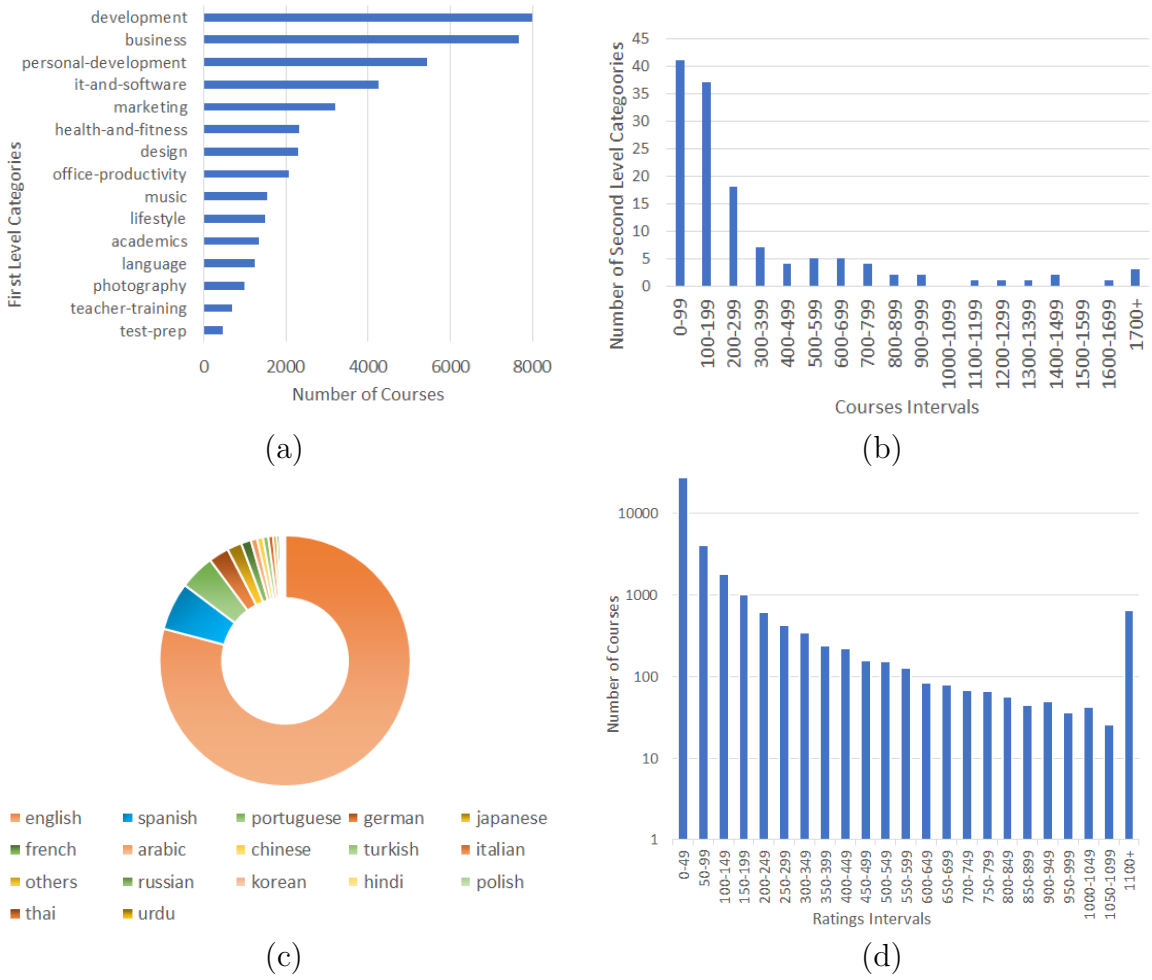
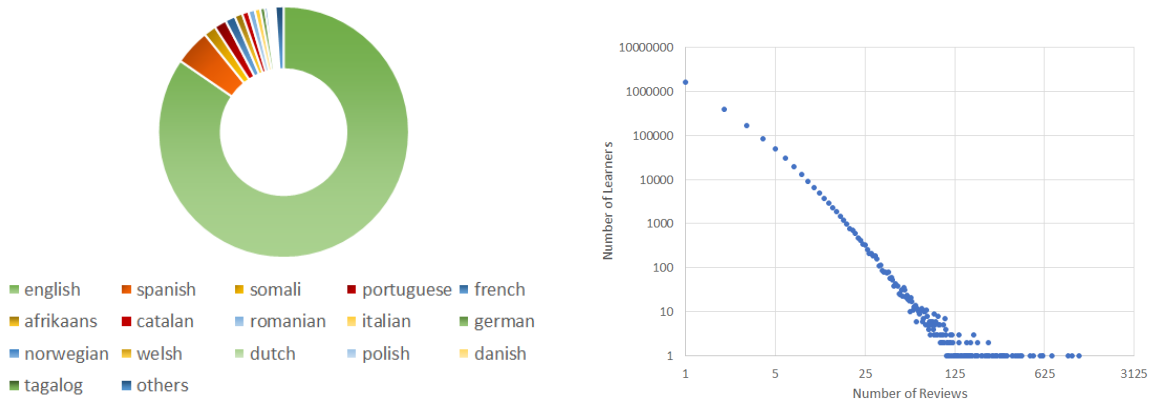


Figure 4.2: The distribution of courses per (a) first-level category, (b) second-level category, (c) content language, (d) number of learners reviews.

Describing COCO in numbers, it includes 43,113 courses distributed into a taxonomy composed of 15 first-level categories and 133 second-level categories, as reported in Figure 4.2 (a,b). The courses distribution is unbalanced for both first-level categories (avg. 2,874; st.dev. 2,334; min 477; max 7,985) and second-level categories (avg. 324; st.dev. 475; min 7; max 3,196). Similarly, the languages distribution along courses follows such trend, as depicted in Figure 4.2(c). The languages employed in at least 25 courses are explicitly named. Only 21% of courses do not use English as primary language. Regarding the courses structure, each course contains 43 lessons in average (st.dev. 46; min 1; max 863).

In COCO, there are 2,546,865 learners who provided 4,584,313 ratings and 2,453,865 comments. The sparsity of the rating matrix is 0.99583%. Only learners



(a) (b)

Figure 4.3: The distribution of learners per (a) review language, (b) number of reviews.

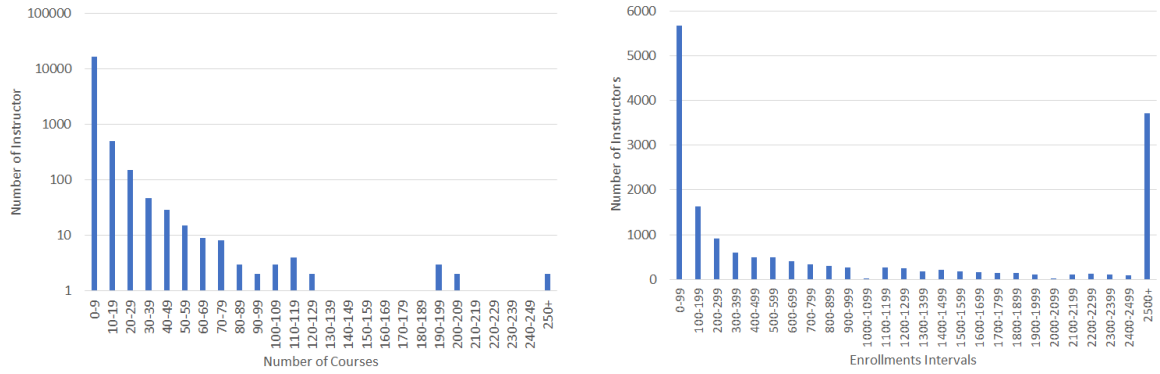
with at least one rating are included, while each course can have zero or more ratings. In Figure 4.2(d), the distribution of the number of ratings per courses (avg. 119; st.dev. 812; min. 1; max. 57,346) shows a downward trend, but there is a large number of courses with a lot of ratings. Figure 4.3 shows the distribution of learners per (a) the review language and (b) the number of provided ratings. Despite some learners have made a lot of reviews, the average number of ratings per learner is low (avg 2; st.dev. 3; min 0; max 1,159). COCO also incorporates 16,963 instructors. Figure 4.4(a) shows their distribution based on the number of courses they teach (avg. 3; st.dev. 10; max 748; min 1). In Figure 4.4(b), the distribution of instructors according to the number of learners enrolled into their courses appears divided in two blocks, with a peak of the number of instructors with few enrolled students (avg. 4,036; st.dev. 19,238; min 1; max 850,496).

4.2.3 Experimental Use Cases

This section depicts two potential use cases made possible by having COCO and a set of experiments to demonstrate how they are as promising as challenging.

Multi-Class Content-Based Course Classification

Multi-class classification assigns each course to one category chosen among a set of different options in a pre-defined taxonomy. E-Learning domain is semantically challenging and hardly leverages several services that other domains have already exploited. The automated classification makes easier both the categorization and the exploration of courses. Given a set of training course records $D = \{d_1, ..., d_n\}$ such that each one of them is labeled with a category c_i of a set $C = \{c_1, ..., c_m\}$,



(a) (b)

Figure 4.4: The distribution of instructors per (a) courses and (b) learners they manage.

multi-class classification is a supervised task aimed to infer a model $f : D \longrightarrow C$ that relates each course record in D to a category in C. Then, the model is able to predict the category for a course whose category is unknown. For the evaluation, we included the most successful algorithms, namely Support Vector Machine (SVM), Decision Trees (DT), Random Forest (RF) and Naive Bayes (NB) [77]. Their implementation was provided by the Scikit-learn library.

Table 4.1: The best classification results with first-level category as target.

Source Attribute	**Algorithm**	**Features Type**	**W-P**	**W-R**	**W-F1**
Long Description	*SVM*	*Nouns-TF-IDF*	*0.79*	*0.78*	*0.77*
Short Description	SVM	TF-IDF	0.61	0.61	0.60
Objectives	SVM+SGD	TF-IDF	0.74	0.73	0.72
Requirements	NB	Nouns-TF-IDF	0.57	0.47	0.43

First, minority categories were over-sampled to account for unbalanced categories. Then, the evaluation protocol worked as follows. For each setting, we adopted 10-fold stratified cross-validation, maintaining the original category distribution in training and test sets. For each fold, the algorithms were fed with each features set extracted from each course attribute in red in Figure 4.1. Then, the performance was evaluated using weighted precision (W-P), recall (W-R) and f-measure (W-F1). Each metric was first calculated for each category, and the average is found, weighted by the number of true instances for each category.

Table 4.1 reports the best results that were obtained in the considered experimental settings. The results provide empirical evidence that the TF-IDF generally

produces better results than high level features. The poor performances of the latter ones could indicate that they are not suited to capture the fine-grained information. Although they clearly capture the most relevant characteristics of each course at the human eyes, the results are not the same when they are analyzed by machines; therefore, advanced semantic enriching models and techniques need to be studied to get meaningful insights from semantic information.

Course Recommendation

Recommender systems can be one of the solutions proposed to navigate among the overwhelming alternative courses. We considered the sets of users U, courses C, ratings R in the range [0-5], supposing that no more than one rating can be made by any user for a given item, writing this rating as r_{ui}. The most popular task in recommender systems is to predict the rating score. The goal is to learn a model $f : U \longrightarrow C$ that predicts the rating f_{ui} of a user u for a new item i. For the evaluation, we employed Normal Predictor (NP) as baseline, SVD, SVD++, NMF, Slope-one, Co-Clustering. Their implementation is in Surprise[8]. The ratings R are divided into a training set R_{train} used to learn f and a test set R_{test} to evaluate prediction accuracy with Root Mean Squared Error (RMSE).

The average RMSEs for various algoritms with their default parameters on a 5-folds cross-validation procedure are depicted in Table 4.2. The folds were the same for all the algorithms and the random seed was set to 0. The results highlight that SVD++ performs the best among all the investigated scenarios. Its performance is significantly better than the baseline Normal Predictor. SVD++ shows an improvement of about 0.28 on RMSE compared to such baseline. However, the results still need to be improved; in this direction, advanced semantic enriching techniques which leverage content-based course information in addition to the ratings can be a viable solution to get better prediction results.

Table 4.2: The rating prediction performance measured with Root Mean Square Error.

Metric	**NP**	**SVD**	**SVD++**	**NMF**	**Slope One**	**Co-Clustering**
RMSE	1.051	0.7796	*0.7755*	0.8334	0.8582	0.9595

4.2.4 Existing E-Learning Datasets

Datasets have been frequently used in technology-enhanced learning. They differ in terms of size and shape, domain, and context of user interaction. Here, we discuss only the most prominent alternatives.

[8] `http://surpriselib.com/`

The Dataset of Joint Educational Entities (DAJEE) [78] includes about 20K resources extracted from 407 online courses distributed in 10 first-level categories and 36 second-level categories. Over 484 academic instructors are mentioned. However, the authors built an ontology aimed to detect patterns in academic teaching. No interaction among learners and courses is listed together.

The Technology Entertainment Design (TED) dataset [79] contains around 1K talks and 69K users who made more than 100K ratings and 200K comments. This collection embraces only resources and no educational information such as course requirements and objectives is given.

Multimedia Education Resource for Learning and Online Teaching (MERLOT) [80] is a collection of free and open online resources contributed by an international education community. It includes over 40K materials with 19 different material type categories. It incorporates a variety of resource types collected from face-to-face learning lessons. It does not include feedback on learners' preferences and interactions between them.

The HarvardX-MITx Person-Course Dataset [81] includes interactions in 17 MITx and HarvardX courses on edX platform. These data are aggregated records representing individual activities in one course. They combine several learner information (e.g. *degree*, *gender*, *birth date*) and provide data on interactions within courses (e.g. *number of viewed activities*, *number of published posts*). The data granularity and the number of courses limit the applicable analysis methods.

The Metadata for Architectural Contents in Europe (MACE) dataset [82] provides metadata-based access to learning resources in repositories all over Europe. It offers access to about 150,000 learning objects, holding together about 47,000 tags, 12,000 classification terms and 19,000 competency values.

4.3 Categorization of E-Learning Contents

4.3.1 Background

Learning Analytics

Existing Learning Analytics approaches typically exploit data generated by users during normal interactions with E-Learning technologies [83]. Several studies focused on the prediction of either the students at risk of failing or the students' grades [84, 85]. However, they tend not to consider how the performance of algorithmic techniques at the basis of Learning Analytics tools influences the students' behavior during common E-Learning tasks. For instance, they investigate whether providing a taxonomy of videos helps students, without considering the performance of the underlying classification algorithm which is essential as well. Moreover, no deep semantic exploration of the resources selected as appropriated for a given learning context is performed. It follows that modern Learning Analytics tools should

consider the content of resources in addition to the interaction with them, especially to find the most appropriate ones.

In the next generation of personalized learning environments, it is essential to provide resources tailored to the learner's need while integrating interactions, skills and competencies with the mapping of knowledge of disciplines [86]. In this direction, powering Learning Analytics tools with content-based resource analysis promises to support content managers during video organization, and learners during search in well-organized educational environments.

Video Content Analysis

Multimedia classification and indexing are two tasks required to organize and store resources so that they can be quickly retrieved. Several Machine Learning techniques try to automatize these tasks in an accurate and non time-consuming way.

However, it is well-known that the usual characteristics of a video presented in form of visual frames and audio tracks make the classification harder in relation to text content. In order to address this issue, researchers tend to use video metadata [87] and video content information such as visual frame sequences [88, 89], audio tracks [90], transcripts [91, 92], or multiple combinations of them [93, 94, 95]. In spite of good results obtained using low-level features, emerging semantic-based alternatives have a huge potential to better describe video content.

Recently, some studies have been focused on higher level features to model the content of videos. In [96], the authors presented a novel system for content understanding and semantic search on a video collection based on low and high level visual features. In a similar way, [97] investigated the problem of detecting or classifying single video-clips by means of the event occurring in them, which is defined by a complex collection of elements including people, objects, actions, and relations among them. The authors in [98] proposed to generate queries on videos exploiting Automatic Speech Recognition (ASR) and Optical Character Recognition (OCR) directly. However, we would like to point out that the type of video represents a relevant characteristic to better understand its content. Hence, information channels used for a reliable analysis should be selected in relation to the way the information is mainly transmitted. In micro-learning videos, the greatest amount of knowledge is transmitted by voice. Therefore, we have chosen video transcripts as source of information.

Analysis of audio tracks as textual transcripts is an attractive way to be exploited as the most rich channel of information for micro-learning videos. For example, state-of-the-art results in this direction can be observed in [99], where authors used ontology-based classifier (e.g., CSO classifier [100, 101]) to mine the knowledge contained in video subtitles. In this direction, we investigate how novel feature types extracted from transcripts by Cognitive Computing tools can improve videos classification. In contrast to the previous works, we face video content analysis exploiting the Natural Language Understanding capabilities provided by IBM Watson,

enriching TF-IDF results with semantic information. Embedding the service in our approach, we then focus on the analysis of various feature sets and classification algorithms to achieve better classification performance.

4.3.2 Problem Statement

The problem we have targeted in this work was to develop a framework which is able to assign a video resource to a class by using the video transcript. The solution of this problem provides a tool which can support the categorization, and consequently, the organization of resources in a E-Learning platform. In details, the problem has been addressed through a supervised approach. Given a set of videos $V = \{v_0, ..., v_n\}$, each one labeled with a class c_j of a set $C = \{c_0, ..., c_m\}$, we want to compute a function $\gamma : V \rightarrow C$. Videos have been fed into the function γ through vectors representations exploiting both TF-IDF and the IBM Watson suite. We have experimented different types of semantic video transcript representations and supervised algorithms, studying which combination of algorithms and data representations better work for solving this task.

4.3.3 Methodology

In this section, we describe the approach for micro-learning video classification. Figure 4.5 depicts its components and how they work together.

The Classification Manager orchestrates the overall process. First, it calls the Pre-Processing Module to extract the transcripts from the videos included into the dataset. Then, transcripts are handled by the Feature Extraction Module which computes the corresponding features. During training, the Classification Manager sends both features and category labels to the Classifer Module. During testing, only features are sent, and returned categories are matched with the original ones stored in the dataset to evaluate the performance. We detail all the modules in the following subsections.

Big Data Manager

We have employed Apache Spark 1.6.1 and MLlib library. MLlib is the Spark's Machine Learning library aimed at making practical Machine Learning scalable and easy. It includes common learning algorithms and utilities, such as classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as lower-level optimization primitives and higher-level pipeline APIs. Including it makes our approach general and flexible. We can easily use any classifier available within the MLlib library, whose code is already optimized with the Map-Reduce paradigm to run over a cluster. We might also use any other classifier of other libraries not specific for Apache Spark.

Pre-Processing Module

This module takes a micro-learning video as an input and returns the cleaned version of the associated transcript as an output. Given a micro-learning video v, the module sends it to the Speech-to-Text service[9] of the IBM Watson suite, which combines grammar rules with knowledge of audio signals for translating the spoken language of an audio track in its written form. Its choice depends on the fact that it is one of the most accurate and easily manageable speech-to-text tools [102]. The service returns a plain text *t(v)* corresponding to the transcript of v. The module converts all the words in *t(v)* to lowercase and each one of them is compared with the WordNet[10] dictionary in order to get the correct one, *w'*, if w contains errors; otherwise, w and *w'* have the same value. Each word w in *t(v)* is substituted in *t(v)* with the correct *w'*, obtaining *t'(v)*. Finally, the module removes stop-words from *t'(v)* and returns it as cleaned transcript.

Features Extraction Module

The Features Extraction Module has the purpose of representing a micro-learning video with a set of features. It takes a cleaned transcript from the Pre-Processing Module as input and returns a set of pairs where the first element is the identifier

[9] https://www.ibm.com/watson/developercloud/speech-to-text.html
[10] https://wordnet.princeton.edu/

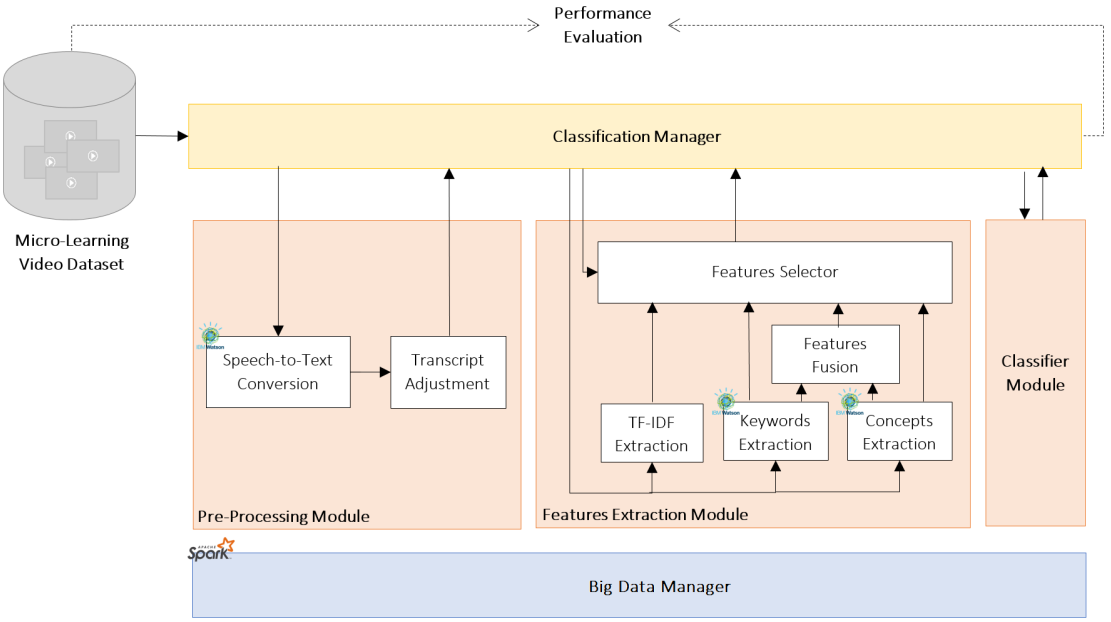


Figure 4.5: A schema for the proposed approach for micro-learning video classification and how the components work.

string of the feature and the second element is the relevance of that feature for the corresponding transcript. The relevance value spans in the range $[0, 1]$ where a value closer to 0 represents a low relevance and a value closer to 1 represents a high relevance of the corresponding feature for the transcript.

From the transcript, the module can extract TF-IDF features and high-level features. It returns one of these feature sets according to a flag set up into the Feature Selector submodule. More precisely, for the high-level features, the module calls the Natural Language Understanding API provided by the IBM Watson suite. Currently, the module exploits only concepts and keywords, but it can be easily extended to a wider set of features. This service works well for semantic content extraction since it contains a wide range of pre-trained models, and can process large samples of actual human language to derive rules governing the natural language in a sentence. In our case, the Features Extraction Module shapes keywords and concepts into vectors. The first one includes important topics typically used when indexing data. The service identifies and ranks keywords directly mentioned in the text. The second one represents concepts not necessarily referenced in the text, but with which the text transcript can be associated. Both for concepts and keywords, the service computes a weight that indicates the relevance we exploited in our vectors. These feature sets enable our approach to perform high-level analysis than just TF-IDF.

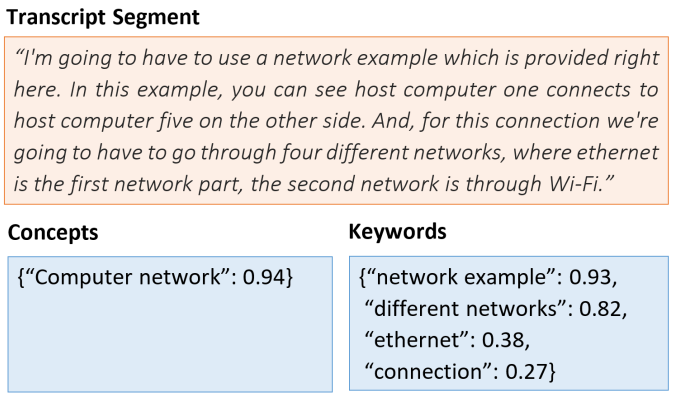


Figure 4.6: Example of extraction of features with IBM Watson.

Given a cleaned transcript *t'(v)* as returned from the Pre-Processing Module, the module first computes for each word $w \in t'(v)$ its TF-IDF value building the TF-IDF vector *tf-idf(t'(v))*. Then, it sends *t'(v)* to the Natural Language Understanding service and requests to obtain a concepts vector *c(t'(v))* and a keywords vector *k(t'(v))*. The service returns them as JSON data. For each vector, a list of pairs is returned where the first element of each pair is the string identifier and the second element is the relevance associated to it in the range [0,1]. After that, the module builds a unique feature vector *kc(t'(v))* by concatenating *c(t'(v))* and *k(t'(v))*. As an example, we consider a short segment of a video transcript about computer networks. The text and the corresponding concepts and keywords ex-

tracted using IBM Watson are shown in Figure 4.6. "Computer network" is the only concept extracted even though it is not directly mentioned in the text. A number of four keywords were returned. In the case of a collision between two identifiers from concepts and keywords vectors, the module computes the mean between the associated relevance values and stores a single instance of the identifier accordingly.

Classifier Module

The Classifier Module aims at finding the most appropriate category for a given video using the underlying classifier trained on a number of labeled samples. The classifier implements a function $f(t'(v)) \rightarrow c$ where $f(t'(v))$ is the features vector of $t'(v)$ and c is the category returned by the classifier and of the given taxonomy. The module can implement any classification algorithm, independently from the feature type. In particular, our approach tests DT, SVM, RF, and SVM+SGD since they are the most widely used algorithms as emerged from literature review. However, our approach does not depend on this design choice and any classification algorithm can be used.

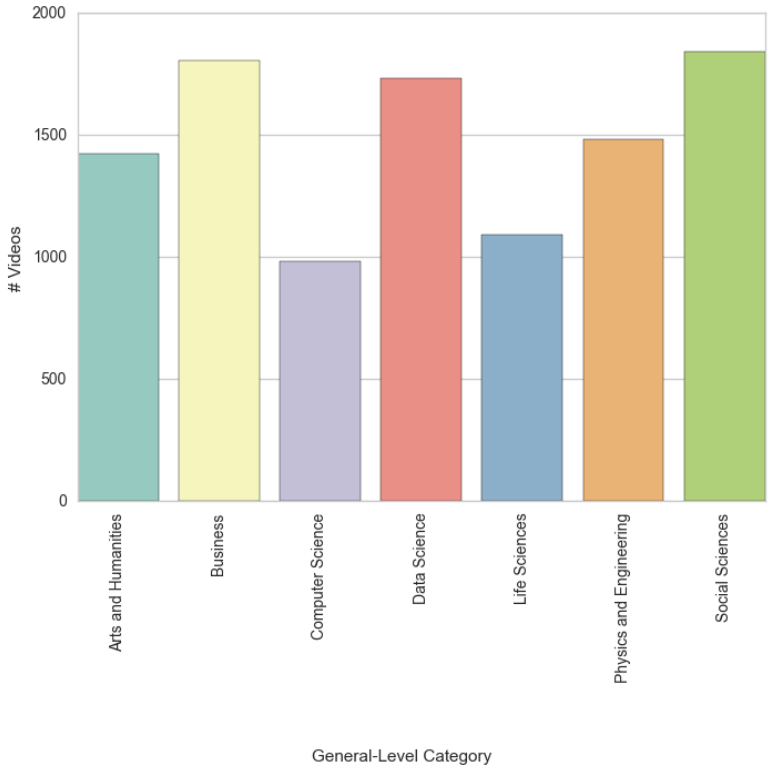


Figure 4.7: Video distribution over general categories in our extracted dataset.

4.3.4 Results and Discussion

Dataset Description

This work was performed before to collect COCO, therefore, the data used to perform the evaluation of the methodology is not the same we released with COCO. We collected one real-world micro-learning video dataset from Coursera to validate our methodology. It is worth to note that the Coursera provides a taxonomy of classes where videos are assigned to. We collected 10,328 videos from 617 courses whose primary language is English. Coursera pre-assigned the courses to 7 general-level categories and 34 specific-level categories. Coursera has a courses catalog; each course has one general-level category and one specific-level category. Moreover, each course contains a set of videos. Each downloaded video was assigned to the same categories of the course it belongs.

Each extracted video is assigned to one general-level category and one specific-level category and contains a number of words ranging from 200 to 10,000 (avg. 1,525; std. 1,017). The overall distribution of videos per category is reported in Figure 4.7 and Figure 4.8. The dataset is challenging because it contains fine-grained categories that require subtle details to differentiate (e.g., Business Essentials and Business Strategy). Moreover, video transcripts contain less words than documents typically used in text classification. The language style is different, since the transcripts are derived from speaking activities, not written.

Performance Measures

We mapped the video classification problem as a text classification problem. Hence, we adopted the most common performance measures for this task: precision, recall, and F-measure. Because we were dealing with a multi-class classification problem, we first define how to calculate them for each category.

Given a specific category c_i from the category space $c_1, \ldots, c_n$, the corresponding precision P_{c_i}, recall R_{c_i}, and F-measure F_{1c_i} are defined by the following formulas:

$$P_{c_i} = \frac{TP_{c_i}}{TP_{c_i}+FP_{c_i}} \quad R_{c_i} = \frac{TP_{c_i}}{TP_{c_i}+FN_{c_i}} \quad F_{1c_i} = 2\frac{R_{c_i}\cdot P_{c_i}}{R_{c_i}+P_{c_i}}$$

where TP_{c_i} (true positives) is the number of videos assigned correctly to the category c_i, FP_{c_i} (false positives) is the number of videos that do not belong to the category c_i, but they are assigned to this category incorrectly and FN_{c_i} (false negatives) is the number of videos that actually belong to the category c_i, but they are not assigned to this class.

Then, we need to compute the average performance of a binary classifier (i.e., one for each category) over multiple categories. There are three main methodologies for the computation of averaged metrics (see [103] for more information). They can be summarized as follows:

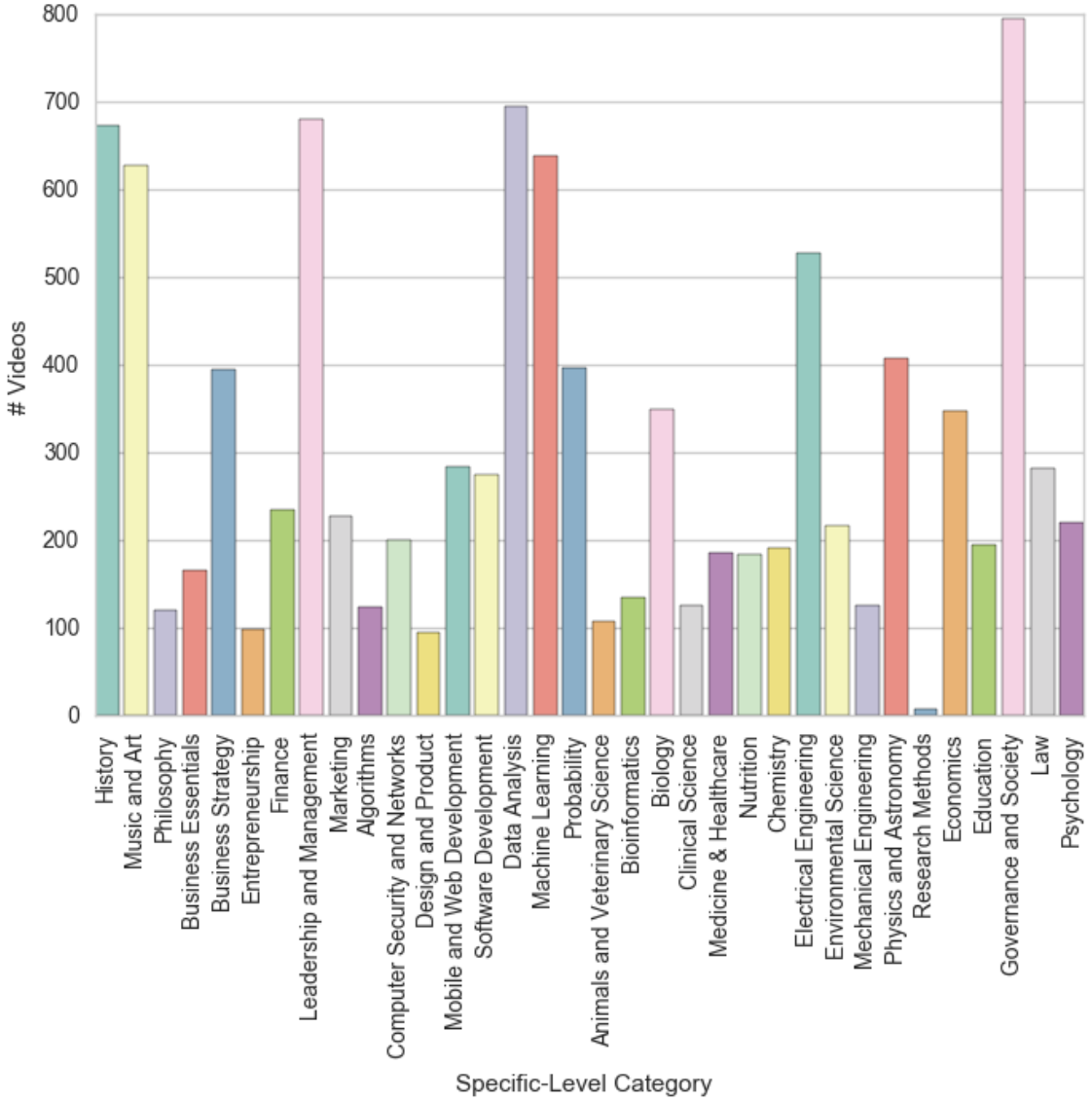


Figure 4.8: Video distribution over specific categories in our extracted dataset.

- *Micro-Averaged (Micro).* The metrics are globally calculated by counting the total number of true positives, false negatives and false positives, with no category differences.

- *Macro-Averaged (Macro).* The metrics are locally calculated for each category and the unweighted mean is computed. This does not consider categories imbalance.

- *Weighted-Averaged (Weight).* The metrics are locally calculated for each category, then their average is obtained by weighting each category metric with the number of instances of the category in the dataset. Therefore, each category does not contribute equally to the final average and some of them contribute more than the others.

To test the performances of our approach, we only considered weighted scores obtained from the combination of the metrics (precision, recall, F-measure) for each one of the two categories spaces of our dataset (general-level and specific-level).

Performance Evaluation

In this section, we focus on exploring the effectiveness and the efficiency of our approach for a certain number of combinations of the features sets, classification algorithms and metrics. The software was developed using PyCharm 2016.2 environment. We used Python as programming language. For each experiment, we adopted a 10-fold stratified cross validation, so that the folds were made by preserving the percentage of samples for each category, then the reported results were averaged over ten runs. Apache Spark was set underneath and the code was developed on top using map reduce functions that allows scalability.

Feature-Classifier Combinations In order to evaluate our approach, it was tested with a number of alternative combinations of four features representations and four classification algorithms. The features representations included TF-IDF (baseline), concepts, keywords, and the combination of keywords and concepts. While,

Table 4.3: Basic statistics about the used features sets.

Feature Set	Number of Features	Avg	Std. Dev.
TF-IDF (baseline)	117,073	260	513
Concepts	29,952	21	13
Keywords	332,023	78	26
Keywords + Concepts	361,975	99	31

Table 4.4: Computational time for both training and test phases.

Features Set	**Algorithm**	**Execution Time [s]** *General Categories*	*Specific Categories*
TF-IDF (baseline)	DT	5.23	7.92
	RF	12.20	39.81
	SVM	3.70	16.25
	SVM+SGD	**0.25**	**1.12**
Keywords	DT	3.32	7.21
	RF	11.06	43.96
	SVM	3.32	11.21
	SVM+SGD	**0.20**	**1.00**
Concepts	DT	1.13	1.88
	RF	2.38	7.67
	SVM	0.31	2.21
	SVM+SGD	**0.05**	**0.20**
Keywords + Concepts	DT	4.00	8.34
	RF	13.53	50.38
	SVM	3.95	12.84
	SVM+SGD	**0.26**	**1.06**

the classifiers were the implementations of the following algorithms provided by [104]:

- Decision Trees (DT).
- Support Vector Machine (SVM).
- Random Forest (RF).
- Support Vector Machine + Stochastic Gradient Descent (SVM+SGD).

Computational Time Evaluation We evaluated the efficiency of the proposed approach in terms of the size of the features vectors used and the time required to perform both the training and the test for a given classifier.

Table 4.3 summarizes the basic statistics with respect to the different feature sets. The first column indicates the name of the feature set, the second column shows its size, while the other two columns report the average and the standard deviation of the number of non-zero elements. The vector sizes for the combination concepts and keywords is greater than all the others. However, the average number of its

Table 4.5: Performance measures for micro-learning video classification over general categories.

Feature Set	Algorithm	Precision	Recall	F-Measure
TF-IDF (baseline)	DT	0.55	0.48	0.48
	RF	0.60	0.58	0.57
	SVM	**0.71**	**0.70**	**0.69**
	SVM+SGD	0.62	0.62	0.61
Keywords	DT	0.47	0.35	0.33
	RF	0.49	0.43	0.41
	SVM	0.67	0.65	0.65
	SVM+SGD	0.66	0.66	0.65
Concepts	DT	0.66	0.44	0.45
	RF	0.66	0.52	0.53
	SVM	0.63	0.62	0.61
	SVM+SGD	0.64	0.63	0.62
Keywords + Concepts	DT	0.64	0.46	0.47
	RF	0.63	0.55	0.55
	SVM	0.70	0.69	**0.69**
	SVM+SGD	0.69	0.68	0.67

non-zero elements is smaller than that of the TF-IDF. Hence, high-level features can be more discriminative. Table 4.4 reports the total time required for a given algorithm with a given features set to perform the training and the test phases over a single fold. In the results, SVM+SGD algorithm provided the best computational time, especially in case of concepts with no large vector size. The computational time mostly depends on the particular classifier and the number of features. For example, the computational time for SVM+SGD classifier using concepts as features takes 0.05 seconds due to the lower time required by SVM+SGD and the lower size of the concepts feature set.

Precision-Recall Analysis We evaluated the performances of all the classifiers trained on TF-IDF, keywords, concepts, and keywords plus concepts using precision, recall and F-measure. The results in Table 4.5 indicate that using keywords outperforms TF-IDF only with the SGD approach. Using concepts generally gives better results than using keywords, and as with the use of keywords, they outperform TF-IDF in case of SGD algorithm. With concepts, we obtain higher precision and lower recall. For SVM+SGD, keywords plus concepts outperform TF-IDF.

Table 4.6 shows the results of the classifiers on specific categories. In this case,

Table 4.6: Performance measures for micro-learning video classification over specific categories.

Feature Set	Algorithm	Precision	Recall	F-Measure
TF-IDF (baseline)	DT	0.52	0.46	0.46
	RF	0.59	0.58	0.56
	SVM	**0.73**	**0.71**	**0.70**
	SVM+SGD	0.69	0.58	0.61
Keywords	DT	0.48	0.34	0.34
	RF	0.49	0.43	0.42
	SVM	0.67	0.62	0.61
	SVM+SGD	0.70	0.66	0.66
Concepts	DT	0.52	0.39	0.40
	RF	0.53	0.48	0.47
	SVM	0.59	0.57	0.56
	SVM+SGD	0.58	0.57	0.56
Keywords + Concepts	DT	0.52	0.41	0.41
	RF	0.54	0.51	0.50
	SVM	0.69	0.66	0.65
	SVM+SGD	0.69	0.66	0.66

SGD classifier trained using keywords continues to outperform TF-IDF, while its performance decreases if trained on concepts. When combining keywords and concepts, the overall performance improves and, in case of SVM+SGD, can be considered better than TF-IDF with an improvement up to 5%. The worst case shows a maximum loss of 6% when RF is adopted. As far as the specific categories classification is concerned, it is worth noticing that they are not equally distributed as the general categories (as clearly shown in Figures 4.7 and 4.8) and this high variance negatively affects the classification using high-level features (keywords, concepts and their combination), especially those with a low number of features for the training.

Overall Evaluation In most cases, our approach can produce good performance, regardless the increasing algorithm complexity in terms of video size or transcript size. In fact, the first one influences only the time required to extract transcripts from videos. No assumptions can be done on transcript sizes in relation to video sizes since the number of words included into a transcript depends on the amount of content orally provided by the teacher during the video lesson. The second one has an impact on feature extraction. Training and test are not directly influenced by the transcript size because the size of each feature vector depends on the size of the word dictionary used to map features. Moreover, the number of

relevant features extracted from a long transcript can be less than those extracted from a short one due to repetitions and stop words.

Considering the trade-off between effectiveness in terms of precision, recall, F-measure and efficiency in terms of computational time, the combinations achieving best results include SVM or SVM+SGD as algorithm and TF-IDF or Keywords+Concepts as features. However, SVM+SGD algorithm is generally over ten times faster than SVM. With SVM+SGD, our approach using Keywords+Concepts outperforms that using TF-IDF in terms of precision, recall and F-measure. The combination using SVM+SGD and Keywords+Concepts achieves performance comparable with that using SVM and TF-IDF in terms of effectiveness, but strongly better in efficiency. The experimental results demonstrate that Keywords+Concepts combined with SVM+SGD can scale well, maintaining good performances in almost all cases.

4.3.5 Practical Implications

The promising experimental results inspire several practical applications of our approach worthy of future study to develop cognitive-driven Learning Analytics tools within a broad domain of educational research. This section presents a set of relevant examples.

One of the primary application domains of Learning Analytics and Data Mining techniques is the recommendation of learning materials to students. Semantic techniques can lead to the evolution from a TF-IDF-based to a concept-based representation of items and user profiles. In this context, content-based recommendation techniques can combine our classification approach with students data in order to build tailored student's profiles from their set of watched videos, by looking for most similar resources to the ones that a student has previously used. Using deep semantic content analytics, students can gain improvements in learning because they can be directly driven to resources that best fit their interests, reducing the time required for retrieval.

Since Learning Analytics makes use of different Data Mining and Machine Learning techniques for analysis of educational content, our approach has a potential for automating repetitive and time-consuming tasks usually performed by educational researchers and practitioners. These include automation of micro-learning classification processes, tagging generation and organization of learning resources. MOOC platforms usually contain a large amount of videos which need to be organized based on their content for being provided to students. Content managers need to know the concepts behind videos which should be accurately allocated into a taxonomy for a fast and qualitative organization. With our approach, micro-learning videos can be automatically placed into a predefined taxonomy, increasing the quality of their arrangement into MOOC platforms while reducing the effort required to content managers.

Another primary challenge of working with large amounts of video content is in-

dexing and retrieval. The increasing development of advanced multimedia applications requires new technologies for organizing and retrieving from content databases of digital videos. To this aim, video content must be described and adequately coded. Our approach can be easily extended to index and retrieval, allowing semantic annotation and querying in video databases. In this way, learners can be supported during the search of the most appropriate content fitting their requirements. Even more, our automated pipeline facilitates the development of video-to-video search tools, making a step forward to mitigate the well-known semantic gap problem.

Since the majority of learning forms involve use and creation of videos, several analytics techniques have been applied to investigate them. In an E-Learning platform, the success of Learning Analytics tools requires both a simple user interface to directly interact with users and algorithms supporting them for achieving and managing resources they are interested in. Our contribution provides a back-end powered by Cognitive Computing and Big Data for several Learning Analytics tool interfaces which require to work with micro-learning videos.

4.4 Sentiment Scores Prediction of Learners' Reviews

4.4.1 Background

Sentiment Analysis in E-Learning Systems

E-Learning domain has recently gained the attention of Sentiment Analysis in order to get knowledge from new dynamics that E-Learning platforms allow to. By leveraging students' emotions, it might be possible contributing to increase the students' motivation and improve learning processes. More specifically, the study of sentiments in a E-Learning platform can contribute to learning and teaching evaluation, investigate how technology can influence students' learning process, evaluate the use of E-Learning tools, and improve learning content recommendations [105].

The measurement of sentiments and emotions in such platforms should be as less invasive as possible for not disturbing the learning process and not influencing the overall opinion. Adoption of textual reviews left by students' for analyzing sentiments and emotions is one of the less invasive techniques, because data collection and analysis are completely transparent for students. In literature, various scenarios where Sentiment Analysis was used to study learning aspects using textual reviews can be found. For instance, there are works that embrace the use of Sentiment Analysis to mine students' interactions in collaborative tools, guaranteeing students' communication privacy in their opinions [106]. Another relevant area that exploited Sentiment Analysis was the teachers' assessment. For example, the authors in [76] adopted a Support Vector Machine to evaluate the teachers' performance using $1,040$ comments of systems engineering students as a dataset. The evaluation

of sentiments in the teaching-learning process was also object of study in [107]. Its authors adopted comments posted by both students and teachers. Similarly, the authors in [108] studied text sentiment to build an adaptive learning environment with improved recommendations. For example, they described how to choose a learning activity for a student based on his/her goals and emotional profile.

Deep Learning for Sentiment Analysis

Machine Learning has been extensively adopted for Sentiment Analysis tasks, using different types of algorithm and fitting various types of features. For example, authors in [109] used Maximum Entropy (ME) and Naive Bayes (NB) algorithms, adopting syntactical and semantic patterns extracted from words on Twitter. Their method relies on the concept of contextual semantic i.e., considering word co-occurrences in order to extract the meaning of words [110]. In the evaluation on nine Twitter datasets they obtained better performance when the Machine Learning algorithms were trained with their method both at tweet and entity level. More recently, authors in [111] applied NB, ME, SGD, and SVM algorithms to classify movies reviews in a binary classification problem (i.e., positive or negative evaluation of reviews). They showed that the use of a n-gram model to represent reviews with the above algorithms obtains higher levels of accuracy when the value n was small. Moreover, they showed that combining uni-gram, bi-gram, and tri-gram features enabled to enhance the accuracy of the method against the use of a single representation at once. Machine Learning methods rely on lexical syntactical features representations which are derived from text, not considering semantic relationships that can occur between words. Hence, in spite of feature engineering advancements, there has been a growth of techniques to infer semantics as Deep Learning that has emerged as an effective paradigm to automatically learn continuously information from text.

Sentiment Analysis domain also experienced the influence of the wide spread of Deep Learning approaches. For example, in [19] a CNN composed by two layers was designed to capture features from character to sentence level. An ensemble Deep Learning method was proposed by [112], where various sentiment classifiers trained on different sets of features were combined. They performed experiments on six different datasets coming from Twitter and movies reviews. With their experiments, they improved the state-of-art against Deep Learning baselines. Another approach to combine various classifiers with Deep Learning ones was also proposed by authors in [113], where a SVM classifier was mounted on top of a 7-layer CNN in order to complement the characteristics of each other and obtain a more advanced classifier. With their variation they were able to obtain more than 3% of improvement compared to a classic Deep Learning model. To learn continuous representations of words for Sentiment Analysis a combination of CNN with a LSTM was exploited by authors in [114]. They were able to assign fixed-length vectors to sentences of varying lengths, showing how Deep Learning approaches outperform common Ma-

chine Learning algorithms. Although the use of Deep Learning models have showed amazing improvements in many domains, they have not been deeply studied for applications in E-Learning, which can have benefits for exploring users' opinions.

Word Embeddings for Sentiment Analysis

Word embeddings have been successfully used in various domains, ranging from behavioural targeting [115] to Sentiment Analysis. Within the latter, they have been widely employed for improving accuracy of baselines methods not using word embeddings. As traditional word embeddings methods do not usually take into account words distributions for a specific task, resulting representations might lose important information for a given task. In the context of Sentiment Analysis, authors in [116] incorporated prior knowledge at both word and document level with the aim to investigate how contextual sentiment was influenced by each word. On the same direction, other researchers [117] employed sentiment of text for the generation of words embeddings. In particular, they joined context semantics and sentiment characteristics, so that in the word embeddings model neighboring words have both a similar meaning and sentiment. The rationale behind that depends on the fact that many words with a similar context are usually mapped on similar vector representations even if they have an opposite sentiment polarity (e.g., *bad* and *good*). Similarly, authors in [118] augmented sentiment information into semantic word representations and extended Continuous Skip-gram model (Skip-gram), coming up with two sentiment Word Embedding models. The learned sentiment word embeddings were able to correctly represent sentiment and semantics. Furthermore, authors in [119] presented a model that uses a mix of unsupervised and supervised techniques to learn word vector representations, including semantic term-document features. The model showed performances higher than several Machine Learning methods adopted for sentiment detection. Focusing on Twitter sentiment classification, authors in [120] trained sentiment-sensitive word embeddings through the adoption of three neural networks designed to detect the sentiment polarity of texts. Their method encoded sentiment information in the continuous representation of words. Experiments on a benchmark Twitter classification dataset in SemEval 2013 showed that competitors were outperformed. Last but not least, authors in [121] described a procedure with word embeddings for the estimation of levels of negativity in a sample of 56,000 plenary speeches from the Austrian parliament. They found out that the different levels of negativity shown by speakers in different roles from government or opposition parties agree with expected patterns indicated by common sense hypotheses. Their results showed that the word embeddings approach offers a lot of potential for Sentiment Analysis and automated text analysis in the social sciences.

Several challenges have been created to solve Sentiment Analysis polarity detection task and several resulting winning systems employed word embeddings within their core. For example, the Semantic Sentiment Analysis challenge [122, 123, 124,

125], held within the ESWC conference, reached its fifth edition[11][126]. The 2018 edition included a polarity detection task where participants were asked to train their systems by using a combination of word embeddings already generated by the organizers. The aim was to both validate the quality of their systems (precision/recall analysis) and detect which combination of embeddings worked better. SemEval is a workshop on semantic evaluation that takes place each year and includes a set of tasks in NLP and Semantic Web (e.g., Sentiment Analysis polarity detection). One participant of the SemEval-2018 edition targeted the task of irony detection in Twitter [127]. It employed a simple neural network architecture of Multilayer Perceptron with various types of input features, including lexical, syntactic, semantic and polarity features. The proposed system used 300-dimensional pre-trained word embeddings from GloVe [23] to compute a tweet embedding as the average of the embeddings of words in the tweet. By applying latent semantic indexing and extracting tweet representation through the Brown clustering algorithm, it achieved high performance in both subtasks of binary and multi-class irony detection in tweets. It ranked third using the accuracy metric and fifth using the F-measure. Kaggle[12] is the world's largest community of data scientists and offers Machine Learning competitions, a public data platform, and a cloud-based workbench for data science. It hosts several challenges, and some were related to Sentiment Analysis. For instance, the *Sentiment Analysis on Movie Reviews* challenge[13] asked participants to label the movie reviews collected in the Rotten Tomatoes dataset [128] on a scale of five values: negative, somewhat negative, neutral, somewhat positive, positive. One recent challenge, namely *Bag of Words Meets Bags of Popcorn*[14], looked for Deep Learning models combined with word embeddings for polarity detection of movie reviews collected by authors in [119].

4.4.2 Problem Statement

The Sentiment Analysis problem we targeted within the E-Learning domain aims at automatically predicting the positiveness students' opinions about courses. The problem is defined as follows: given a set of students' reviews $R = \{r_1, \ldots, r_n\}$, each one labelled with a score $S = \{s_1, \ldots, s_n\}$ we want to compute an assignment $\gamma : R' \rightarrow S$, where R' is a set of reviews that have not been labelled. The objective is to build γ by exploring Deep Learning methodologies in order to support the development of E-Learning Analytics tools for students of online platforms. Reviews are first mapped through the use of state-of-the-art word embeddings. Then a Deep Learning model has been trained by feeding the word embeddings representations and measuring the MAE and MSE measures. The metrics we have analyzed measure the effectiveness of the employed word embeddings representations of reviews and

[11] http://www.maurodragoni.com/research/opinionmining/events/challenge-2018/
[12] https://www.kaggle.com/
[13] https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews
[14] https://www.kaggle.com/c/word2vec-nlp-tutorial

how well the Deep Learning model assigns scores to them.

4.4.3 Methodology

This section describes as a guide the designing of sentiment prediction models for educational reviews, and presents practical information on how to implement such systems. The main components of the proposed solution (Figure 4.9) and the benefits of the designing choices for each of these components will be described. This helps readers have a broader view of difficulties and solutions behind sentiment prediction models, and make appropriate decisions during their design.

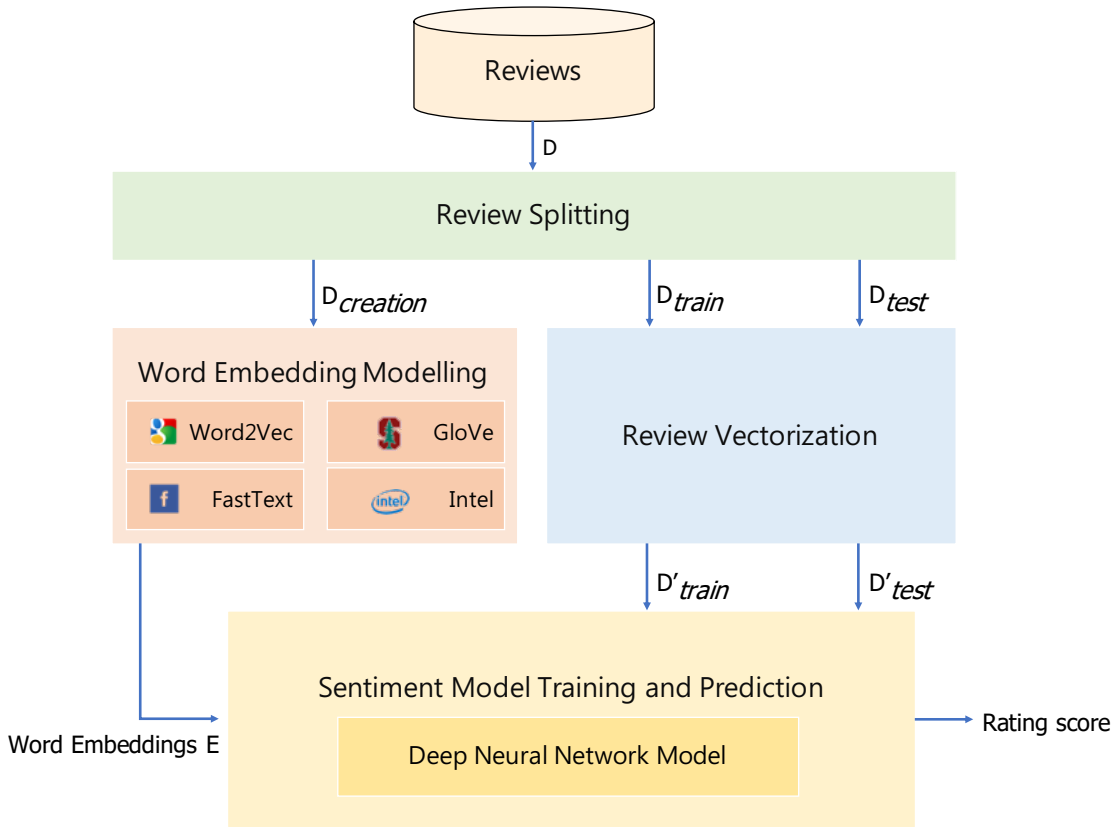


Figure 4.9: The base components of our sentiment prediction model.

Review Splitting

The *review splitting* step serves to define the various input dataset splits while developing the sentiment prediction model. Firstly, we consider the input dataset as a set D of N reviews organized as follows:

$$D = \{(text_1, score_1), ..., (text_N, score_N)\} \tag{4.1}$$

where $text_i$ is a textual review and $score_i$ is an integer rating belonging to the set $C = \{score_1, ..., score_M\}$.

During this step, we thus need to split input data D in three subsets, each for a specific phase of the development:

1. $D_{creation}$: the sample of data used to create word embeddings.
2. D_{train}: the sample of data used to fit the model (i.e., weights and biases).
3. D_{test}: the sample of data used as the gold standard to evaluate the model.

In order to do this, we set up two split ratios and we assign the text-score pairs in D to the different subsets $D_{creation}, D_{train}, D_{test}$ according to them:

1. $s_{creation} \in [0, 1]$: the percentage of reviews for each class $c \in C$ that are randomly chosen from the set D to create word embeddings, yielding $D_{creation}$.
2. $s_{training} \in [0, 1]$: the percentage of reviews for each class $c \in C$ that are randomly chosen from $D \setminus D_{creation}$ to train the model, yielding D_{train}.

The remaining reviews represent D_{test}. The overall procedure ensures that the subsets are disjoint and their union covers the entire dataset D.

Word Embedding Modelling

The state-of-the-art method to model a word with a vector is using word embeddings; it is common to see word embeddings that are 256-dimensional, 512-dimensional, or 1,024-dimensional when dealing with very large vocabularies. There are two ways to generate and leverage word embeddings:

1. Learn word embeddings jointly with the same context we are interested in by starting with random word vectors and, then, learning word vectors along the process, iteratively.
2. Load into the sentiment prediction model word embeddings pre-computed using a different Machine Learning task than the one we are interested in. If the amount of training data in D_{train} is small, this is the common solution.

To the best of our knowledge, no word embeddings database specifically targets the E-Learning context. Therefore, this step goes through the first most general solution of learning word embeddings from scratch, while we also use Word Embedding pre-computed on other contexts for comparison along the chapter.

In order to generate word embeddings from scratch, the subset of pre-processed reviews $D_{creation}$ was employed. We concatenated them into a large corpus and this corpus was fed into a given Word Embedding generation algorithm selected among the following ones: *Word2Vec*, *GloVe*, *FastText*, or *Intel*. Each of them outputs a set of feature vectors E for words in that corpus. The feature values are non-negative real numbers. For each distinct word w in the vocabulary in $D_{creation}$, there exists a corresponding feature vector $e \in E$ which represents the Word Embedding for that word. All the feature vectors share the same size. The size of the resulting word embeddings and of the window where word embeddings generator algorithms look at contextual words can be arbitrarily selected.

Review Vectorization

The *review vectorization* is the process of transforming each review into a numeric sequence. This can be done in multiple ways (e.g., segment text into words and transform each word into a vector, segment text into characters and transform each character into a vector, extract n-grams of words or characters, and transform each n-gram into a vector). The different units into which the text is broken (words, characters, or n-grams) are called *tokens*, and breaking text into such tokens is called *tokenization*. The process consists of applying some tokenization schemes and then associating numeric vectors with the generated tokens. These vectors, packed into sequences, are needed for manipulating text during sentiment model training and inference.

In order to be treated by machines, we need to turn the datasets D_{train} and D_{test} into a set of integer-encoded pre-processed reviews defined as follows:

$$D'_{train} = \{(text'_1, score_1), ..., (text'_K, score_K)\} \forall (text_i, score_i) \in D_{train} \tag{4.2}$$

$$D'_{test} = \{(text'_1, score_1), ..., (text'_J, score_J)\} \forall (text_i, score_i) \in D_{test} \tag{4.3}$$

where each pair $(text'_i, score_i)$ includes an integer encoding of the text comment $text_i$ and the original rating $score_i$ from D_{train} and D_{test}, respectively.

The process for generating D'_{train} and D'_{test} works as follows. Each word has a unique associated integer value chosen from a range going from 0 to $|V| - 1$, where V is the vocabulary of words in D. For each input review $(text_i, score_i)$, we build an integer-encoded vector $text'_i$ from $text_i$, where an integer value at position j in $text'_i$ represents the mapped value for word w for that position in $text_i$. The sets D'_{train} and D'_{test} are thus vectorized.

Sentiment Model Definition

This step is necessary for defining the architecture of the deep neural network which takes pairs of integer-encoded texts and sentiment scores, maps such texts into word embeddings, and tries to predict the sentiment score from them.

The proposed architecture tailored for sentiment score prediction is shown in Fig. 4.10. Given that our training process requires running the network on a rather large corpus, our design choices are mainly driven by the computational efficiency of the network. Hence, differently from [129], which presents an architecture with two Bidirectional LSTM layers, we adopt a single Bidirectional LSTM layer architecture. Moreover, we configure the last layer to return a single continuous value, i.e., the predicted sentiment score. Therefore, our network is composed by an Embedding layer followed by a Bidirectional LSTM layer, a Neural Attention mechanism, and a Dense layer. Each layer works as follows:

1. **Embedding Layer** takes a two-dimensional tensor of shape (N, M) as input, where N represents the number of integer-encoded text comment samples, while M the maximum sequence length of such samples. Each entry is a sequence of integers passed by the Input Layer. The output of the Embedding

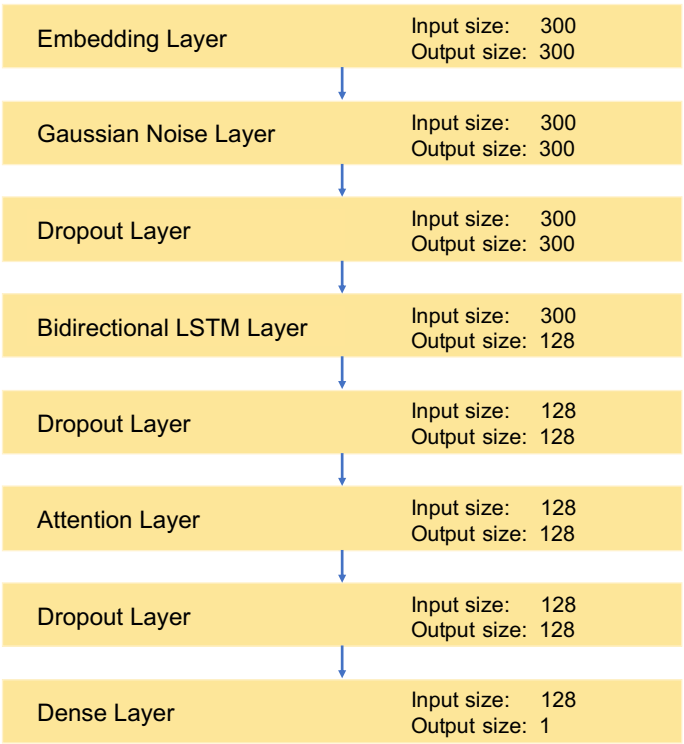


Figure 4.10: The proposed Deep Learning model for sentiment score regression designed to leverage 300-dimensional input text sequences.

layer is a two-dimensional vector with one embedding for each word w in the input sequence of words of each text comment t. Before receiving data, the Embedding Layer loads the pre-trained word embeddings computed during the previous step as weights. Such weights are frozen, so that the pre-trained parts are not updated during training and testing to avoid forgetting what they already know.

2. **Bidirectional LSTM Layer** is an extension of the traditional LSTM that generally improves model performance on sequence classification problems. It trains two LSTM instead of just one: the first is trained on the input sequence as it is and the second on a reversed copy of the input sequence. The forward and backward outputs are then concatenated before being passed on to the next layer, and this is the method often used in studies of bidirectional LSTM. Through this layer, the model is able to analyze a reviews as a whole, binding first and last words coming up with a more precise score. Moreover, exploiting the bidirectional version of a LSTM, the model is able to get patterns that depend on the learners' writing style.

3. **Attention Layer** enables the network referring back to the input sequence, instead of forcing it to encode all the information forward into one fixed-length vector. It takes n arguments $y_1, ..., y_n$ and a context c. It returns a vector z which is supposed to be the summary of the y_i, focusing on information linked to the context c. More specifically, in our model it returns a weighted arithmetic mean of the y_i, and the weights are chosen according to the relevance of each y_i given the context c. This step can improve performance, detecting which words more influence the sentiment assignments.

4. **Dense Layer** is a regular densely-connected layer that implements a function $output = activation(dot(input, kernel)+bias)$ where $activation$ is the element-wise activation function, while $kernel$ and $bias$ are a matrix of weights and a bias vector created by the layer, respectively. The layer uses a linear activation $a(x) = x$ and provides a single output unit representing the sentiment score.

To mitigate the overfitting, the network augments the cost function within layers with l_2-norm regularization terms for the parameters of the network. It also uses Gaussian Noise and Dropout layers to prevent feature co-adaptation.

Sentiment Model Training and Prediction

The fresh instance of the sentiment model takes a set of neural word embeddings E together with a set of pre-processed reviews D'_{train}, as input. With these embeddings and reviews, the component trains the deep neural network. As an objective, the network measures the MSE (Mean Squared Error) of the predicted sentiment score against the gold standard value for each input sequence. Parameters are optimized

using RMSProp (Root Mean Square Propagation) [130] with $learning\_rate = 0.001$. The network was configured for training on batches of size 128 along 20 epochs, shuffling batches between consecutive epochs. The trained deep neural network takes a set of unseen reviews D'_{test} and returns the sentiment score $score'$ predicted for that text comment $text'$, as output.

4.4.4 Results and Discussion

Baselines

We experimented and compared our Deep Learning approach with the following common Machine Learning algorithms:

- SVM.
- RFs. We use Random Forests with 10 trees with depth 20.
- FNN. We used a common Feed-forward Neural Network (FF) with 10 hidden layers, as described in Section 2.3.

We exploited the regression algorithm implementations available within the scikit-learn library. To feed data into these baseline models, we compute the average of word embeddings for each review. More specifically, given a review r with terms $\{t_0, ..., t_{n-1}\}$, we took the associated word embeddings $\{w_0, ..., w_{n-1}\}$ and computed their average w, which is used to represent the review text. In order to evaluate the performance of our model, we measured the MSE (Mean Squared Error) and the MAE (Mean Absolute Error) scores.

Deep Neural Network Model Regressor Performance

Figure 4.11 reports the MAE of regressors used in our experiments. First of all, our results confirm that Neural Networks, both using a single Feed-forward layer and using our model, perform better than common Machine Learning algorithms, showing a lower error. Comparing the Feed-forward baseline with our Deep Neural Network model, there is a little error difference. It is possible to note that the combination *FF + FastText* obtains similar performances of both *DNNR + GloVe* and *DNNR + FastText*. The best performance was obtained by *DNNR + Word2Vec*. Similar considerations also apply when analyzing the MSE. In fact the *DNNR* model gets best performance as well. In contrast with the *MAE*, no baseline obtains performances similar to our model.

Contextual Word Embeddings Performance

This further experiment aims to show how the context-trained word embeddings we generated have advantage over reference generic-trained word embeddings, when

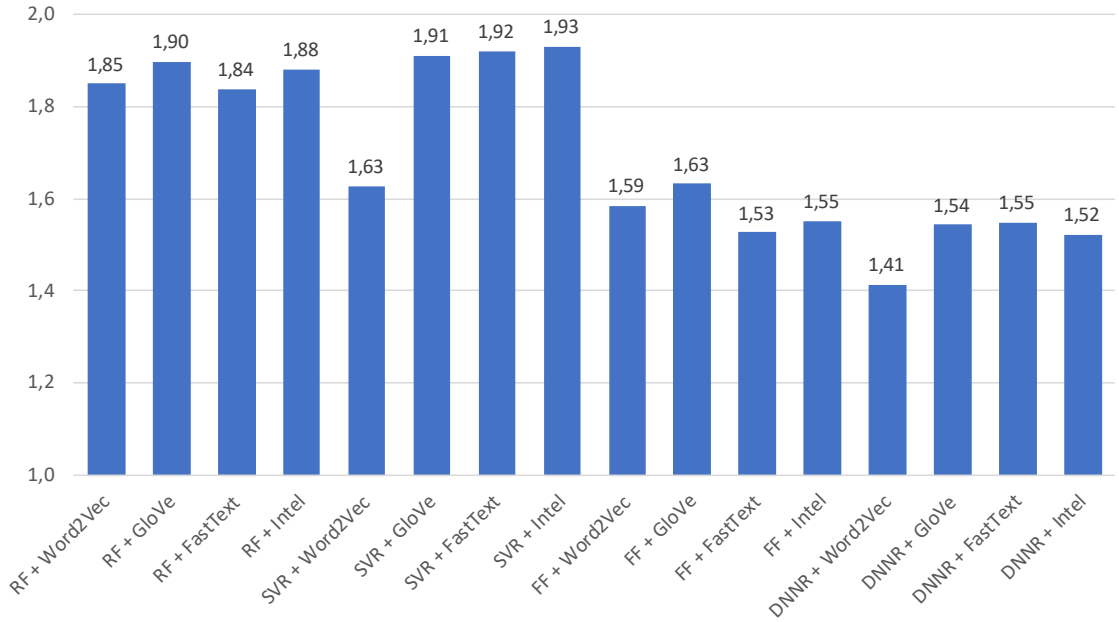


Figure 4.11: Mean Absolute Error (MAE) of Experimented Regressors.

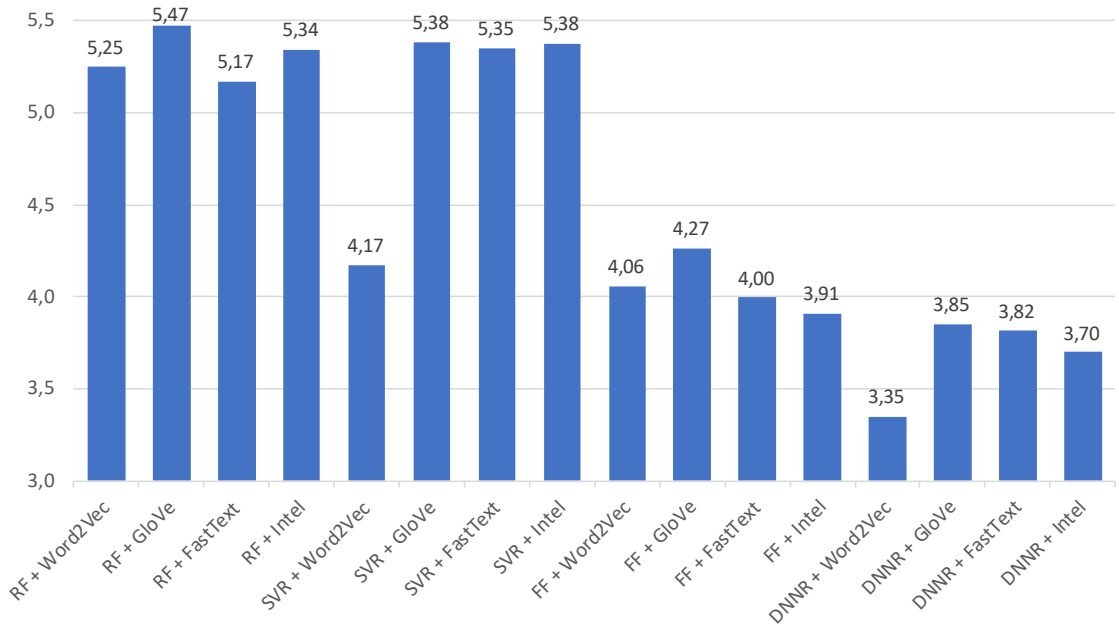


Figure 4.12: Mean Square Error (MSE) of Experimented Regressors.

they are fed into our deep neural network as frozen weights of the Embedding Layer. In order to evaluate the effectiveness of our approach, we performed experiments using embeddings of size 300 trained on COCO's online course reviews. We compared them against the following reference generic-trained word embeddings of size 300 commonly adopted in literature:

- The *Word2Vec*[15] word embeddings trained on a part of the Google News dataset including 100 billion words with a vocabulary of 3 million words.
- The *GloVe*[16] word embeddings trained on a Wikipedia dataset including one billion words with a vocabulary of 400 thousand words.
- The *FastText*[17] word embeddings trained on a Wikipedia dataset including four billion words with a vocabulary of 1 million thousand words.

Context-trained *Intel* word embeddings are compared with generic *Word2Vec* word embeddings because i) there are not public generic *Intel* word embeddings, and ii) the *Intel* algorithm is an evolution of *Word2Vec* algorithm.

[15] `https://code.google.com/archive/p/word2vec/`
[16] `https://nlp.stanford.edu/projects/glove/`
[17] `https://s3-us-west-1.amazonaws.com/fasttext-vectors/wiki.en.vec`

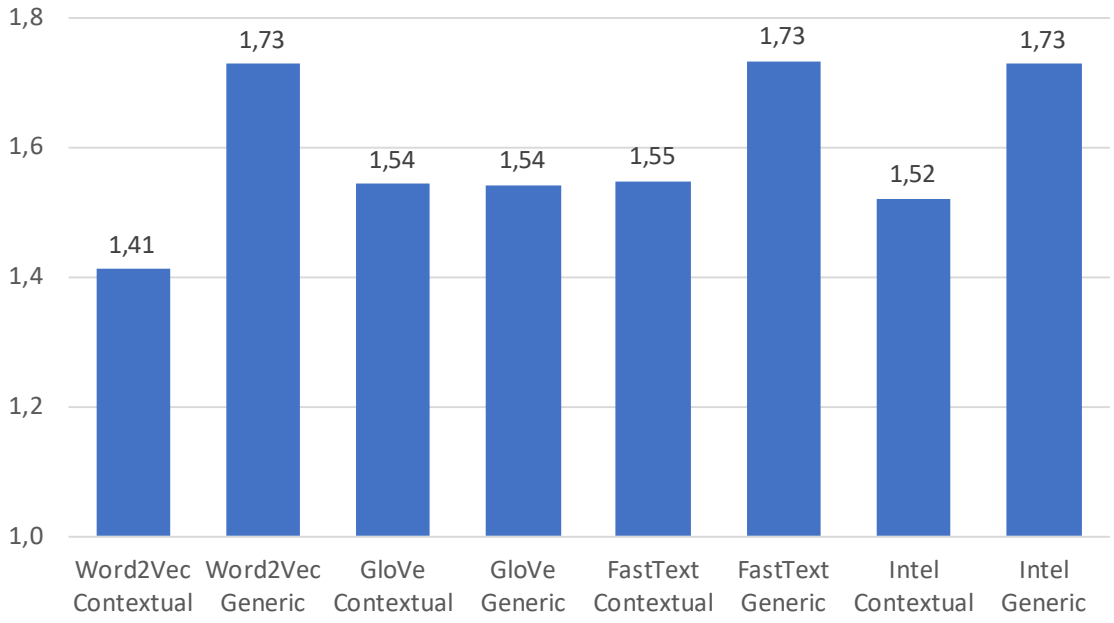


Figure 4.13: Comparison between Contextual word embeddings and Generic word embeddings considering the Mean Absolute Error (MAE).

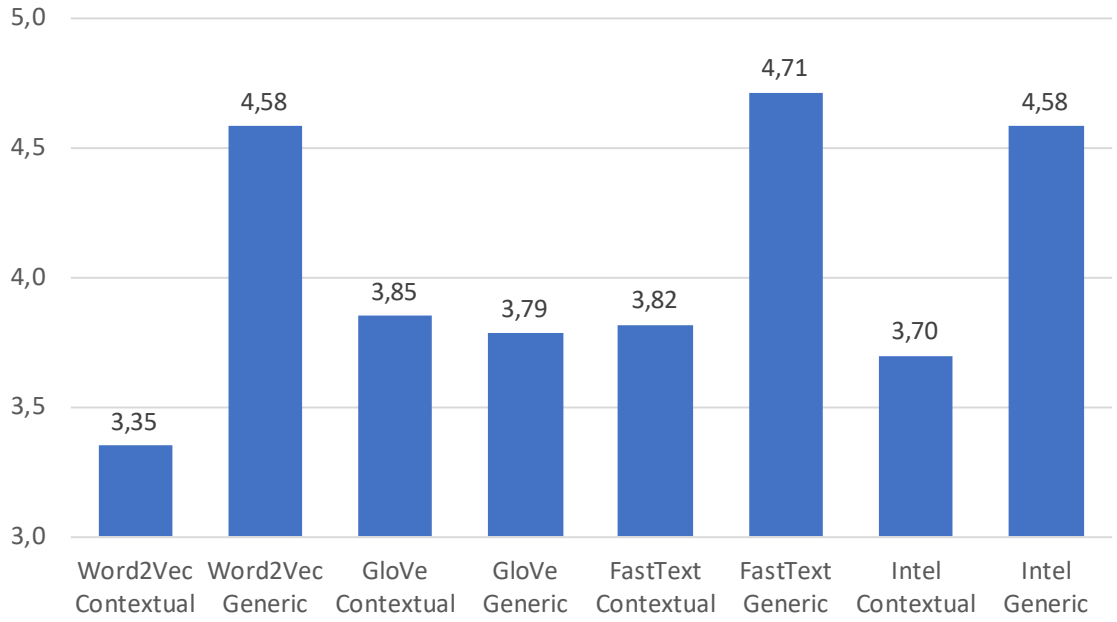


Figure 4.14: Comparison between contextual word embeddings and generic word embeddings considering the Mean Square Error (MSE).

Figure 4.13 shows that there is not a relevant difference between context-trained word and generic-trained embeddings when the MAE is used for the comparison. Nevertheless, it is worth underling how the type of embeddings enables to obtain better results in the E-Learning domain. Context-trained *Word2Vec* embeddings show the lowest values of MAE compared to other embeddings types. In contrast, when the MSE is considered, context-trained embeddings perform better, as shown in Figure 4.14. In this case, context-trained embeddings have low values of MSE in almost all cases except for the *GloVe* word embeddings. The best performance was obtained by context-trained *Word2Vec* embeddings, proving that i) *Word2Vec* is the best algorithm to learn word representations from our dataset, and ii) context-trained word embeddings are able to capture specific patterns of the E-Learning domain. This makes possible to adapt our Deep Learning model on the E-Learning domain and improve the results in sentiment score prediction.

Chapter 5

Scholarly Domain

5.1 Open Issues

Knowledge graphs are large networks of entities and relationships, usually expressed as RDF triples, relevant to a specific domain or an organization [131]. Many state-of-the-art projects such as DBPedia [132], Google Knowledge Graph, BabelNet, and YAGO build knowledge graphs by harvesting entities and relations from textual resources, such as Wikipedia pages. The creation of such knowledge graphs is a complex process that typically requires to extract and integrate various information from structured and unstructured sources.

Scientific knowledge graphs focus on the scholarly domain and typically contain metadata describing research publications such as authors, venues, organizations, research topics, and citations. Good examples are Open Academic Graph[1], Scholarlydata.org [133], and OpenCitations [134]. These resources provide substantial benefits to researchers, companies, and policy makers by powering several data-driven services for navigating, analyzing, and making sense of research dynamics. One of their main limitations is that the content of scientific papers is represented by unstructured texts (title, abstract, sometimes the full text) or in the best scenario as list of terms from a vocabulary (e.g., MeSh, ACM, PhySH, CSO). Therefore, a significant open challenge in this field regards the generation of Scientific knowledge graphs that contain also an explicit representation of the knowledge presented in scientific publications [135], and potentially describe entities such as approaches, claims, applications, data, and so on. The resulting knowledge graph would be able to support a new generation of content-aware services for exploring the research environment at a much more granular level.

Most of the relevant information for populating such a knowledge graph might be derived from the text of research publications. In the last year, we saw the emergence of several excellent NLP for entity linking and relationship extraction [136, 135, 137, 138, 139]. However, integrating the outputs of these tools in a coherent knowledge

[1] `https://www.openacademic.ai/oag/`

graph is still an open challenge.

In this chapter, we present an approach that uses a set of NLP and Machine Learning methods for extracting entities and relationships from research publications, and then integrates them in a knowledge graph. Within our work, we refer to an entity as a linguistic expression that refers to an object (e.g., topics, tools names, a well-know algorithm, etc.). We define a relation between two entities when they are syntactically or semantically connected. As an example, if a tool T adopts an algorithm A, we may build the relationship (T, *adopt*, A).

The main contributions of the research presented in this chapter are: i) an approach that combines different tools for extracting entities and relations from research publications ii) an approach for integrating these entities and relationships, iii) a qualitative analysis of a generated scientific knowledge graph in the field of Semantic Web, and iv) an evaluation in terms of precision, recall, and f-measure of generated triples that compose the scientific knowledge graph.

5.2 Background

Many information extraction approaches for harvesting entities and relationships from textual resources can be found in literature.

First, entities in textual resources have been detected by applying Part-Of-Speech (PoS) tags. An example is constituted by [140], where authors provided a graph based approach for Word Sense Disambiguation (WSD) and Entity Linking (EL) named Babelfly. Later, other approaches started to exploit various resources (e.g., context information and existing knowledge graphs) for developing ensemble methodologies [138]. Following this idea, we exploited an ensemble of tools to mine scientific publications and get information out of them. Then, we designed and implemented a software pipeline for the purpose of creating a scientific knowledge graph that organizes entities and their relations.

An important task for modeling data regards the discovering of relations that connect two entities. This task has been already addressed in literature in order to connect data coming from pieces of text. For example, authors in [137] developed a machine reader called FRED[2] which exploits Boxer [141] and links elements to various ontologies in order to represent the content of a text in a RDF representation. Among its features FRED extracts relations between frames, events, concepts and entities. One more project that enables the extraction of RDF triples from text is [142], where a framework called PIKES has been designed to exploit the frame analysis to detect entities and their relations. Although these works model text knowledge into a machine readable format, they have been thought for extracting general information which makes difficult the use of the generated graphs for specific domain applications. Moreover, they only consider a single text at a time and do not consider the source of text they parse. In contrast with them, our approach aims

[2] `http://wit.istc.cnr.it/stlab-tools/fred/`

at parsing specific type of textual data and, moreover, at combining information from various textual resources. We decided to rely on open domain information extraction tool results refined by considering the domain of data. In addition, we combined entities and relations coming from different scientific papers instead of considering single results of our texts at a time. With our approach the resulting knowledge graph represents the overall knowledge presented in the input scientific publications.

Focusing more on the Scholarly Domain, extraction of relations from scientific papers has recently raised interest within the SemEval 2018 Task 7 *Semantic Relation Extraction and Classification in Scientific Papers* challenge [143], where participants had to face the problem of detecting and classifying domain-specific semantic relations. An attempt to build knowledge graphs from scholarly data was performed by [139], as an evolution of their work at SemEval 2018 Task 7. Authors proposed both a Deep Learning approach to extract entities and relations, and then they built a knowledge graph on a dataset of $110,000$ papers. Our work finds inspiration from it, but we used different strategies to address open issues for combining entities and relations. For example, for solving ambiguity issues that regard the various representations entities can have, authors of [139] considered clusters of co-referenced entities to come up with a representative entity in the cluster. On the contrary, we adopted textual and statistics similarity to solve it. Furthermore, they only used a set of predefined relations that might be too generic for the purpose of yielding insights from the research landscape.

5.3 Problem Statement

Given a large collection of research papers, we want to generate a large-scale knowledge base, that will include all relevant entities in a certain domain and their relationships. More in detail, given a set of scientific documents $D = \{d_1, \ldots, d_n\}$, we build a model $\gamma : D \rightarrow T$, where T is a set of triples (also referred as relationships) (s, p, o) where s and o belong to a set of entities E and p belongs to a set of relations labels L. Each triple needs also to be associated with the set of papers it was extracted from, allowing to assess how supported is the relevant claim in the original collection of documents. The resulting knowledge graph can be employed for different problems of new research fields (e.g., detection of research communities, their dynamics and trends, forecasting of research dynamics using sentiment analysis, measuring fairness of open access datasets, etc.), and, in general, as a support resource for scientists in conducting scientific research.

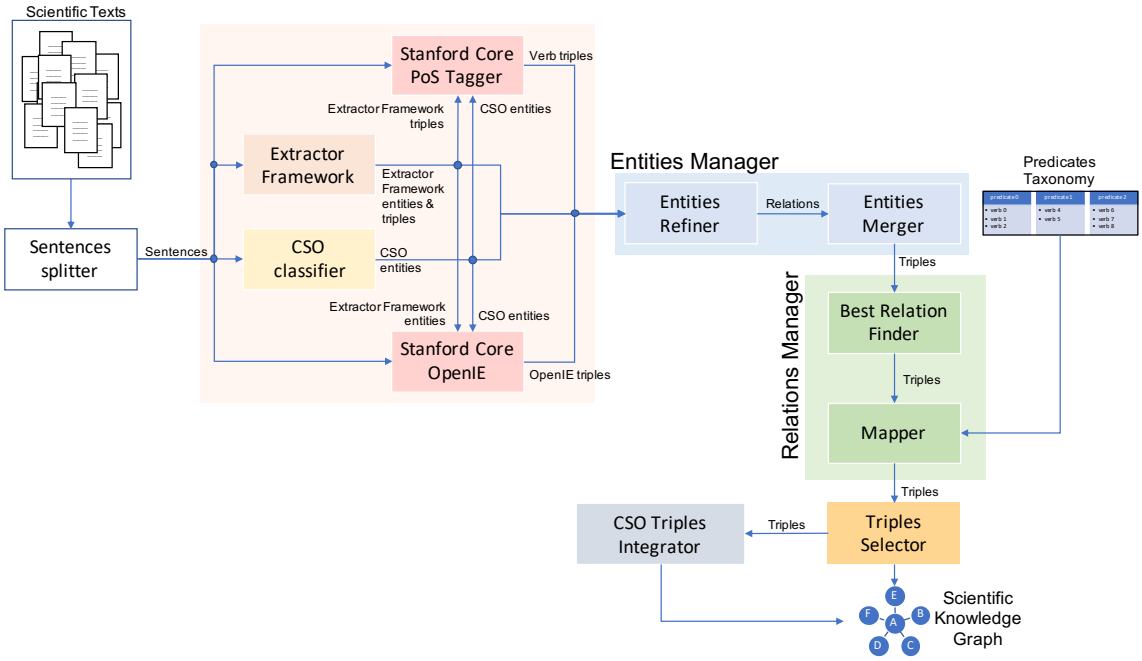


Figure 5.1: Workflow of our approach for building a scientific knowledge graph from scientific textual resources.

5.4 Methodology

In this section, we describe the approach that we applied to produce a scientific knowledge graph of research entities. The workflow of our pipeline is shown in Figure 5.1. In short, our framework includes the following steps:

1. **Extraction of entities and triples**, which exploits an ensemble of several NLP and Machine Learning tools to extract triples from text.

2. **Entity refining**, in which the resulting entities are merged and cleaned up.

3. **Triple refining**, in which the triples extracted by the different tools are merged together and the relations are mapped to a common vocabulary.

4. **Triple selection**, in which we select the set of "trusted" triples that will be included in the output by first creating a smaller knowledge graph composed by triples associated with a good number of papers and then enriching this set with other semantically consistent triples. In the following subsection we will describe the architecture in more details and discuss the specific NLP and Machine Learning tools that we used in the implementation of our prototype.

5.4.1 Extraction of Entities and Relations

For extracting entities and relations, we exploited the following methods:

- *The extractor framework* [139] designed by Luan Yi et al. that we modified and embedded within our pipeline. It is based on Deep Learning models and provides modules for detecting entities and relations from scientific literature. It detects six types of entities (*Task, Method, Metric, Material, Other-Scientific-Term, and Generic*) and seven types of relations among a list of predefined choices (*Compare, Part-of, Conjunction, Evaluate-for, Feature-of, Used-for, Hyponym-Of*). For the purpose of this work, we discarded all the triples with relation *Conjunction*, since they were too generic. In particular, the extractor framework uses feed-forward neural networks over span representations of the input texts to compute two scores v_1 and v_2. The score v_1 is computed on single spans and measures how likely a span may be associated to an entity type. The second score v_2 is a pairwise score on a pair of span representations and measures how likely spans are involved in a relation. Therefore, for a given pair of span representations, let's say (t_1, t_2), the scores $v_1^{t_1}$, $v_1^{t_2}$, and $v_2^{(t_1,t_2)}$ are computed. If both $v_1^{t_1}$ and $v_1^{t_2}$ meet a threshold t_{entity}, and $v_2^{(t_1,t_2)}$ meets a threshold $t_{relation}$ than the span representations t_1 and t_2 are labelled as entities, and their pair as relationship (t_1, t_2). The type of entity where the value v_1 is the highest is associated to the entity itself. Similarly, a pair (t_1, t_2) is associated to the type of relation r where the pair has the highest value of v_2, yielding the triple (t_1, r, t_2). We refer to this framework as Extractor Framework.

- *The CSO Classifier [144]*[3], a tool for automatically classifying research papers according to the Computer Science Ontology (CSO)[4] [101], which is a comprehensive automatically generated ontology of research areas in the field of Computer Science. It identifies topics by means of two different components, the syntactic module and the semantic module. Then it combines their outputs and enhances the resulting set by including all relevant super-topics. The *syntactic module* removes English stop words and collects unigrams, bigrams, and trigrams. Then, for each n-gram, it computes the Levenshtein similarity with the labels of the topics in CSO. Finally, it returns all research topics whose labels have a similarity score equal to or higher than a threshold to one of the n-grams. The *semantic module* uses part-of-speech tagging to identify candidate terms composed of a proper combination of nouns and adjectives and maps them to the ontology topics by using a Word2Vec model. Then, the module computes a relevance score for each topic in the ontology by considering the number of times the topic was identified within the retrieved words.

[3] `https://github.com/angelosalatino/cso-classifier`
[4] `http://cso.kmi.open.ac.uk`

- *OpenIE [145]* provided by the Stanford Core NLP suite. It detects general entities and relations among them. Relations are detected by analyzing clauses (i.e., groups of words that contain at least a subject and a verb) which are built by exploring the parse tree of the input text. In the first stage, the methodology produces clauses from long sentences which stand on their own syntactically and semantically. For doing so, it uses a multinomial logistic regression classifier to recursively explore the dependency tree of sentences from governor to dependant nodes. Then, it applies logical inferences to capture natural logic within clauses by using semantics dictating contexts. Doing so, OpenIE is able to replace lexical items with something more generic or more specific. Once a set of short entailed clauses is produced, it segments them into its output triples. In our approach we keep triples where entities match those found by the Extractor Framework and the CSO Classifier, so that we caught only those triples that refer to entities of the target domain.

- *The Stanford Core NLP PoS tagger*[5] which extracts predicates between the entities identified by the Extractor Framework and the CSO Classifier. More specifically, for each sentence s_i it detects all verbs $V = \{v_0, \ldots, v_k\}$ between each pair of entities (e_m, e_n) and generates triples in the form $<e_m, v, e_n>$where $v \in V$.

We processed each sentence from all the abstracts and used the tools and methods above to assign to each sentence s_i a list of entities E_i and a list of triples R_i.

First, we run the extractor framework to extract both entities E_i and triples R_i. Secondly, we used the CSO Classifier to extract all Computer Science topics, further expanding E_i. Thirdly, we processed each sentence s_i with OpenIE, and retrieved all the triples composed by subject, verb, and object in which both subject and object matched the entities resulting from the previous steps. Finally, for each sentence s_i we took all the verbs within two entities through the PoS Tagger, yielding R_i thoroughly expanded.

5.4.2 Entities Manager

Different entities in E_i may have a wrong representation due to errors in the extraction process. For example, two different entities may actually refer to the same concept with alternative forms, or may represent too generic concepts that do not carry meaningful information. In this section, we briefly describe which steps we have performed by the Entities Refiner and Entities Mapper modules in order to address these issues.

[5] `https://nlp.stanford.edu/software/tagger.shtml`

Entities Refiner Module

Many of the entities resulting from previous steps can be noisy, ambiguous, and too generic. The goal of this module is to preprocess the entities, merging alternative labels, discarding ambiguous and generic entities, and splitting the ones that include compound expressions.

Cleaning up entities. First, we removed punctuation (e.g., dots and apostrophes) and stop-words (e.g., pronouns) from entities. We also removed some words that might be mixed up (e.g., *it* might be the pronoun *it* or the acronym of *information technology*) by using a blacklist.

Splitting entities. Some entities actually contained multiple compound expressions, e.g., *Machine Learning and Data Mining*. Therefore, we split entities that contain the conjunction *and*. Referring to our example, we obtained the two entities *Machine Learning* and *Data Mining*.

Handling Acronyms. Acronyms are usually defined, appearing the first time near their extended form (e.g., *Web Ontology Language (OWL)*) and then by themselves in the rest of the abstract (e.g., *CSO*). In order to map acronyms with their extended form in a specific abstract we use a regular expression. We then substituted every acronym (e.g., *OWL*) in the abstract with their extended form (e.g., *Web Ontology Language*). Since acronyms can be ambiguous, we perform this operation only on entities from the same abstract.

Detection of Generic Entities. Entities might be too generic for the purpose to describe the knowledge of a domain (e.g., *content*, *time*, *study*, *article*, *input*, and so on.) We discard these kinds of entities by applying a frequency-based filter which compares the frequency of the entities in three set of documents:

- the set of publications of the target domain.
- a set of the same size covering *Computer Science* domain, but not the target domain.
- a set of the same size containing papers from various domains, but not about the target domain nor the *Computer Science*.

For each entity e, we computed the number of times it appeared in the above datasets, so that we had three different counts $c_e^{'}, c_e^{''}, c_e^{'''}$. We normalized the counts by dividing them with the number of words of the set where they were computed. Then we computed the ratios $r_e^{'} = \frac{c_e^{'}}{c_e^{''}}$ and $r_e^{''} = \frac{c_e^{'}}{c_e^{'''}}$. If the ratio $r_e^{'}$ met a threshold $t_e^{'} = 2$, and the ratio $r_e^{''}$ met a threshold $t_e^{''} = 10$ the entity e was included in the graph. Thresholds were empirically defined by manually evaluating which entities were saved/discarded. In addition, we automatically preserved all entities within a whitelist composed by CSO topics and all the paper keywords in the initial dataset.

Entities Merger Module

We merge entities with the same meaning by using both a lemmatizer and the CSO ontology. Singular and plural forms are combined by using the Lemmatizer available in the SpaCy[6] library. Then we exploited the alternative labels described by CSO to merge entities that refer to the same research topic (e.g., "ontology alignement" and "ontology matching"). More specifically, given an entity $e \in E$ that is known by CSO, let $A = \{e_0, ..., e_{k-1}, e\}$ be the set of alternatives of e in CSO. The module first found the longest label $e_{longest} \in A$, then each instance $e \in E \cap A$ is substituted by $e_{longest}$.

5.4.3 Relations Manager

This step aims at (i) finding the best relation predicate for each pair of entities e_i, e_j where a relation exists (each element in R), and (ii) mapping all the relations within a table we have defined.

Best Relation Finder Module

Here, the set of triples R presents three different types of triples: those extracted by the Extractor Framework, let us say R_{EF}, those coming from OpenIE, let us say R_{OIE}, and those detected with the PoS tagger, called R_{PoS}. We performed the following operations on these sets:

- On the set of triples in R_{EF} we acted as follows. Given a pair of entities (e_p, e_q) in R_{EF}, we merged into a list L_r all relations' labels r_i such that $(e_p, r_i, e_q) \in R_{EF}$. Then we chose the most frequent relation $r_{most\_frequent} \in L_r$, and built a single triple $(e_p, r_{most\_frequent}, e_q)$. Triples so built formed the set T_{EF}. Clearly, the size of the set T_{EF} is lower than the size of the set R_{EF}.

- On the set R_{OIE} we performed a deeper merging operation. Similarly to the work performed on R_{EF}, given a pair of entities (e_p, e_q) in R_{OIE}, we first merged into a list L_r all relations' labels r_i such that $(e_p, r_i, e_q) \in R_{OIE}$. In R_{OIE} all triples have a verb as relation predicate. Hence, we assigned each r_i to its word embedding w_i from the word embeddings built on the MAG dataset, yielding the list L_w. With the word embeddings in L_w an averaged word embedding w_{avg} was built. Then, the relation r_i with the word embedding w_i nearest to w_{avg} according to the cosine similarity was chosen as final relation for the pair (e_p, e_q), yielding the triple (e_p, w_i, e_q). The same procedure was also applied on R_{PoS}. The execution of this procedure on R_{OIE} and R_{PoS} yielded the sets T_{OIE} and T_{PoS}, respectively.

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

- Finally, for the sets T_{EF}, T_{OIE}, and T_{PoS}, we saved for each triple (e_p, r_i, e_q) the number of papers where the pair of entities (e_p, e_q) appeared. We refer to this number as the *support* of triples.

Mapper Module

In order to reduce the number of relations available in the output, we designed a mapping M to assign to similar predicate a single label (for example, the list of verbs *uses*, *adopts*, *employs*, and so on, were mapped to the label *uses*). The mapping was created by first clustering all the relations in T_{OIE} and T_{PoS} according to their word embeddings. We used a hierarchical clustering algorithm provided by the SciKit-learn library[7] which uses $1-$ cosine similarity as distance. The resulting dendrogram was cut by an empirically determined threshold of Silhoutte-width $= 0.65$.

Finally, we manually revised resulting clusters and defined a mapping M, which map each predicate to a specific label. Relations of all triples from the union of T_{EF}, T_{OIE}, and T_{PoS} were mapped by using M.

5.4.4 Triples Selection

In this section the method we employed to choose only certain triples is presented. We also define what we mean with the words *valid* and *consistent* associated to our triples in order to build the scientific knowledge graph.

Valid Triples

For the purpose of including meaningful triples within our knowledge graph, we first define a smaller knowledge graph composed of "valid" triples. These can be defined in different way according to the performance of the tools in the first step and the number of papers supporting a certain triples.

In the current prototype we define as valid the following triples:

- All triples extracted directly from a specific sentence. Therefore, we consider *valid* all the triples obtained by the Extractor Framework (T_{EF}) and the OpenIE tool (T_{OIE}).

- All triples associated with at least 10 papers (indicating a fair consensus). Therefore, we consider *valid* the triples that were detected by the PoS tagger associated with at least 10 papers. We refer to this set as T'_{PoS} such that $T'_{PoS} \subseteq T_{PoS}$.

The union of T_{TF}, T_{OIE}, and T'_{PoS} composed the set of all valid triples T_{valid}.

[7] `https://scikit-learn.org`

Consistent Triples

The set of triples not in T_{valid}, that we label $T_{invalid}$, may still include several good triples that were not associated to sizable number of papers. We thus use the triples in T_{valid} as examples to learn which triples are consistent with the valid ones and could still be included in the final outcome. Specifically, we trained a classifier $\gamma : P \rightarrow L$ where P is a set of pair of entities in $T_{invalid}$ and L is the set of relations used in M (e.g., "uses", "provides", "improves"), with the aim of comparing the actual relation with the one returned by the classifier. The intuition is that a triple consistent with T_{valid} would have its relation correctly guessed by the classifier. In order to do so, we performed the following steps:

1. We generated word embeddings of size 300 by processing with the Word2Vec algorithm [22, 16] all the input abstracts. For multi-word entities we replaced white spaces with underscore characters within our abstracts texts (e.g., the entity *semantic web* becomes *semantic_web*).

2. We trained a Multi-Perceptron Classifier (MLP) to return the relation between a couple of entities. We used the concatenation of the embeddings of subject and object entities as input and the relation as output.

3. The validation step was performed by applying the classifier on all the triples (e_p, r, e_q) in $T_{invalid}$ and comparing the actual relation r with the relation returned by the classifier r'. If $r = r'$ then the triple (e_p, r, e_q) was considered valid and added to T_{valid}. Otherwise we computed the cosine similarity *cos_sim* and the *Wu-Palmer*[8] similarity between the embedding of r and r'. If the average between *cos_sim* and *wup_sim* was higher than a threshold ($t = 0.5$ in the prototype) then the triple (e_p, r, e_q) was considered valid and added to T_{valid}.

5.4.5 Knowledge Graph Enhancement

In order to augment the extent of the information we also added to the resulting knowledge graph all the additional triples that could be inferred by exploiting the hierarchical relations in CSO. More precisely, given a triple (e_1, r, e_2), if in CSO the entity e_3 is *superTopicOf* of the entity e_1 and there is no triple involving e_3 and e_2, we also infer the triple (e_3, r, e_2). For instance, given the triple ("NLP systems", "use", "DBpedia") if "Semantic Web Technologies" is *superTopicOf* "DBpedia", we can infer the triple ("NLP systems", "use", "Semantic Web Technologies"). This last step was performed by the CSO Triples Integrator module in the pipeline. Finally, the triples are converted to RDF and returned.

[8] `http://www.nltk.org/howto/wordnet.html`

Table 5.1: Examples of triples that our pipeline detects.

Subject Entity	Relation	Object Entity
semantic web technologies	supports	contextual information
semantic relationship	defines	ontologies
structural index	uses	structural graph information
thesaurus	hyponymy-of/is	knowledge organization system
web page classification	uses	text of web page
question answering systems	uses	semantic relation interpreter

5.5 Results and Discussion

This section details the scientific knowledge graph we have produced and shows how we have performed its validation.

5.5.1 The Semantic Knowledge Graph

Here we report the result of our framework, focusing on the field of *Semantic Web* domain. We used an input dataset composed by 26,827 abstracts of scientific publications about this domain that was retrieved by selecting publications from the Microsoft Academic Graph dataset[9]. It is a knowledge graph related to the scholarly domain that describes more than 200 million scientific publications through metadata such as title, abstract texts, authors, venue, field of study and so on. For our purpose we considered only abstracts that were classified under Semantic Web by the CSO Classifier [144]. This same dataset has also been used for exploring the relationship between Academia and Industry by Angioni et al. [146].

A few examples of retrieved triples can be seen in Table 5.1.

The resulting knowledge graph includes 109,105 triples (87,030 from the Extractor Framework (T_{EF}), 8,060 from OpenIE (T_{OIE}), and 14,015 from the PoS tagger method and classifier (T'_{PoS} + *Cons. Triples*).

However, the raw number of triples extracted by each method can be misleading. Indeed, not all triples are equal. Some are supported by a large number of papers, suggesting a large consensus of the scientific community and more in general a claim that can easily be trusted, while some others appear in one or very few papers.

Figures 5.2 reports the distribution of the support of the triples produced by T_{EF}, T_{OIE} and T'_{PoS} + *Cons. Triples*.

While T_{EF} produces the most sizable part of those triples, most of them has a very low support. In fact, 80,030 of them are supported by a single paper and

[9] `https://www.microsoft.com/en-us/research/project/microsoft-academic-graph`

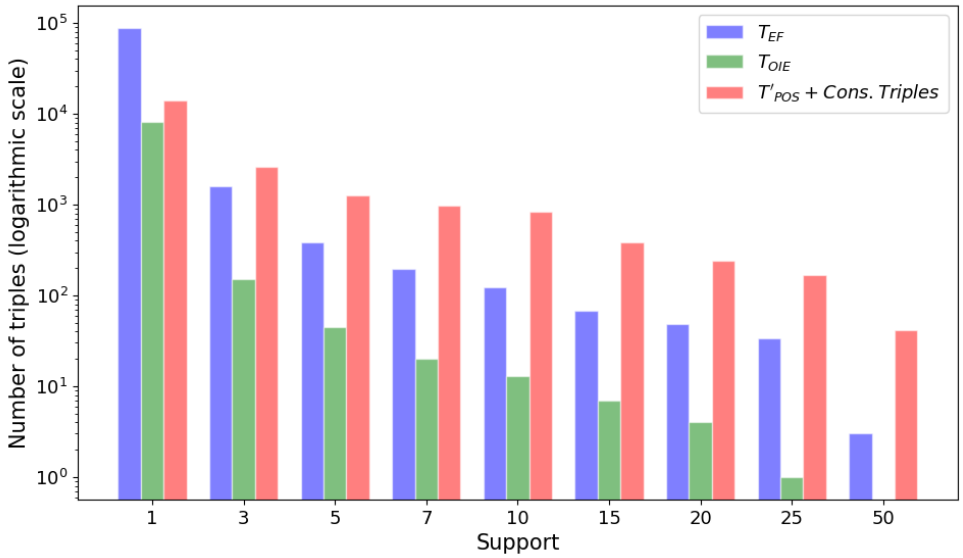


Figure 5.2: Comparison of the distribution of the support of the three methods.

862 by only three papers. They may thus contain claims that did not reach yet a consensus in the community. For all the other support values, the set T'_{PoS} + *Cons. Triples* has a higher number of triples than T_{EF} and T_{OIE} and, hence, it is possible to assume that T'_{PoS} triples may be more in accordance within the community of *Semantic Web*. For instance, if we take in consideration only the triples whose support is equal or greater than 5, only 393 triples are provided by the set T_{EF}, 45 by T_{OIE} and, 1,268 by T'_{PoS} + *Cons. Triples*. It is also worth to note that when the support is very high (e.g., equal or greater than 50) there are not triples provided by the set T_{OIE}, and few triples provided by T_{EF}. This still stresses the fact that those triples might not express valuable knowledge or have consensus within the *Semantic Web* community.

5.5.2 Gold Standard Creation

We first used several different approaches to generate triples from the 26,827 abstracts described in the previous section. Specifically, we applied on this dataset: 1) T_{EF} (i.e.,the Extractor Framework), 2) T_{OIE} (i.e., OpenIE), 3) T'_{PoS} (considering only the triples with support ≥ 10), and T'_{PoS} + *Cons. Triples*.

The resulting set of 109,105 triples would be unfeasible to manually annotate, since it is very large and too sparse in term of expertise. We thus focused only on 818 triples which contain (as subject or object) at least one of the 24 sub-topics of Semantic Web and at least another topic in the CSO ontology. This set contains 401 triples from T_{EF}, 102 from T_{OIE}, 60 triples from T'_{PoS} and 110 relevant *Cons.*

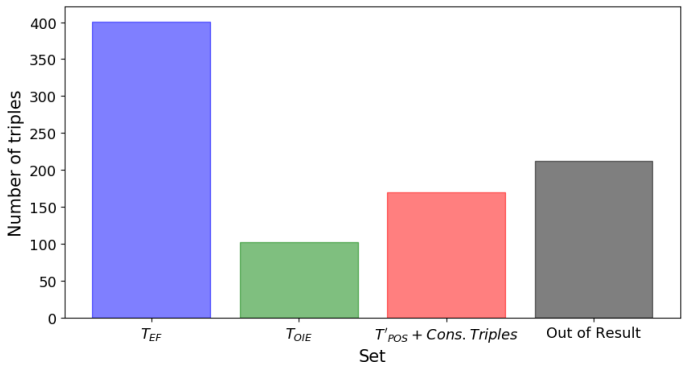


Figure 5.3: Distribution of triples within the gold standard.

Triples. In order to measure the recall, we also added 212 triples that were discarded by the framework pipeline. The reader notices that the total number of triples (818) is slightly less than the sum of various sets (401+102+60+110+212) because some triples have been derived by more than one tool. The triples distribution of the gold standard can be observed in Figure 5.3.

We recruited five researchers in the field of Semantic Web and asked them to annotate each triple either as *true* or *false*. The averaged agreement between experts was 0.747 ± 0.036, which indicates a high inter-rater agreement. We then created the gold standard using the majority rule approach. Specifically, if a triple was considered relevant by at least three annotators, it was labeled as true, otherwise as false. The purpose of this gold standard is twofold. First, it allows us to evaluate the proposed pipeline to extract triples from scholarly data and, second, it provides a resource which will facilitate further evaluations.

5.5.3 Precision, Recall, F-measure Analysis

For evaluating our methodology, we performed a precision, recall, F-measure analysis considering various combinations of relations sources.

We tested eight alternative approaches:

- The Extractor Framework from Luan Yi et al. [139] (**EF**) described in section 5.4.1.
- OpenIE, from Angeli et al. [145] (**OpenIE**) described in section 5.4.1.
- The Stanford Core NLP PoS tagger described in section 5.4.1, after merging the relevant triples as described in section 4.3.1. (T'_{PoS}). We considered only the triples with support ≥ 10.
- The previous approach enriched by consistent triples as described in section 4.4.2 (T'_{PoS} + **Cons. Triples**).

- The combination of EF and OpenIE (**EF + OpenIE**).
- The combination of EF and T'_{PoS} + Cons. Triples (**EF + T'_{PoS} + Cons. Triples**).
- The combination of OpenIE and T'_{PoS} + Cons. Triples (**OpenIE + T'_{PoS} + Cons. Triples**).
- The final framework that integrates all the previous methods (**OpenIE + EF + T'_{PoS} + Cons. Triples**).

Table 5.2 reports precision, recall, and F-measure of all the methods.

EF obtains an high level of precision (84.3%), but a recall of only 54.4%. OpenIE and T'_{PoS} shows a slightly lower level of precision and an even lower recall. T'_{PoS} + Cons. Triples obtains the best precision of all the methods (84.7%), highlighting the advantages of using a classifier for selecting consistent triples. Overall, all these basic methods produce triples with good precision, but suffers in term of recall.

Combining them together generally raises the recall without paying too much in term of precision. EF + OpenIE yields a F-measure of 72.8% with a recall of 65.1% and EF + T'_{PoS} + Cons. Triples a F-measure of 77.1% with a recall of 71.6%. The final version of our framework which combines all the previous methods, obtains the best recall (80.2%) and F-measure (81.2%) and yields also a fairly good precision (78.7%). This seems to confirm the hypothesis that an hybrid framework combining supervised and unsupervised methods would produce the most comprehensive set of triples and the best performance overall.

Table 5.2: Precision, Recall, and F-measure of each method adopted to extract triples. To note that the last row identified the triples extracted using the full pipeline.

Triples identified by	**Precision**	**Recall**	**F-measure**
EF	0.8429	0.5443	0.6615
OpenIE	0.7843	0.1288	0.2213
T'_{PoS}	0.8000	0.0773	0.1410
T'_{PoS} + Cons. Triples	**0.8471**	0.2319	0.3641
EF + OpenIE	0.8279	0.6506	0.7286
EF + T'_{PoS} + Cons. Triples	0.8349	0.7166	0.7712
OpenIE + T'_{PoS} + Cons. Triples	0.8145	0.3253	0.4649
OpenIE + EF + T'_{PoS} + Cons. Triples	0.7871	**0.8019**	**0.8117**

5.5.4 Examples and considerations about the Scientific Knowledge Graph

In this section, we show some sample of the triples extracted for the Semantic Web Knowledge Graph and discuss benefits and limitations of our output.

Table 5.3 shows a selection of the triples about the research topic *ontology alignment*, ranked by *support*. It is easy to see that many of these triples define the fundamental characteristics of *ontology alignment*. The topic is contextualized (via "skos:broader" relations) within the areas of *semantic web technologies* and *information integration*. *Ontology alignment* is defined as an entity that uses *ontologies*, selects *semantic correspondences*, and supports *semantic interoperability*.

Several other triples add further details, such as that *ontology alignment* finds *semantically related entities*, adopts *semantic similarity measures*, and limits the need for *human intervention*. Naturally, the representation also suffers from some issues that we plan to address in future work. For instance, the triples <*ontology alignment, selects, mapping*> and <*ontology alignment, supports, semantic relations*> appear too ambiguous. This may be either a limitation of our vocabulary of relations or an issue in the methodology used for merging together the triples from the PoS tagger. Similarly in <*ontology alignment, produces, semantic web application*> the predicate does not appear to be correct, maybe "support" would be a better choice in this case. We thus plan to work further on our approach for merging triples and select the best predicate between two entities.

The triple <*ontology alignment, produces, semantic web application* > shows another typical issue. In the knowledge graph we have both *distributed and heterogeneous ontology* and *heterogeneous ontology* but no link between the two. In the future we need to be able to detect that *distributed and heterogeneous ontology* is actually a sub-concept of *heterogeneous ontology*.

Figure 5.4 shows a graphical representation of the research topic *ontology evaluation*. It is interesting to notice how this representation is also fairly interpretable by human users. Some examples about the information that can be derived includes:

- *ontology evaluation* uses *natural language techniques*. It suggests that there might be tools or methodologies that exploit textual resources written in natural language that have been involved in ontologies evaluation.
- *ontology evaluation* is hyponym of the entity *ontology construction* indicating that a specialized task within *ontology construction* involves the evaluation of the produced ontologies.
- *ontology evaluation* is hyponym of *instance data evaluation* which shows in which more general task the *ontology evaluation* falls.

Finally, it is also interesting to consider an entity that is not so much represented in the input dataset. Figure 5.5 shows the subgraph of the entity *supervised machine*

Table 5.3: Examples of triples from the Semantic Web Knowledge Graph.

Subject Entity	**Relation**	**Object Entity**	**Support**
ontology alignment	uses	ontologies	194
ontology alignment	skos:broader	semantic web technologies	65
ontology alignment	selects	semantic correspondence	45
ontology alignment	supports	semantic interoperability	34
ontology alignment	maintains	heterogeneous ontology	25
ontology alignment	selects	mapping	21
ontology alignment	selects	semantically related entity	19
ontology alignment	supports	semantic relation	17
ontology alignment	produces	semantic web application	14
ontology alignment	combines	concept similarity	13
ontology alignment	supports	semantic heterogeneity problem	13
ontology alignment	limits	human intervention	12
ontology alignment	executes	semantic similarity measures	12
ontology alignment	produces	ontology mapping method	11
ontology alignment	provides	distributed and heterogeneous ontology	10
ontology alignment	skos:broader	information integration	10
ontology alignment	uses	mapping system	10
ontology alignment	provides	matching technique	10

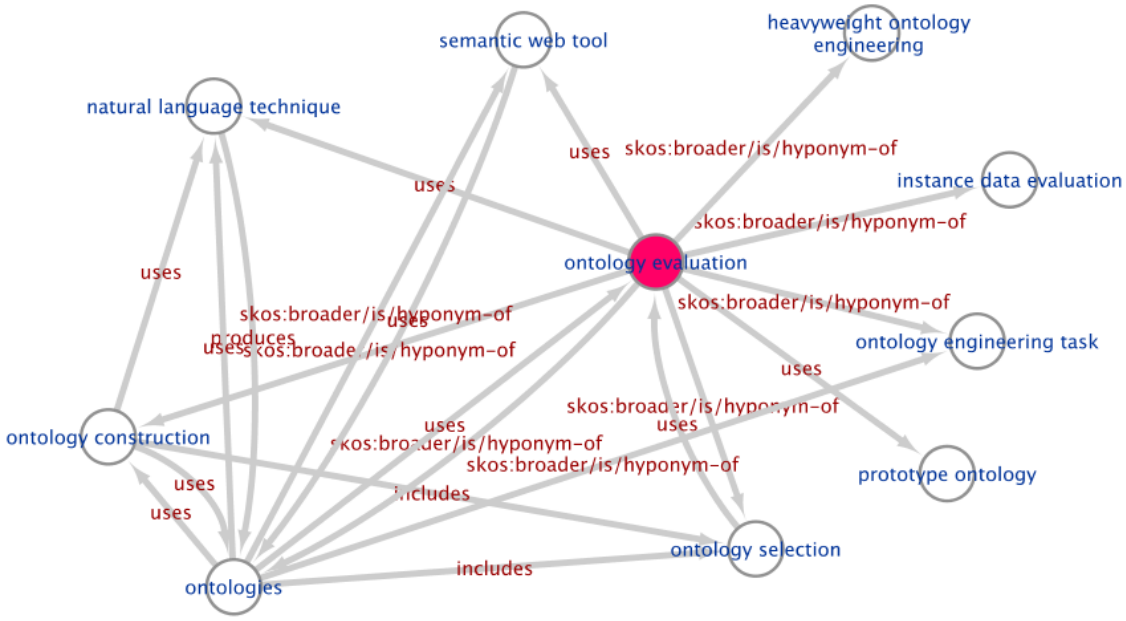


Figure 5.4: The subgraph of the entity "ontology evaluation" with related relationships in our Scientific Knowledge Graph within the Semantic Web domain.

learning. This representation is useful to highlight which topics and kind of resources are employed by *supervised machine learning* within the *Semantic Web* domain. As an example, it is easy to see that this entity uses both *structured data model* and *rich semantics*, and how these two entities are related as well. In the example, only two types of relations appear (i.e., *uses* and *includes*). They seem too generic, in fact, it is not clear how *supervised machine learning* adopts the other linked entities. This can indicate that our taxonomy of predicates may be too general and we may have to adopt a more fine grained representation in future work.

Overall, the knowledge graph seems to contain triples of good quality that well represent the main characteristics of research entities within the context of the input dataset. We thus believe that this version may already be used for enhancing the representation of research items and supporting users in understanding and navigating research outcomes.

Although the good results we obtained within this research work, there are open issues that need to be still addressed and solved. One of these regards entities that are not present in CSO (although they might be related to the computer science domain, they might be written with different terms thus not matching those present in CSO) and, therefore, they are not topics, leaving entities with different forms within our triples. This can raise ambiguity issues within the knowledge graph, not allowing detailed analysis of the research landscape of the produced knowledge graph. Further improvements by considering semantic similarity between entities need to be developed for the sake of intelligibility of the graph information.

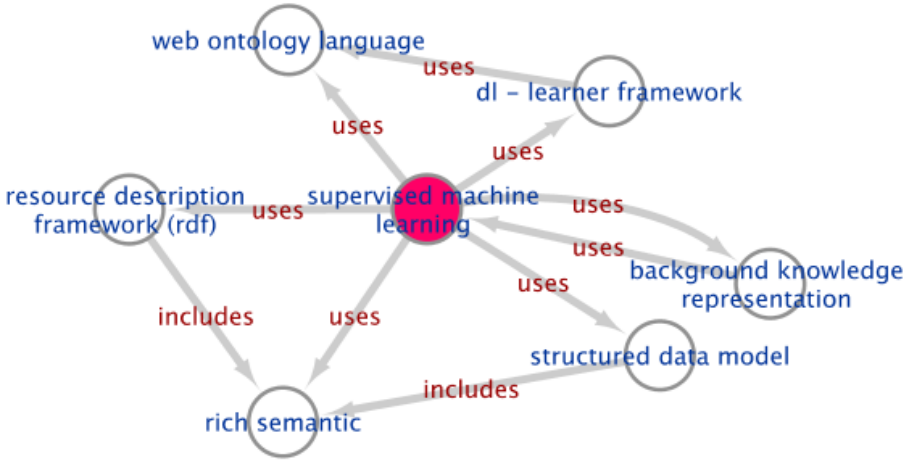


Figure 5.5: The subgraph of the entity "supervised Machine Learning" with related relationships in the produced Scientific Knowledge Graph within the Semantic Web domain.

One more direction we are interested to explore regards the possibility to have more relations between a pair of entities. Our current scientific knowledge graph allows only one relation from each embedded tool, thus it may have limitations in the exploration of the research landscape. Having more than one relation can suggest different applications or uses of entities, increasing the probability of finding unconsidered issues and solutions within research areas.

Specifically on the pipeline, we would like to add new features by including trending developments such as various kind of embeddings (e.g., embeddings built on graphs) to better capture the knowledge of scientific literature and allow further analysis on data. Moreover, we would like to make the pipeline more efficient since during various tests we noticed that there are bottlenecks that could make difficult, if not impractical, the parsing of big datasets. First, the Extractor Framework requires a lot of hard disk space. This entails that data must be sampled to be processed. Second, the current pipeline only adopts the Stanford Core NLP server with just one thread, asking for a long time to mine textual resources sentence-by-sentence, thus limiting the adoption of the whole pipeline on big datasets. However, this is not a big issue since it would be possible to run the Stanford Core NLP server in multi-thread mode, so speeding up the extraction process. Finally, we intend to test several kinds of word and graph embeddings on this task.

Chapter 6

Patterns on Graphs

6.1 Open Issues

Imagine describing a road map with words alone. The task would be difficult and unclear to most people. Networks provide a far better representation of any data representing interrelationships. However, because the size of networks (for example, in social science) can extend to thousands, millions, or even billions of nodes, networks themselves need to be abstracted for the sake of intelligibility and insight.

A frequent way to reduce the size of the problem is to discover similar components and give them a common name. Linguists do this when they categorize parts of speech (noun, verb, adverb etc). Biologists do this when they group animals into species and families. In networks, scientists do this by finding connected labeled sub-components that are isomorphic in label and topology. Formally, this entails finding common subgraphs or motifs that occur with a certain frequency.

Much research has proposed algorithms that aim at finding frequent motifs [147, 148, 149, 150, 151]. The motivation is usually to gain insights about metabolic and protein-protein interactions, ecological food-webs, social networks, collaboration networks, information networks of interlinked documents and products [152, 153, 154, 155, 156, 157, 158, 159, 160].

Most of this work does not distinguish between motifs that overlap and motifs that do not. However, this distinction can be critical for understandability. For example, households are a convenient abstraction in social graphs because they are disjoint whereas friendship motifs do not tend to be. For networks whose motifs are not naturally disjoint, identifying disjoint motifs may help to understand network structure (e.g., cliques in friendship networks). A recent research work that explored disjoint motifs is [161], which introduced algorithms to find edge-disjoint motifs in unlabeled networks. Similarly, the work presented in this chapter focuses on node-disjoint motifs which share neither nodes nor edges in labeled networks.

Once disjoint motifs of a certain size k have been identified, each such motif can be collapsed into a *supernode*, which is a single node that inherits all the connections

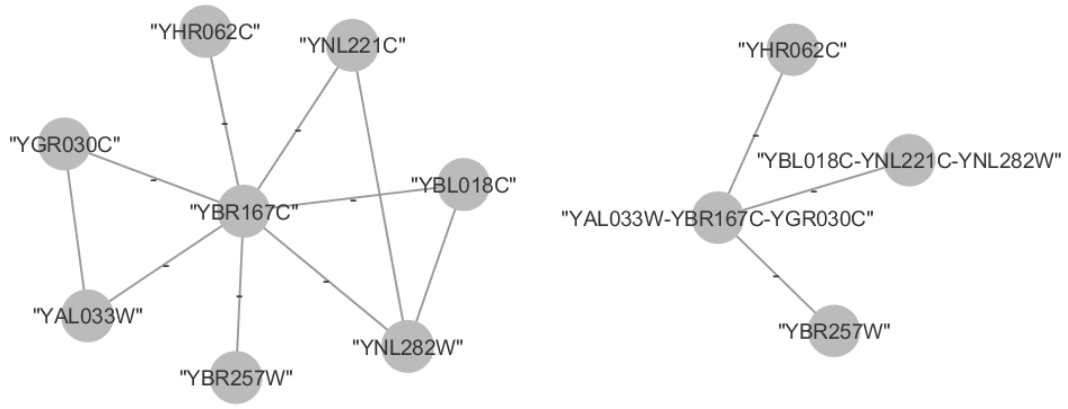


Figure 6.1: Example of motifs collapsed into supernodes in a Protein-Protein Interaction network. (a) The original nodes of the network. (b) The new nodes of the network after two motifs of size three have been collapsed.

and properties of nodes the compose the motif. This procedure can be recursively performed in order to find motifs on graphs consisting of a combination of nodes and super-nodes. Figure 6.1 shows an example where motifs have been collapsed into supernodes.

This kind of approaches results particularly useful for biological networks which usually describe how molecules interact to perform biological functions. These networks are represented by graphs where the nodes and the edges represent the interacting molecules and the interactions between them respectively. Detecting motifs has a great potential to help the study of the biological functions such as gene regulatory or protein-protein interactions served by these networks. Motifs are ever more considered the building blocks of networks. Hence, the scientific community is continuously studying and providing novel tools combining knowledge from biology and computer science background for helping biologists in analyzing biological networks.

Thus, in this chapter a tool called *Supernoder* is presented. It finds disjoint motifs on a base graph G1, reducing G1 to a new graph G2, and then recursively repeats the procedure to find G3, G4, and so on. SuperNoder attempts to find the most possible disjoint frequent motifs of a given size in a target network in each stage of the process. We present several techniques to achieve this goal.

Orthogonally, the SuperNoder tool can take input nodes at different layers in a label hierarchy. For example in phylogeny, there is a hierarchy of species, genus, family, kingdom. Relationships that may be obscure at a low level may be clearer at a high level (e.g., felines eat rodents).

This chapter makes three contributions:

- Efficient algorithms to find disjoint supernodes in labeled networks, including

networks already containing supernodes, yielding a recursive algorithm.

- A tool incorporating these algorithms that is free to the community.
- Example applications to show the usefulness of the approach.

6.2 Background

In this section, we first provide theoretical definitions that will help the reader to understand the work addressed within this chapter. Then, we briefly present the literature about this topic. Labeled networks or graphs are formally characterized by a triple $G = (N, E, L)$ where N denotes a set of nodes, E denotes a set of edges (pairs) $e = (n_i, n_j) \in N$, and L is a mapping from N to some set of labels. Edges represent an application-dependent relationship. For instance, an edge may connect two nodes representing people if the people are friends.

We say that a graph is *undirected* if every edge from n to n' implies the existence of an edge from n' to n. Otherwise the graph is said to be *directed*. A *subgraph* is a *connected* component $G_S = (N_S, E_S)$ such that $N_S \subseteq N$ and $E_S \subseteq E$ if there exists a path from each $n_i \in N_S$ to each $n_j \in N_S$. A $k - subgraph$ is a subgraph with k nodes.

Two subgraphs S_1, S_2 are *isomorphic* if (i) there exists a bijective function $f : N_{S1} \rightarrow N_{S2}$ such that for each pair $(n_i, n_j) \in E_{S1} \leftrightarrow (f(n_i), f(n_j)) \in E_{S2}$ and (ii) for all k, the label of n_k i.e., $L(n_k)$ is the same as $L(f(n_k))$. To count the number of occurrences of a given subgraph, three different measures can be used [162]. The first measure, named F1, is the count of each subgraph regardless of whether it overlaps with others. The second one, named F2, avoids overlaps of subgraphs if they share at least an edge (or equivalently a connected pair of nodes). The last one, named F3, requires that two subgraphs share no nodes. Therefore, F3 is the most strict criterion of disjointness (and is the one used in this paper). We define the *frequency* of a subgraph S_1 in G as the number of occurrences of S_1 in G. We call subgraphs $k - motifs$ if they have k nodes occur over a threshold t using the F1 measure.

In literature, frequent (based on the possibly overlapping F1 measure) motifs have been shown to give insights in regulatory [163], food-web [164, 165, 166], and social science [167, 168] networks. Reduction methods aim at minimizing the loss of information while maximizing the understandability, often establishing which components are less interesting for the behavior of networks. Recent studies have focused on finding high-order clusterings [169, 170]. However, most of this research has focused on modeling graphs without considering node labels, despite the fact that many networks have them. Moreover, they usually consider overlapping motifs, therefore, a single node can belong to several patterns, making further analysis (and understandability) difficult.

An early compression graph method was proposed by [171] where the authors show how finding substructures and merging them in vertexes for compressing data. Our approach builds on theirs, but their approach does not find all substructures that occur nor does it attempt to find the most highly repetitive subgraphs which are the best candidates for capturing subgraph regularities.

Our work also draws inspiration from [161] where the authors propose two methods to find disjoint motifs under the F2 frequency measure (where two graphs are disjoint if they do not share a common edge). First, they propose a method to find motifs based on a small set of patterns, and then give methods to find non-overlapping motifs solving the Maximum Independent Set (MIS) problem. They invented their own method for finding frequent motifs and did not choose to compare their method with state-of-the-art motif-finding techniques [172, 173, 174, 175, 176, 177]. By contrast, we have chosen to base our approach on the motif-finding algorithm of [172] because of its simple implementation and promising results [178]. As in [161], the second phase of our algorithm uses an *overlap graph*, and we have explored some heuristics to deal with larger *overlap graphs* beyond what they used. While we do contribute algorithms for finding *disjoint* motifs given a collection of already found motifs, we do not advance the state-of-the-art in finding the motifs themselves. Instead, our work builds on top of an existing overlapping motif finding algorithm which has been compared and studied many times in literature [178].

6.3 Problem Statement

The problem we have targeted in this work was the development of a tools which takes a graph G as an input, detects a set of disjoint motifs M, and yields a new graph G' where disjoint motifs in $M' \subseteq M$ are replaced by single nodes, called *supernodes*, that inherit all properties of nodes of motifs. Our contribution is built on top of the state-of-the-art motifs finder method.

More precisely, once the set of motifs $M = \{m_0, ..., m_n\}$ of size k has been found, our approach finds a subset $M' \subseteq M$ of motifs such that the set of nodes $N_{mi} = \{n_{m_i,0}, n_{m_i,k-1}\}$ that composes a motif $m_i \in M'$ is shared with no other motif $m_j \in M'$. To do this, our methodology adopts an *overlap graph*, let's say $J = (N^J, E^J)$, where each motif $m_i \in M'$ is assigned to a node $n_i^J \in N^J$, and each edge $e_j^J \in E^J$ connects two nodes n_p^J and n_q^J if the motifs m_p and m_q share at least one node. The idea behind the method is that if two nodes are not connected in the *overlap graph* then the underlying motifs are disjoint. Subsequently, the presented approach applies an heuristic (chosen between the five proposed ones) for approximating a Maximum Independent Set yielding the set of nodes $N'^J \subseteq N^J$ where nodes have no connections. Finally, motifs which have been assigned to nodes in N'^J are finally replaced by *supernodes* in G, yielding the graph G'.

6.4 Data Description

For developing and testing our methodology we explore biological networks which are widely used to describe and detect biological processes. We used three different networks:

- A food-web subnetwork of Florida bay network[1] [179] with 93 nodes and 960 edges.
- A Protein-Protein Interaction (PPI) network of yeast[2] [180] with 2,361 nodes and 7,182 edges.
- A PPI network of Arabidopsis[3] [181] with 18,167 nodes and 10,928 edges.

Food-web network. The original nodes have labels that represent animals or plants (e.g. *predatory chanodichthys*, *dinoflagellates*, *coral bryaninops*, etc.). We have mapped the network using a taxonomy[4], retrieving for each node *genus*, *family*, *order*, *class*, *phylum*, and *kingdom*. From the original network we have removed species that did not have higher phylogenetic categories.

Protein-Protein Interaction networks. In a Protein-Protein Interaction (PPI) network, each node represents a different protein. For the higher-level categorization of PPI networks, we have employed the ontology annotations available at this link[5]. First, we have retrieved the Gene Ontology (GO) term that belongs to Biological Processes (BPs) and that has the lowest (i.e., most empirically based) evidence code for each protein. Second, we have traversed the ontology *go-basic*[6] starting from each GO term in our network to the GO term which represents all Biological Processes. Since each GO term can have more than one parent, we have chosen the GO term with the lowest (i.e., most conclusive) evidence code going up in the hierarchy. More precisely, given a label of a node l, we retrieve a GO term g with the lowest evidence code. Let $\{g_1, g_2, ..., g_n\}$ be the parents of g, then we choose the g_i with $1 \leq i \leq n$ with the lowest evidence code, building a hierarchy l, g, g_i. Then, we repeat the same operation as long as the GO term which represents all Biological Processes (BPs) has not been yet reached. In doing so, we have built a taxonomy that can enable the analysis of protein functions.

[1] `https://snap.stanford.edu/data/Florida-bay.html`
[2] `http://vlado.fmf.uni-lj.si/pub/networks/data/bio/yeast/yeast.htm`
[3] `http://interactome.dfci.harvard.edu/A_thaliana/index.php?page=download`
[4] `ftp://ftp.ncbi.nlm.nih.gov/pub/taxonomy/`
[5] `http://www.geneontology.org/page/download-annotations`
[6] `http://www.geneontology.org/page/download-ontology`

6.5 Methodology

6.5.1 Overview of the Proposed Methodology

The methodology has been implemented into the SuperNoder tool whose pipeline consists of the following steps:

1. Solicit a size s from the user corresponding to the number of nodes each motif should have.
2. Solicit a threshold t from the user corresponding to the number of times that a motif should be present to be considered.
3. Search for all possible motifs on the input network meeting threshold t, using the F1 measure (i.e., allowing overlaps). Call that set M.
4. Search for the maximum number of non-overlapping motifs from M.
5. Collapse non-overlapping motifs into supernodes.
6. Repeat steps 2 through 5 until satisfied.

6.5.2 Input network and Motifs Finding

SuperNoder requires two series of data as an input:

- A list of node rows, where each row represents a node by means of a unique *ID* and a *label* separated by a blank space.
- A list of edge rows, where each row consists of two node *IDs* separated by a blank space.

SuperNoder uses the *Randomized Enumeration* algorithm [172] for the purpose of motif finding. The result of the algorithm is a set of all possible undirected motifs in the network, allowing overlaps.

6.5.3 Motifs Count and Thresholding

To count motifs, we implemented a function to compute isomorphisms between subgraphs similar to the one of Cordella and colleagues [182]. First, the algorithm takes the labels of subgraph nodes and counts how many nodes have the same label. Second, for each label it computes the sum of in-degrees and the sum of out-degrees (i.e., for each node label, it computes $l_{n,i,o}$, where n is the number of nodes with label l, i is the sum of in-degree of nodes with label l, and o is the sum of out-degree of nodes with label l). Finally, it sorts these labels using the lexicographic order and computes their hash. If the number of subgraphs having hash value h is greater than

the user-given threshold t, then all such subgraphs are checked to see how many are in fact isomorphic. If, after the check, the number is greater than t, then those subgraphs pass the initial filter to be a motif and thus belong to the "frequent motif set". Thus the frequent motif set may contain different topologies, e.g., at least t stars of size s, at least t paths of length s, and so on.

6.5.4 Finding Disjoint Motifs

Our methods to find disjoint motifs, given the potentially overlapping frequent motif set, uses the concept of an *overlap graph*. An *overlap graph* is a pair (M, E) where M is the set of motifs and there is an edge between motif m_p and motif m_q if they share at least one node in the original graph. (In the case of recursive reduction, the original graph at reduction i is the one produced from the graph at reduction i-1, containing both normal nodes and supernodes.)

We briefly present an overview of our heuristics for finding disjoint motifs here.
H1 (Greedy Elimination). This simple but effective heuristic finds disjoint motifs by using a *Maximal Independent Set* technique. Given the frequent motif set M and a user-given parameter n, randomly shuffle the potentially overlapping motif instances from the frequent motif set M. For each motif instance m, if the motif instance overlaps no other motif instances of M, then output it. Otherwise remove it and all its edges from the overlap graph. Because this approach is naively greedy, SuperNoder tries n (parameter given by the user) different random shufflings to try to obtain the greatest number of disjoint motifs.
H2 (Ramsey) Heuristic-2 exploits both sampling and the Ramsey method whose functions can be seen in [183]. Given the list of motif instances M and a number k, the heuristic (i) takes disjoint subsets of size k from M and constructs the induced subgraph of the overlap network from each subset. (ii) On each subgraph, it performs the Ramsey algorithm obtaining a $MIS_{subgraph}$. (iii) Then, it merges all $MIS_{subgraph}s$ into a reduced list of motif instances which takes the role of M. The algorithm repeats steps (i) through (iii) until there are no more overlaps and outputs the resulting set of motifs.
H3 (Ranked Elimination). Heuristic-3 assigns to each (possibly overlapping) motif instance m a degree equal to the sum of degrees of the nodes in m ignoring the edges between nodes in m (i.e., the sum of the degrees of the nodes in m pertaining to edges that connect to nodes outside m). The algorithm then orders the motif instances in ascending order of degree so calculated, forming a list called $MotifDegree$. For each node n in the original graph, find the first motif instance in $MotifDegree$ and discard all other motifs in $MotifDegree$ containing n. This process yields a new list called $PotentialSuperNodes$. Then traverse this $PotentialSuperNodes$ list, preserving motif instances having no overlaps and deleting motif instances that have higher degrees when there are overlaps.
H4 (Repeated Ranked Elimination). This approach is an improvement over H3, because H3 misses some motif instances when one or more overlapping mo-

tif instances are removed and the nodes of the removed motif instances then have no chance to be included in any other motif instances. Given as input the list of motif instances M found using the *Randomized Enumeration* method seen above, build the $MotifDegree$ list as in Heuristic-3. For each node n, the motif instance $m \in MotifDegree$ with the lowest degree that contains n is copied to a list of potential supernodes, called $PotentialSuperNodes$. All the motif instances in $PotentialSuperNodes$ with no overlaps are considered valid. Then, for each pair $\{m', m''\}$ of overlapping motif instances in $PotentialSupernodes$, discard the motif instance with the higher degree. Continue until there are no more motif instances. Now consider all the nodes N_{orphan} that are not in any disjoint motif instance found so far and consider motif instances based on the F1 measure that apply to nodes of N_{orphan}. Repeat the above procedure to generate more disjoint motif instances. Repeat until there are no more nodes in N_{orphan}.

H5 (Sampled Ranked Elimination). This heuristic unifies sampling with the overlap graph approach. After the sampling is done as for the Ramsey algorithm, the heuristic constructs an *overlap graph* on the surviving motif instances. The heuristic considers the motif instances in ascending order by degree in the *overlap graph*. If a motif instance has no edges, then put it in the result. If a motif instance $m1$ has an edge with another motif instance $m2$, then remove the motif instance with the largest degree.

Table 6.1: Summary of the characteristics of the heuristics. The symbol V indicates that the heuristic exploits that characteristic, - if not. H1 = Greedy Elimination. H2 = Ramsey. H3 = Ranked Elimination. H4 = Ranked Replacement. H5 = Sampled Ranked Elimination.

Heuristic **ID**	**Overlap Graph**	**Ramsey**	**Order by degree**	**Random approach**	**Sampling approach**
H1	-	-	-	V	-
H2	V	V	-	-	V
H3	-	-	V	V	-
H4	-	-	V	V	-
H5	V	-	V	V	V

6.5.5 Network Reduction

After the non-overlapping motif instances have been found, each one is collapsed into a supernode, preserving the external connections of the original nodes of motifs. The label of each supernode is the concatenation of labels of its member nodes in alphabetical order. The new network can be saved as an output using the same format as the input network and the whole pipeline can be iterated on it.

6.6 An Use Case on Biological Graphs

In the analysis of biological networks, interactions often occur between proteins of the same class [184]. SuperNoder can find these relations when high level functional classes are considered, highlighting frequent related processes and simplifying their identification.

To show how SuperNoder may help to simplify networks, we focus on the yeast network, and explain how higher levels of the Gene Ontology (GO) terms enable the abstraction of protein functions allowing SuperNoder to reduce the network complexity. The motivation is simple: at a lower level in the hierarchy of GO terms there may be no motifs that occur more than t times for a moderately large t. At higher levels, there might be. In the example, the yeast network has been mapped onto five levels of the GO terms hierarchy. To be considered a motif, a subgraph has to occur at least 50 times, i.e., with threshold $t = 50$.

Figure 6.2 shows a motif of size three in each row that is mapped on the base level (gene labels), the fifth-level (L5) and the third-level (L3) hierarchy labels (i.e. in ascending order of abstraction). More motifs appear at higher levels in the hierarchy (i.e., first on L5 and then on L3 levels). In fact, with L5 labels the triples in row 2 and row 3 are isomorphic. When L3 labels are used, all triples are isomorphic, thus becoming relevant motifs. Those triples are collapsed into supernodes thus forming a new simplified network. Supernodes indicate proteins that belong to the same class helping biologists with the analysis of basic interactions.

As a specific case study, focus on motifs composed of proteins (*YNL306W, YDR175C, YBR251W*) and (*YGR156W, YKR002W, YLR115W*). Analyzing the network on the base labels, there are not supernodes, since they do not show common features in the labeled graph. Already at lower hierarchical levels (i.e., L5), the motifs GO terms are abstracted into functions, viz, *macromolecule biosynthetic process* and *cellular macromolecule metabolic process* respectively. At hierarchical level L3, the proteins in this example have the label GO:0071704 which indicates that their proteins are related to *organic substance metabolic process*. At that level, we find

Original labels	*GO terms L5*	*GO terms L3*
YNL306W, YDR175C, YBR251W	GO:0009059, GO:0009059, GO:0009059	GO:0071704, GO:0071704, GO:0071704
YGR156W, YKR002W, YLR115W	GO:0044260, GO:0044260, GO:0044260	GO:0071704, GO:0071704, GO:0071704
YGL128C, YER013W, YMR213W	GO:0044260, GO:0044260, GO:0044260	GO:0071704, GO:0071704, GO:0071704
YKL190W, YLR433C, YML057W	GO:0019538, GO:0019538, GO:0019538	GO:0071704, GO:0071704, GO:0071704

Figure 6.2: An example of four supernodes built using SuperNoder with motifs of size three on the yeast network. From left to right, labels of original nodes, labels of the fifth level hierarchy, labels of the third level hierarchy. On the third level, many proteins share the same pattern and these patterns are often disjoint.

out that *organic substance metabolic process* (GO:0071704) covers an important role into the yeast network, and that is mainly composed of *macromolecule biosynthetic process* (GO:0009059), *cellular macromolecule metabolic process* (GO:0044260) and *protein metabolic process* (GO:0019538). This shows an example of how our tool can help biologists understand the behavior of proteins (with frequent motifs) belonging to the same class.

The higher the hierarchy levels, the larger the number of relevant motifs that can be used to further reduce the current network (an example of this behavior can be observed in Table 6.2). In addition, higher level labels enable higher thresholds, sometimes leading to the discovery of very frequent motifs.

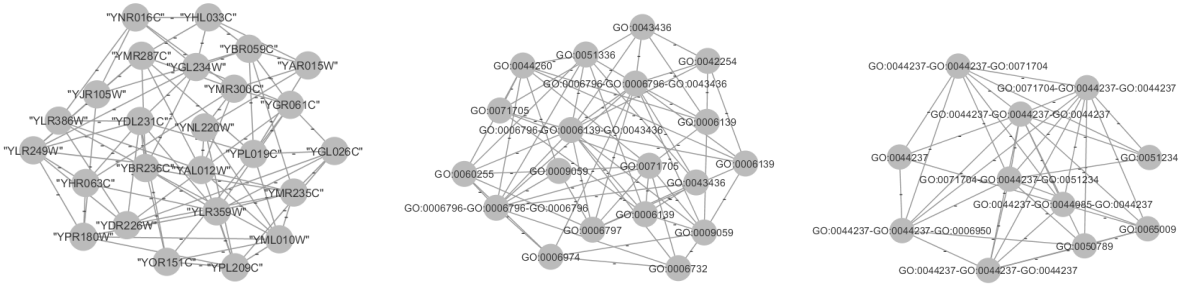
Table 6.2: An example of a hierarchical exploration of the yeast network. The table reports the number of found motifs, the number of nodes and edges, when the network is mapped to different levels of the GO terms hierarchy and then reduced. At higher levels (L1 is higher level than L2 etc.) more motifs pass the threshold.

	th	**Original**	**L5**	**L4**	**L3**	**L2**	**L1**
Motifs	25	0	290	292	319	377	389
Nodes	25	2,361	1,781	1,776	1,607	1,583	1,333
Edges	25	7182	5,234	5,305	5,018	5,020	5,322
Motifs	50	0	240	236	304	388	390
Nodes	50	2,361	1,841	1,889	1,585	1,361	1,581
Edges	50	7,182	5,339	5,429	5,029	5,347	4,990

For example, connections of proteins in Figure 6.3(a) do not show functionalities but those become evident at higher hierarchical levels 6.3(b) and 6.3(c). For example, the frequent relation between proteins which have *GO:0044237, GO:0044237, GO:0044237* as GO terms that are showed in Figure 6.3(c) are only detectable at that level of the hierarchy. Finally, images 6.3(b) and 6.3(c) show that the reduction at a high level of abstraction enables a better understandability of the network.

6.7 Results and Discussion

In this section, we report the time performance, the number of disjoint motifs and the reduction ability of our heuristic algorithms. The time performance is based on the wall clock time required for the execution of the heuristics on all relevant motifs. The number of disjoint motifs is the number of motifs found by each algorithm. The reduction ability is the extent of reduction of networks. All experiments have been performed considering motifs with size = 3 and size = 5 (i.e., having three nodes in



(a) (b) (c)

Figure 6.3: Figures show samples of the yeast network with 25 nodes mapped with original and GO terms labels of our yeast GO hierarchy, and where supernodes have been found by means of SuperNoder. (a) Original network with 25 nodes and 83 edges. (b) Network reduced on low level GO terms hierarchy (19 nodes and 67 edges). (c) Network reduced on high level GO terms hierarchy (11 nodes and 43 edges).

the original graph and three nodes or supernodes after each step of the recursion). H1 has been performed with five shufflings. H2 and H5 adopted subsets of the overlap graphs consisting of 1000 motif nodes. In our simulations, we chose different thresholds in different networks, as shown in Tables 6.3 and 6.4. The reason is that certain thresholds make no sense for certain networks. For example, a threshold of 100 for our food-web network is meaningless because no motifs occur that frequently.

Food-web network

Figure 6.4 reports the performance of the heuristics applied on the food-web network. In this case, heuristics H1, H2 and H5 which exploit repetitive random approaches (H1), sampled *overlap graph* (H2 and H5), and H4 show better performance than others in finding disjoint motifs. Heuristics H3 shows a poor reduction factor on this network. The reason is that there are many motifs with the same sums of degrees, so degree-based heuristics do not work well. Heuristic H1 is the fastest. This holds

Table 6.3: Rows list the number of all motifs, the threshold applied in our experiments and the number of motifs that meet that threshold when L3 labels are considered and motifs have size 3.

Network	**N motifs**	**threshold**	**N repetitive motifs**
Food-Web	20,283	5	5,085
Yeast	96,444	50	49,294
Arabidopsis	268,437	100	155,185

Table 6.4: Rows list the number of all motifs, the threshold applied in our experiments and the number of motifs that meet that threshold when L3 labels are considered and motifs have size 5.

Network	N motifs	threshold	N repetitive motifs
Food-Web	26,841	5	407
Yeast	188,733	50	11,550
Arabidopsis	425,895	100	14,474

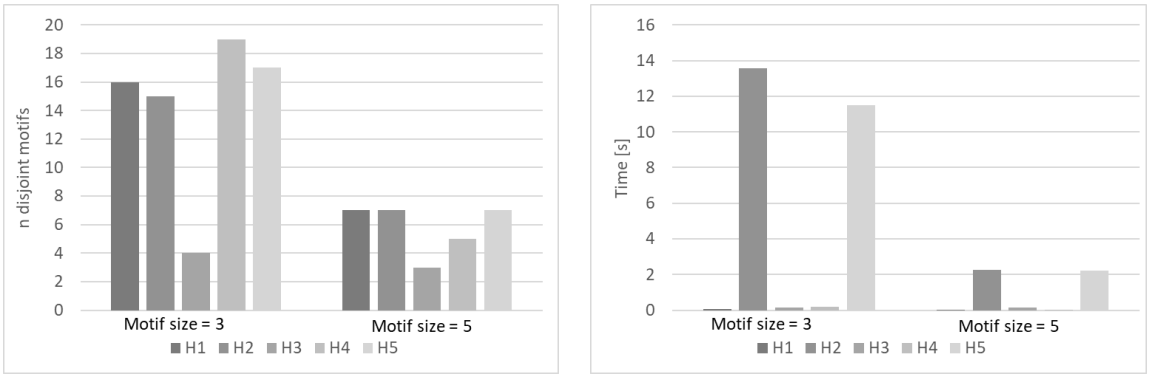
regardless of motif size. In fact, overall, heuristic H1 is both fast and has a good reduction factor.

Yeast network

Figure 6.5 shows the performance on the yeast network. In contrast to the food-web network, heuristics H2 and H5 based on the sampled *overlap graph* do not obtain the best reduction factor. In this case, heuristic H4 enjoys a greater reduction factor. Although heuristics H2 and H5 can find a large number of disjoint motifs, they require excessive time to find a solution, hence, their use on a network of this size might be avoided. The heuristics H1 and H3 are still the fastest.

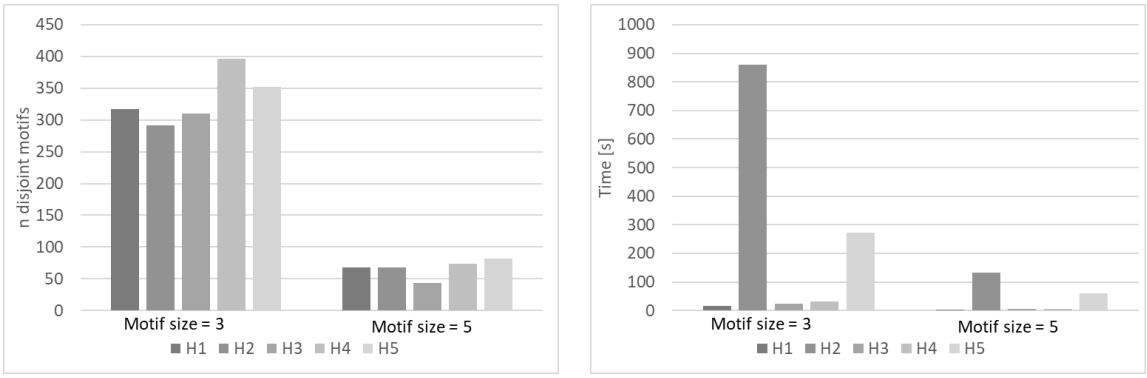
Arabidopsis network

Experimental results on arabidopsis networks are similar to those on the yeast network and the same considerations hold. Note that the arabidopsis network is a



(a) (b)

Figure 6.4: SuperNoder heuristics performance on the food-web network considering motifs of size 3 and 5 in terms of (a) the number of unique motifs found (b) the running time.



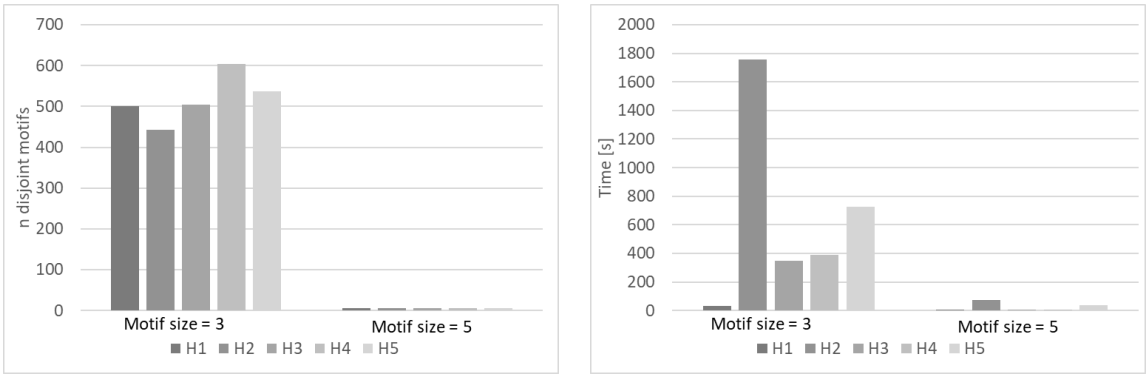
(a) (b)

Figure 6.5: SuperNoder heuristics performance on the yeast network considering motifs of size 3 and 5 in terms of (a) the number of unique motifs found (b) the running time.

Protein-Protein Interaction network like the yeast network but is very different in term of size.

Observations from the Experiments

Heuristic H1 achieves the best time performance and finds a large number of disjoint motifs though not always the maximum number. Heuristic H4 which is slower can sometimes find more disjoint motifs so should be considered if time is available. The size of motifs and the threshold also matter. Larger motifs entail the processing of more data, but there are fewer repetitive motifs (i.e., motifs that exceed the threshold) so the overall time is sometimes less.



(a) (b)

Figure 6.6: SuperNoder heuristics performance on the Arabidopsis network considering motifs of size 3 and 5 in terms of (a) the number of unique motifs found (b) the running time.

In summary, heuristic H1 shows good performance on all types of network since its greedy approach is fast. The resulting reduction may not however be best. Heuristics H2 and H5 which employ sampling are useful for those networks whose overlap graphs are very large. The size of samples can be chosen according to the available computational resources to balance the execution time and memory use. Heuristic H2 should show better reduction performance than H5 when there are few distinct motifs degree values. By contrast, H3 and H4 should be useful for all those networks that have many distinct motifs degree values, because motifs having less probability to overlap are detected faster.

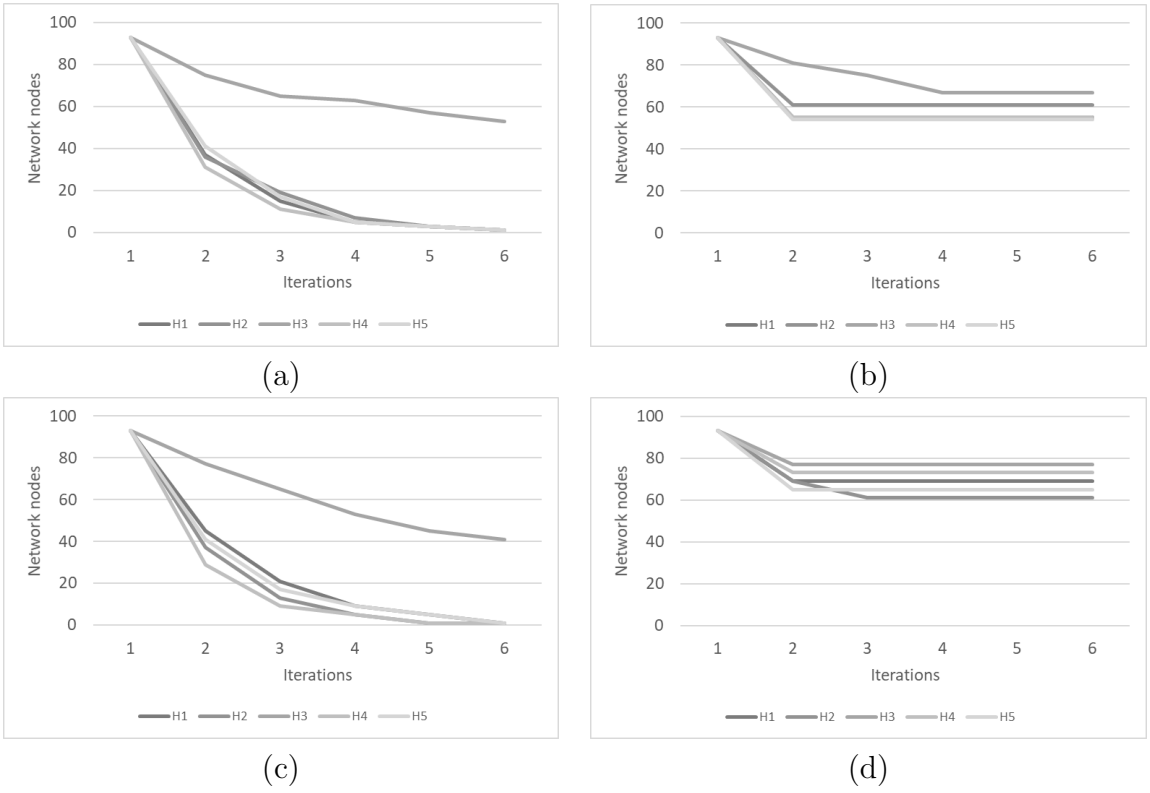


Figure 6.7: Reduction performance on five iterations on the food-web network (a) motifs of size 3 without threshold (b) motifs of size 3 with threshold (c) motifs of size 5 without threshold (d) motifs of size 5 with threshold.

Reduction

Figures 6.7 and 6.8 show the extent of graph reduction on the food-web and yeast networks respectively. Unsurprisingly, lowering the threshold generates more F1 motifs, increasing the number of F3 motifs and reducing the network size. In our example networks, after a few iterations, the networks are no longer reduced. When this plateau-ing happens depends entirely on the data. In addition, the threshold

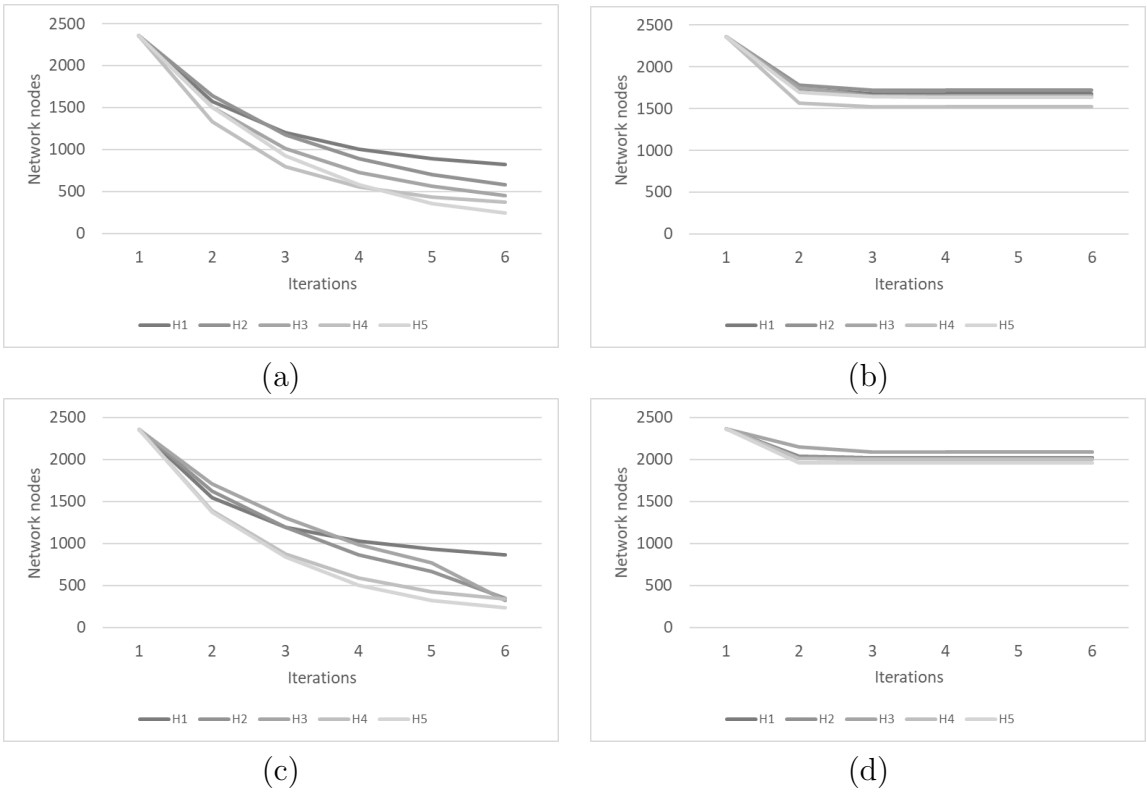
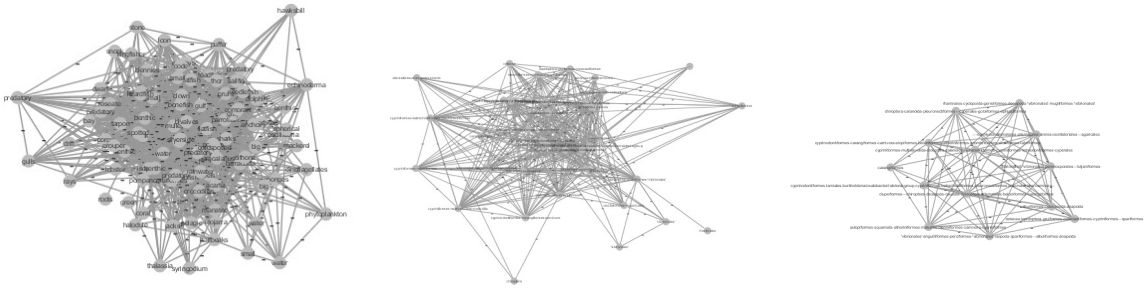


Figure 6.8: Reduction performance on five iterations on the yeast network (a) motifs of size 3 without threshold (b) motifs of size 3 with threshold (c) motifs of size 5 without threshold (d) motifs of size 5 with threshold.

and the motif size both affect the reduction factor, because a small motif has a higher probability of occurring more often (see Table 6.3 and Table 6.4). This is well illustrated by our tests where motifs of size 3 show a greater reduction than motifs of size 5. For an illustration of the extent of reduction, consider Figure 6.9 where (a) shows the original food web network, (b) after one iteration and (c) after two iterations.

Tool description

Figure 6.10 shows the graphical interface of SuperNoder that users without programming skills can adopt to analyze networks. On the left, users can use a panel to create nodes, in the center there is one panel to create edges, and, on the right, a list of parameters the user can set. With the first option users can choose the size of motifs they are interested in. The minimum value is 3. The next option is related to the heuristic that should be employed to find disjoint motifs. The user can also choose the type of network: *direct* or *undirect*. The fourth parameter is the threshold which represents the minimum value each motif should meet to be consid-



(a) (b) (c)

Figure 6.9: Reduction in size of the food web network mapped on the species order (e.g. *kingfisher* mapped on *coraciiformes*). (a) The original network. (b) The network reduced after 1 iteration. (c) The network reduced after 2 iterations.

ered over-represented (it corresponds to the threshold t of the SuperNoder pipeline algorithm). The last required parameter is the number of iterations. In addition, if the user selects the H1 heuristic, he/she can set the number of repetitions to be executed, specific for H1. If the user selects either the H2 or H5 heuristic, he/she can also choose the size of samples. When the *Submit network* button is clicked, the SuperNoder pipeline will be run and results will be printed and shown online (but not saved anywhere).

The output consists of two sections (nodes and edges) for each chosen iteration using the same input format. Supernodes are indicated by the tag *#supernode*.

The code has been developed in Python 3.6 using NetworkX library. SuperNoder functionalities operate on graphs using the standard NetworkX format. The web interface is provided by a python server which runs on a Docker[7] container. SuperNoder is hosted on a GitHub[8] page and distributed as a Docker file with the source code freely available under GPLv3 License.

[7] `https://www.docker.com/`

[8] `https://github.com/danilo-dessi/SuperNoder-v1.0`

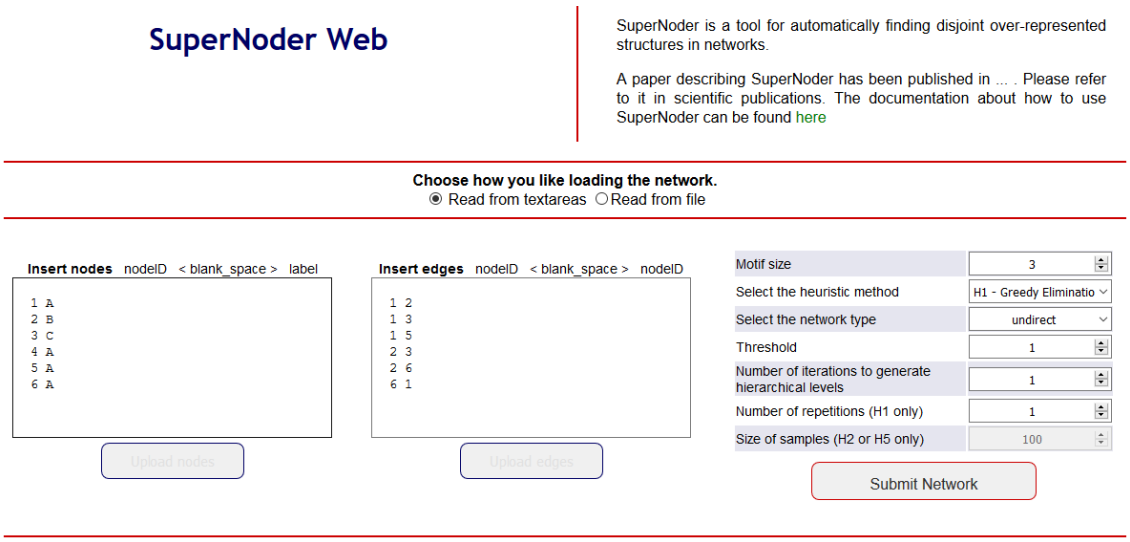


Figure 6.10: SuperNoder web application.

Chapter 7

Conclusions

The use of data in an ever increasing number of domains is becoming an essential aspect for providing novel services to the modern society. In this thesis, the use of data for extracting knowledge and providing services has been investigated.

For the healthcare domain, knowledge from medical transcripts has been involved in a data analysis process to provide a content-based recommender system whose goal was to support physicians in their job. This work addressed the challenge to find out which types of information can be directly processed by machines on large collections of medical reports, combining emergent Cognitive Computing systems in order to return reliable recommendation results. We discussed about the quality of the features for the representation of the medical reports content, underlying how they can capture the semantics from unstructured texts. We also report results about a Machine Learning methodology with two clustering approaches our recommender system currently implements. We used them to handle various VSMs and explained their advantages and uses based on the type of features. In future, novel extraction methodologies to represent knowledge (e.g., word embeddings) can be used on the healthcare domain. In addition, novel data may be useful in order to face ever more difficult issues that can be raised within the healthcare domain.

Within our research work in the E-Learning domain, we presented COCO, a complete and comprehensive online courses collection enriched with stakeholder interactions crawled from Udemy. It presently refers to more than 43K online courses, 16K instructors and 2,5M learners who have provided 4,5M reviews. COCO provides data about courses, learners and instructors, including enrollments, reviews, and wish-lists. Furthermore, we proposed possible use cases supporting online course delivering. The experiments demonstrated that such use cases are challenging and need novel research to manage online courses proliferation. Advanced semantic-based techniques can extract insightful information to support stakeholders during organization and delivery of contents. COCO is expected to support reproducible evaluation in technology-enhanced learning approaches.

Subsequently, a novel framework for micro-learning video classification has been presented. The approach extracts transcripts from videos using novel speech-to-

text systems, exploits Cognitive Computing to move knowledge from texts to vector representations, and applies advanced Machine Learning algorithms for classifying contents. Moreover, it leverages Big Data technologies for fast computation. The experimental results showed how our approach achieves good performance in most cases in term of computational time and precision-recall analysis. Our feature representation combines concepts and keywords extracted from cutting-edge Cognitive Computing tools and exploits the semantic behind the text that traditional approaches fail to capture. Considering the experimental results, we expect our approach powered by Cognitive Computing can facilitate the development of Learning Analytics tools aimed at supporting content managers to arrange micro-learning video collections. We also expect that it will improve how learners explore the online platforms. Learning Analytics services powered by Cognitive Computing promise to shape the future of higher education. Hence, our contribution tries to make a jump-start towards this upcoming era of cognitive-driven education.

Finally, for the E-Learning domain, a Sentiment Analysis case of study has been addressed by combining state-of-the-art Deep Learning methodologies and Word Embedding text representations. We introduced and described a deep neural network aimed to predict a sentiment score for text reviews posted by learners after attending online courses. As most of the current approaches for Sentiment Analysis are built on top of different Word Embedding representations, we showed how some types of word embeddings can better represent the semantics behind the E-Learning context and help supervised models to better predict a sentiment score. Furthermore, considering that word embeddings tend to be sensitive to the context where they are trained, and that the current publicly-available word embeddings were trained on general-purpose resources, we proved that word embeddings generated from E-Learning resources enable to capture more peculiarities of the target domain. What is still missing in this context is the analysis of sentiments by means of ontological resources that can enrich the knowledge of embeddings. Moreover, novel embeddings like BERT can play a relevant role due to their ability to infer syntactical and semantics of words by considering their use in a target context.

Within the Scholarly domain we described a preliminary workflow for producing a scientific knowledge graph from the text of research publications. We built a Scientific knowledge graph derived from a set of 26k publications in the field of the Semantic Web, with the aim to provide a resource that will help researchers to better understand research dynamics. In particular, we focused on extraction of entities and relations for detecting relationships. In our pipeline we faced with issues related to both the extraction and managing of entities and relations in order to model them in a format that can be easily understood. Moreover, we provided an evaluation of our approach by building a gold standard that can be also used for further studies. With our work on Scholarly domain we made a first step in modeling the content of scientific publications for purpose of inclusion and integration within actual academic knowledge graphs. Future work on this research line will include the design of new modules to better capture insightful knowledge and the development of

applications that adopt the knowledge graph for the purpose to explore the research landscape.

Finally, the thesis presents SuperNoder, a novel tool which enables the simplification and compression of graphs based on high frequent motifs. By identifying disjoint motifs, SuperNoder enhances understandability as the network is reduced. Experiments on the biological domain are reported by describing and comparing various algorithms on real networks, both to show the benefits of the approach and to find high-performing algorithms. With the continuous generation of networks within the biological domain, we would also like to adapt SuperNoder to specific tasks, enhancing the detection of relevant motifs for specific biological functions.

In conclusion of this thesis, brief discussions about the research questions introduced in the Chapter 1 are reported.

Q1. How to use existing Semantic Web technologies to retrieve useful information in order to model contents of texts in machine readable formats?

In order to answer to this question, in the Ph.D. work tools like IBM Watson, Framester, word embeddings generators, and others have been employed to prove how semantics based technologies can be adopted to represent the knowledge for specific domain applications. In almost all the cases these technologies have shown improvements against the most used methodologies. In addition, they often have proved to be state-of-the-art technologies that should be used to come up with better results for future applications.

Q2. What are the Machine Learning algorithms more suitable to infer knowledge from the modelled information?

Within the explored domains, many Machine Learning algorithms have been used to address the tasks. For the healthcare domain clustering algorithms have shown good performance in recognizing patterns within the VSMs used to model clinical notes which were not previously labelled. In particular the combination between hierarchical clustering and IBM Watson concepts resulted the best solution to find clinical cases similar to a new one. Within the E-Learning domain, SVM-based classifiers obtained the best results with IBM Watson concepts to assign videos to the right class, outperforming the tf-idf approach. Moreover, Deep Learning models have proved to be efficient and effective to infer sentiments within online reviews for this domain, suggesting that Neural Network models can play an important role to address domain specific tasks for MOOCs. The same applies to the scholarly domain, where a Neural Network proved to enhance results for detecting relevant relationships for the *Semantic Web* domain. Finally, approaches exploited within biology result useful to get insightful knowledge out from biological graphs to detect and study biological functions.

Q3. In literature many general purpose methods can be found to address text-based applications. May these methods be used for specific domain applications?

Across all the experiments reported in the research work, it is clear that many state-of-the-art methods can be applied to address specific tasks in the studied target domains. Nevertheless, it is worth to note that domain experts might not be able to

use ready-made applications for their tasks, and computer science expertise is needed in order to make adjustments to tune the general purpose algorithms to specific and domain-dependent tasks. From the experiments, it turns out that general algorithms need case-by-case adjustments to be exploited for specific domain applications.

Q4. How different techniques and their combinations will impact the performances of specific applications for a target domain?

In order to answer to this research question, results reported in this thesis are often comparative results in order to detect which peculiarities really matter to improve current state-of-the-art approaches for a given task. Each chapter of this thesis ends with the illustration of the obtained results and discussions about them, highlighting which approaches led to the best outcome.

Bibliography

[1] Danilo Dessì, Mauro Dragoni, Gianni Fenu, Mirko Marras, and Diego Reforgiato Recupero. Evaluating neural word embeddings created from online course reviews for sentiment analysis. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 2124–2127. ACM, 2019.

[2] Danilo Dessì, Gianni Fenu, Marras Marras, and Diego Reforgiato Recupero. Bridging learning analytics and cognitive computing for big data classification in micro-learning video collections. *Computers in Human Behavior*, 2019.

[3] Davide Buscaldi, Danilo Dessì, Enrico Motta, Francesco Osborne, and Diego Reforgiato Recupero. Mining scholarly data for fine-grained knowledge graph construction. In *CEUR Workshop Proceedings*, volume 2377, pages 21–30, 2019.

[4] Danilo Dessì, Diego Reforgiato Recupero, Gianni Fenu, and Sergio Consoli. A recommender system of medical reports leveraging cognitive computing and frame semantics. In *Machine Learning Paradigms*, pages 7–30. Springer, 2019.

[5] Gianluca Bardaro, Danilo Dessì, Enrico Motta, Francesco Osborne, and Diego Reforgiato Recupero. Parsing natural language sentences into robot actions. In *International Semantic Web Conference ISWC 2019*, pages 93–96, 2019.

[6] Davide Buscaldi, Danilo Dessì, Enrico Motta, Francesco Osborne, and Diego Reforgiato Recupero. Mining scholarly publications for scientific knowledge graph construction. In *European Semantic Web Conference*, pages 8–12. Springer, 2019.

[7] D. Dessi, M. Dragoni, G. Fenu, M. Marras, and D. Reforgiato Recupero. Deep learning adaptation with word embeddings for sentiment analysis on online course reviews. In *Deep Learning based Approaches for Sentiment Analysis book.*, pages 57–83. Springer, 2020.

[8] Diego Reforgiato Recupero, Danilo Dessì, and Emanuele Concas. A flexible and scalable architecture for human-robot interaction. In *European Conference on Ambient Intelligence*, pages 311–317. Springer, 2019.

[9] Danilo Dessì, Jacopo Cirrone, Diego Reforgiato Recupero, and Dennis Shasha. Supernoder: a tool to discover over-represented modular structures in networks. *BMC bioinformatics*, 19(1):318, 2018.

[10] Danilo Dessì, Gianni Fenu, Mirko Marras, and Diego Reforgiato Recupero. Coco: Semantic-enriched collection of online courses at scale with experimental use cases. In *World Conference on Information Systems and Technologies*, pages 1386–1396. Springer, 2018.

[11] Danilo Dessì, Gianni Fenu, Mirko Marras, and Diego Reforgiato Recupero. Leveraging cognitive computing for multi-class classification of e-learning videos. In *European Semantic Web Conference*, pages 21–25. Springer, 2017.

[12] Danilo Dessì, Diego Reforgiato Recupero, Gianni Fenu, and Sergio Consoli. Exploiting cognitive computing and frame semantic features for biomedical document clustering. In *Proceedings of the Workshop on Semantic Web Solutions for Large-scale Biomedical Data Analytics co-located with 14th Extended Semantic Web Conference, SeWeBMeDA@ESWC 2017*, pages 20–34, 2017.

[13] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[14] Li Deng. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, 3, 2014.

[15] Chu-Cheng Lin, Waleed Ammar, Chris Dyer, and Lori Levin. Unsupervised pos induction with word embeddings. *arXiv preprint arXiv:1503.06760*, 2015.

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[17] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[18] M. Giatsoglou, M. G Vozalis, K. Diamantaras, A. Vakali, G. Sarigiannidis, and K. Chatzisavvas. Sentiment analysis leveraging emotions and word embeddings. *Expert Systems with Applications*, 69:214–224, 2017.

[19] Cicero Dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, 2014.

[20] Yang Li, Quan Pan, Tao Yang, Suhang Wang, Jiliang Tang, and Erik Cambria. Learning word representations for sentiment analysis. *Cognitive Computation*, 9(6):843–851, 2017.

[21] Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*, pages 151–161. Association for Computational Linguistics, 2011.

[22] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[23] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[24] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov. Fasttext. zip: Compressing text classification models. *arXiv:1612.03651*, 2016.

[25] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[26] S. Ji, N. Satish, S. Li, and P. Dubey. Parallelizing word2vec in multi-core and many-core architectures. *arXiv preprint arXiv:1611.06172*, 2016.

[27] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.

[28] Ying Chen, JD Elenee Argentinis, and Griff Weber. Ibm watson: how cognitive computing can be applied to big data challenges in life sciences research. *Clinical therapeutics*, 38(4):688–701, 2016.

[29] Charles Fillmore. Frame semantics. *Linguistics in the morning calm*, pages 111–137, 1982.

[30] F. C. Baker, C. J. Fillmore, and J. B. Lowe. The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (Volume 1)*, ACL '98, pages 86–90, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics.

[31] A. Gangemi. What's in a Schema? *Cambridge University Press, Cambridge*, pages 144–182, 2010.

[32] Xindong Wu, Xingquan Zhu, Gong-Qing Wu, and Wei Ding. Data mining with big data. *IEEE transactions on knowledge and data engineering*, 26(1):97–107, 2014.

[33] Francesco Maria Aymerich, Gianni Fenu, and Simone Surcis. A real time financial system based on grid and cloud computing. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1219–1220. ACM, 2009.

[34] Gianni Fenu and Marco Nitti. Strategies to carry and forward packets in vanet. *Digital Information and Communication Technology and Its Applications*, pages 662–674, 2011.

[35] Alberto Fernández, Sara del Río, Victoria López, Abdullah Bawakid, María J del Jesus, José M Benítez, and Francisco Herrera. Big data with cloud computing: an insight on the computing environment, mapreduce, and programming frameworks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(5):380–409, 2014.

[36] Rashmi Mishra, Jiantao Bian, Marcelo Fiszman, Charlene R Weir, Siddhartha Jonnalagadda, Javed Mostafa, and Guilherme Del Fiol. Text summarization in the biomedical domain: a systematic review of recent research. *Journal of biomedical informatics*, 52:457–467, 2014.

[37] Sergio Consoli, Diego Reforgiato Recupero, and Milan Petkovic. *Data Science for Healthcare - Methodologies and Applications*. Springer Nature, Switzerland, 2019.

[38] S. Bleik, M. Mishra, J. Huan, and M. Song. Text categorization of biomedical data sets using graph kernels and a controlled vocabulary. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(5):1211–1217, 2013.

[39] A. M. Cohen and W. R. Hersh. A survey of current work in biomedical text mining. *Briefings in bioinformatics*, 6(1):57–71, 2005.

[40] R. Toor and I. Chana. Application of IT in Healthcare: A systematic review. *ACM SIGBioinformatics Rec.*, 6(2):1–8, 2016.

[41] V. Presutti, S. Consoli, A. G. Nuzzolese, D. Reforgiato Recupero, A. Gangemi, I. Bannour, and H. Zargayouna. Uncovering the semantics of wikipedia pagelinks. In *Lecture Notes in Computer Science*, volume 8876, pages 413–428, 2014.

[42] V. Presutti, A. G. Nuzzolese, S. Consoli, A. Gangemi, and D. Reforgiato Recupero. From hyperlinks to semantic web properties using open knowledge extraction. *Semantic Web*, 7(4):351–378, 2016.

[43] M. Lushnov, T. Safin, M. Lapaev, and N. Zhukova. Medical text processing for SMDA project. In *EMSA-RMed@ESWC*, 2016.

[44] S. Consoli and N. I. Stilianakis. A quartet method based on variable neighbourhood search for biomedical literature extraction and clustering. *International Transactions in Operational Research*, 24(3):537–558, 2017.

[45] M. Chernyshevich and V. Stankevitch. IHS-RD-BELARUS: Clinical named entities identification in French medical texts. *Physiology*, 279:291, 2015.

[46] A. S. Yeh, L. Hirschman, and A. A. Morgan. Evaluation of text data mining for database curation: Lessons learned from the KDD Challenge Cup. *Bioinformatics*, 19 Suppl. 1:331–339, 2003.

[47] Y. Regev, M. Finkelstein-Landau, and R. Feldman. Rule-based extraction of experimental evidence in the biomedical domain: The KDD Cup 2002 (task 1). *ACM SIGKDD Explorations Newsletter*, 4(2):90–92, 2002.

[48] I. Donaldson, J. Martin, B. de Bruijn, C. Wolting, V. Lay, B. Tuekam, S. Zhang, B. Baskin, G. D. Bader, K. Michalickova, T. Pawson, and C. W. V. Hogue. PreBIND and Textomy - Mining the biomedical literature for protein-protein interactions using a support vector machine. *BMC Bioinformatics*, 4(1):11, 2003.

[49] S. Shehata, F. Karray, and M. Kamel. An efficient concept-based mining model for enhancing text clustering. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1360–1371, 2010.

[50] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[51] Marco Degemmis, Pasquale Lops, and Giovanni Semeraro. A content-collaborative recommender that exploits wordnet-based user profiles for neighborhood formation. *User Modeling and User-Adapted Interaction*, 17(3):217–255, 2007.

[52] J. Gu, W. Feng, J. Zeng, H. Mamitsuka, and S. Zhu. Efficient semisupervised MEDLINE document clustering with MeSH-semantic and global-content constraints. *IEEE transactions on cybernetics*, 43(4):1265–1276, 2013.

[53] Stefano Bromuri, Damien Zufferey, Jean Hennebert, and Michael Schumacher. Multi-label classification of chronically ill patients with bag of words and supervised dimensionality reduction algorithms. *Journal of biomedical informatics*, 51:165–175, 2014.

[54] Ben J Marafino, Jason M Davies, Naomi S Bardach, Mitzi L Dean, R Adams Dudley, and John Boscardin. N-gram support vector machines for scalable procedure and diagnosis classification, with applications to clinical free text data from the intensive care unit. *Journal of the American Medical Informatics Association*, 21(5):871–875, 2014.

[55] Mark Hughes, I Li, Spyros Kotoulas, and Toyotaro Suzumura. Medical text classification using convolutional neural networks. *Stud Health Technol Inform*, 235:246–50, 2017.

[56] Rui-Wei Zhao, Guo-Zheng Li, Jia-Ming Liu, and Xiao Wang. Clinical multi-label free text classification by exploiting disease label relation. In *Bioinformatics and Biomedicine (BIBM), 2013 IEEE International Conference on*, pages 311–315. IEEE, 2013.

[57] Kinga Glinka, Rafał Woźniak, and Danuta Zakrzewska. Improving multi-label medical text classification by feature selection. In *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2017 IEEE 26th International Conference on*, pages 176–181. IEEE, 2017.

[58] Tal Baumel, Jumana Nassour-Kassis, Michael Elhadad, and Noémie Elhadad. Multi-label classification of patient notes a case study on icd code assignment. *CoRR*, abs/1709.09587, 2017.

[59] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saied Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919*, 2017.

[60] Xiaodan Zhang, Liping Jing, Xiaohua Hu, Michael Ng, Jiali Xia Jiangxi, and Xiaohua Zhou. Medical document clustering using ontology-based term similarity measures. *International Journal of Data Warehousing and Mining (IJDWM)*, 4(1):62–73, 2008.

[61] Yunxuan Zhang, Ziping He, Ji-Jiang Yang, Qing Wang, and Jianqiang Li. Restructuring and specific similarity computation of electronic medical records. In *Computer Software and Applications Conference (COMPSAC), 2017 IEEE 41st Annual*, volume 2, pages 230–235. IEEE, 2017.

[62] Annie T Chen. Exploring online support spaces: using cluster analysis to examine breast cancer, diabetes and fibromyalgia support groups. *Patient education and counseling*, 87(2):250–257, 2012.

[63] Yingjie Lu, Pengzhu Zhang, and Shasha Deng. Exploring health-related topics in online health community using cluster analysis. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 802–811. IEEE, 2013.

[64] Martin Wiesner and Daniel Pfeifer. Adapting recommender systems to the requirements of personal health record systems. In *Proceedings of the 1st ACM International Health Informatics Symposium*, pages 410–414. ACM, 2010.

[65] Darcy A Davis, Nitesh V Chawla, Nicholas Blumm, Nicholas Christakis, and Albert-László Barabási. Predicting individual disease risk based on medical history. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 769–778. ACM, 2008.

[66] Yin Zhang, Min Chen, Dijiang Huang, Di Wu, and Yong Li. idoctor: Personalized and professionalized medical recommendations based on hybrid matrix factorization. *Future Generation Computer Systems*, 66:30–35, 2017.

[67] Mercato e-learning: trend e previsioni 2017-2021, 2017. Accessed: 2017-11-20.

[68] G2crowd grid for online course providers, 2017. Accessed: 2017-11-20.

[69] Ryan Shaun Baker and Paul Salvador Inventado. Educational data mining and learning analytics. In *Learning analytics*, pages 61–75. Springer, 2014.

[70] Wanli Xing, Xin Chen, Jared Stein, and Michael Marcinkowski. Temporal predication of dropouts in moocs: Reaching the low hanging fruit through stacking generalization. *Computers in Human Behavior*, 58:119–129, 2016.

[71] Miri Barak, Abeer Watted, and Hossam Haick. Motivation to learn in massive open online courses: Examining aspects of language and social engagement. *Computers & Education*, 94:49–60, 2016.

[72] Carlos Delgado Kloos, Patrick Jermann, Mar Pérez-Sanagustín, Daniel T Seaton, and Su White. *Digital Education: Out to the World and Back to the Campus*. Springer, 2017.

[73] Maria-Cruz Valiente, Miguel-Angel Sicilia, Elena Garcia-Barriocanal, and Enayat Rajabi. Adopting the metadata approach to improve the search and analysis of educational resources for online learning. *Computers in Human Behavior*, 51:1134–1141, 2015.

[74] Subhasree Basu, Yi Yu, and Roger Zimmermann. Fuzzy clustering of lecture videos based on topic modeling. In *Content-Based Multimedia Indexing (CBMI), 2016 14th International Workshop on*, pages 1–6. IEEE, 2016.

[75] K. L. Cela, M. Á. Sicilia, and S. Sánchez. Social network analysis in e-learning environments. *Educational Psychology Review*, 27(1):219–246, 2015.

[76] G. Esparza, A. de Luna, A. O. Zezzatti, A. Hernandez, J. Ponce, M. Álvarez, E. Cossio, and J. de Jesus Nava. A sentiment analysis model to analyze students reviews of teacher performance using support vector machines. In *Int. Symp. on Distributed Computing and Artificial Intelligence*, pages 157–164. Springer, 2017.

[77] Charu C Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. *Mining text data*, pages 163–222, 2012.

[78] Vladimir Estivill-Castro, Carla Limongelli, Matteo Lombardi, and Alessandro Marani. Dajee: A dataset of joint educational entities for information retrieval in technology enhanced learning. In *Proceedings of the 39th International ACM SIGIR*, pages 681–684. ACM, 2016.

[79] Nikolaos Pappas and Andrei Popescu-Belis. Combining content with user preferences for ted lecture recommendation. In *Content-Based Multimedia Indexing (CBMI), 2013 11th International Workshop on*, pages 47–52. IEEE, 2013.

[80] Merlot, 2017. Accessed: 2017-11-20.

[81] Andrew Dean Ho, Justin Reich, Sergiy O Nesterko, Daniel Thomas Seaton, Tommy Mullaney, Jim Waldo, and Isaac Chuang. Harvardx and mitx: The first year of open online courses. 2014.

[82] Mace. Accessed: 2017-11-20.

[83] Dragan Gašević, Shane Dawson, Tim Rogers, and Danijela Gasevic. Learning analytics should not promote one size fits all: The effects of instructional conditions in predicting academic success. *The Internet and Higher Education*, 28:68–84, 2016.

[84] Faisal M Almutairi, Nicholas D Sidiropoulos, and George Karypis. Context-aware recommendation-based learning analytics using tensor and coupled matrix factorization. *IEEE Journal of Selected Topics in Signal Processing*, 2017.

[85] Evandro B Costa, Baldoino Fonseca, Marcelo Almeida Santana, Fabrísia Ferreira de Araújo, and Joilson Rego. Evaluating the effectiveness of educational

data mining techniques for early prediction of students' academic failure in introductory programming courses. *Computers in Human Behavior*, 73:247–256, 2017.

[86] George Siemens. Learning analytics: The emergence of a discipline. *American Behavioral Scientist*, 57(10):1380–1400, 2013.

[87] Siddu P Algur and Prashant Bhat. Web video mining: Metadata predictive analysis using classification techniques. *International Journal of Information Technology and Computer Science (IJITCS)*, 8(2):69, 2016.

[88] Nikolaos Gkalelis and Vasileios Mezaris. Video event detection using generalized subclass discriminant analysis and linear support vector machines. In *Proceedings of international conference on multimedia retrieval*, page 25. ACM, 2014.

[89] Li Yang and Xiaokun Wang. Online appearance manifold learning for video classification and clustering. In *International Conference on Computational Science and Its Applications*, pages 551–561. Springer, 2016.

[90] Issam Feki, Anis Ben Ammar, and Adel M Alimi. Automatic environmental sound concepts discovery for video retrieval. *International Journal of Multimedia Information Retrieval*, 5(2):105–115, 2016.

[91] Joseph G Ellis, Brendan Jou, and Shih-Fu Chang. Why we watch the news: A dataset for exploring sentiment in broadcast video news. In *Proceedings of the 16th International Conference on Multimodal Interaction*, pages 104–111. ACM, 2014.

[92] Juho Kim, Philip J Guo, Carrie J Cai, Shang-Wen Daniel Li, Krzysztof Z Gajos, and Robert C Miller. Data-driven interaction techniques for improving navigation of educational videos. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 563–572. ACM, 2014.

[93] Giovanni Garibotto, Pierpaolo Murrieri, Alessandro Capra, Stefano De Muro, Ugo Petillo, Francesco Flammini, Mariana Esposito, Cocetta Pragliola, Giuseppe Di Leo, Roald Lengu, et al. White paper on industrial applications of computer vision and pattern recognition. In *International Conference on Image Analysis and Processing*, pages 721–730. Springer, 2013.

[94] Huizhong Chen, Matthew Cooper, Dhiraj Joshi, and Bernd Girod. Multimodal language models for lecture video retrieval. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1081–1084. ACM, 2014.

[95] Shangfei Wang and Qiang Ji. Video affective content analysis: a survey of state-of-the-art methods. *IEEE Transactions on Affective Computing*, 6(4):410–430, 2015.

[96] Lu Jiang, Shoou-I Yu, Deyu Meng, Teruko Mitamura, and Alexander G Hauptmann. Bridging the ultimate semantic gap: A semantic search engine for internet videos. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 27–34. ACM, 2015.

[97] Sangmin Oh, Scott McCloskey, Ilseo Kim, Arash Vahdat, Kevin J Cannons, Hossein Hajimirsadeghi, Greg Mori, AG Amitha Perera, Megha Pandey, and Jason J Corso. Multimedia event detection with multimodal feature fusion and temporal concept localization. *Machine vision and applications*, 25(1):49–69, 2014.

[98] Jeffrey Dalton, James Allan, and Pranav Mirajkar. Zero-shot video retrieval using content and concepts. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 1857–1860. ACM, 2013.

[99] Marcos Vinicius Macedo Borges, Julio Cesar dos Reis, and Guilherme Pereira Gribeler. Empirical analysis of semantic metadata extraction from video lecture subtitles. In *2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 301–306. IEEE, 2019.

[100] Angelo A Salatino, Thiviyan Thanapalasingam, Andrea Mannocci, Francesco Osborne, and Enrico Motta. Classifying research papers with the computer science ontology. In *ISWC (P&D/Industry/BlueSky). CEUR Workshop Proceedings*, volume 2180, 2018.

[101] Angelo A Salatino, Thiviyan Thanapalasingam, Andrea Mannocci, Francesco Osborne, and Enrico Motta. The computer science ontology: a large-scale taxonomy of research areas. In *ISWC*, pages 187–205, 2018.

[102] Dejan Marković, Jigyasa Popat, Fabio Antonacci, Augusto Sarti, and T Kishore Kumar. An informed separation algorithm based on sound field mapping for speech recognition systems. In *Acoustic Signal Enhancement (IWAENC), 2016 IEEE International Workshop on*, pages 1–5. IEEE, 2016.

[103] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[104] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[105] Marcos Wander Rodrigues, Luiz Enrique Zárate, and Seiji Isotani. Educational data mining: a review of evaluation process in the e-learning. *Telematics and Informatics*, 2018.

[106] Fabio Clarizia, Francesco Colace, Massimo De Santo, Marco Lombardi, Francesco Pascale, and Antonio Pietrosanto. E-learning and sentiment analysis: a case study. In *Proceedings of the 6th International Conference on Information and Education Technology*, pages 111–118. ACM, 2018.

[107] Ganpat Singh Chauhan, Preksha Agrawal, and Yogesh Kumar Meena. Aspect-based sentiment analysis of students feedback to improve teaching–learning process. In *Information and Communication Technology for Intelligent Systems*, pages 259–266. Springer, 2019.

[108] Pilar Rodriguez, Alvaro Ortigosa, and Rosa M Carro. Extracting emotions from texts in e-learning environments. In *2012 Sixth International Conference on Complex, Intelligent, and Software Intensive Systems*, pages 887–892. IEEE, 2012.

[109] Hassan Saif, Yulan He, Miriam Fernandez, and Harith Alani. Semantic patterns for sentiment analysis of twitter. In *International Semantic Web Conference*, pages 324–340. Springer, 2014.

[110] Peter D Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188, 2010.

[111] Abinash Tripathy, Ankit Agrawal, and Santanu Kumar Rath. Classification of sentiment reviews using n-gram machine learning approach. *Expert Systems with Applications*, 57:117–126, 2016.

[112] Oscar Araque, Ignacio Corcuera-Platas, J Fernando Sanchez-Rada, and Carlos A Iglesias. Enhancing deep learning sentiment analysis with ensemble techniques in social applications. *Expert Systems with Applications*, 77:236–246, 2017.

[113] Soujanya Poria, Erik Cambria, and Alexander Gelbukh. Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2539–2544, 2015.

[114] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432, 2015.

[115] L. Boratto, S. Carta, G. Fenu, and R. Saia. Using neural word embeddings to model user behavior and detect user segments. *Knowledge-Based Systems*, 108:5–14, 2016. cited By 10.

[116] Y. Li, Q. Pan, T. Yang, S. Wang, J. Tang, and E. Cambria. Learning word representations for sentiment analysis. *Cogn. Computation*, 9(6):843–851, 2017.

[117] D. Tang, F. Wei, B. Qin, N. Yang, T. Liu, and M. Zhou. Sentiment embeddings with applications to sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, 28(2):496–509, 2016.

[118] Z. Zhang and M. Lan. Learning sentiment-inherent word embedding for word-level and sentence-level sentiment analysis. In *2015 International Conference on Asian Language Processing (IALP)*, pages 94–97, Oct 2015.

[119] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proc. of the Annual Meeting of the Assoc. for Computational Linguistics: Human Language Technologies - Vol. 1*, pages 142–150, 2011.

[120] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, pages 1555–1565, 2014.

[121] E. Rudkowsky, M. Haselmayer, M. Wastian, M. Jenny, S. Emrich, and M. Sedlmair. More than bags of words: Sentiment analysis with word embeddings. *Communication Methods and Measures*, 12(2-3):140–157, 2018.

[122] D. Reforgiato Recupero and E. Cambria. Eswc'14 challenge on concept-level sentiment analysis. In Valentina Presutti, Milan Stankovic, Erik Cambria, Iván Cantador, Angelo Di Iorio, Tommaso Di Noia, Christoph Lange, Diego Reforgiato Recupero, and Anna Tordai, editors, *Semantic Web Evaluation Challenge*, pages 3–20, Cham, 2014. Springer International Publishing.

[123] D. Reforgiato Recupero, M. Dragoni, and V. Presutti. Eswc 15 challenge on concept-level sentiment analysis. In Fabien Gandon, Elena Cabrio, Milan Stankovic, and Antoine Zimmermann, editors, *Semantic Web Evaluation Challenges*, pages 211–222, Cham, 2015. Springer International Publishing.

[124] M. Dragoni and D. Reforgiato Recupero. Challenge on fine-grained sentiment analysis within eswc2016. In Harald Sack, Stefan Dietze, Anna Tordai, and Christoph Lange, editors, *Semantic Web Challenges*, pages 79–94, Cham, 2016. Springer International Publishing.

[125] D. Reforgiato Recupero, E. Cambria, and E. Di Rosa. Semantic sentiment analysis challenge eswc2017. In *Semantic Web Challenges*, pages 109–123. Springer, 2017.

[126] Davide Buscaldi, Aldo Gangemi, and Diego Reforgiato Recupero. *Semantic Web Challenges: Fifth SemWebEval Challenge at ESWC 2018, Heraklion, Crete, Greece, June 3 - June 7, 2018, Revised Selected Papers*. Springer Publishing Company, Incorporated, 3st edition, 2018.

[127] Thanh Vu, Dat Quoc Nguyen, Xuan-Son Vu, Dai Quoc Nguyen, Michael Catt, and Michael Trenell. NIHRIO at semeval-2018 task 3: A simple and accurate neural network model for irony detection in twitter. *CoRR*, abs/1804.00520, 2018.

[128] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *CoRR*, abs/cs/0506075, 2005.

[129] M. Atzeni and D. Reforgiato. Deep learning and sentiment analysis for human-robot interaction. In *Europ. Semantic Web Conference*, pages 14–18. Springer, 2018.

[130] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[131] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48, 2016.

[132] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[133] Andrea Giovanni Nuzzolese, Anna Lisa Gentile, Valentina Presutti, and Aldo Gangemi. Conference linked data: the scholarlydata project. In *ISWC*, pages 150–158. Springer, 2016.

[134] Silvio Peroni, David Shotton, and Fabio Vitali. One year of the opencitations corpus. In *ISWC*, pages 184–192. Springer, 2017.

[135] Sören Auer, Viktor Kovtun, Manuel Prinz, Anna Kasprzik, Markus Stocker, and Maria Esther Vidal. Towards a knowledge graph for science. In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*, page 1. ACM, 2018.

[136] Sepideh Mesbah, Christoph Lofi, Manuel Valle Torre, Alessandro Bozzon, and Geert-Jan Houben. Tse-ner: An iterative approach for long-tail entity extraction in scientific publications. In *ISWC*, pages 127–143. Springer, 2018.

[137] Aldo Gangemi, Valentina Presutti, Diego Reforgiato Recupero, Nuzzolese, et al. Semantic Web Machine Reading with FRED. *Semantic Web*, 8(6):873–893, 2017.

[138] Jose L Martinez-Rodriguez, Ivan Lopez-Arevalo, and Ana B Rios-Alvarado. Openie-based approach for knowledge graph construction from text. *Expert Systems with Applications*, 113:339–355, 2018.

[139] Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the EMNLP 2018 Conference*, pages 3219–3232, 2018.

[140] Andrea Moro, Alessandro Raganato, and Roberto Navigli. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244, 2014.

[141] James R Curran, Stephen Clark, and Johan Bos. Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 33–36, 2007.

[142] Francesco Corcoglioniti, Marco Rospocher, and Alessio Palmero Aprosio. A 2-phase frame-based knowledge extraction framework. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 354–361. ACM, 2016.

[143] Kata Gábor, Davide Buscaldi, Anne-Kathrin Schumann, Behrang QasemiZadeh, Haifa Zargayouna, and Thierry Charnois. Semeval-2018 task 7: Semantic relation extraction and classification in scientific papers. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 679–688, 2018.

[144] Angelo Salatino, Francesco Osborne, Thiviyan Thanapalasingam, and Enrico Motta. The cso classifier: Ontology-driven detection of research topics in scholarly articles. 2019.

[145] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the ACL and the 7th IJCNLP*, volume 1, pages 344–354, 2015.

[146] Simone Angioni, Francesco Osborne, Angelo A Salatino, Diego Reforgiato, and Enrico Motta Recupero. Integrating knowledge graphs for comparing the scientific output of academia and industry. In *International Semantic Web Conference ISWC 2019*, 2019.

[147] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[148] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69(2):026113, February 2004.

[149] Peter J. Mucha, Thomas Richardson, Kevin Macon, Mason A. Porter, and Jukka-Pekka Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980):876, 05 2010.

[150] Clara Granell, Sergio Gómez, and Alex Arenas. Hierarchical multiresolution method to overcome the resolution limit in complex networks. *International Journal of Bifurcation and Chaos*, 22(07):1250171, 2017/10/30 2012.

[151] Brian Ball, Brian Karrer, and M. E. J. Newman. Efficient and principled method for detecting communities in networks. *Phys. Rev. E*, 84:036103, Sep 2011.

[152] Austin R. Benson, David F. Gleich, and Jure Leskovec. Higher-order organization of complex networks. *CoRR*, abs/1612.08447, 2016.

[153] Jaewon Yang and Jure Leskovec. Overlapping communities explain core-periphery organization of networks. *Proceedings of the IEEE*, 102, 2014.

[154] Zhao Yang, René Algesheimer, and Claudio J Tessone. A comparative analysis of community detection algorithms on artificial networks. *Scientific reports*, 6:30750, 2016.

[155] T. S. Evans and R. Lambiotte. Line graphs, link partitions, and overlapping communities. *Phys. Rev. E*, 80:016105, Jul 2009.

[156] Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 06 2005.

[157] Xiaohua Shi, Hongtao Lu, and Guanbo Jia. Adaptive overlapping community detection with bayesian nonnegative matrix factorization. In *International Conference on Database Systems for Advanced Applications*, pages 339–353. Springer, 2017.

[158] Marta Sales-Pardo, Roger Guimer, Andr A. Moreira, and Lus A. Nunes Amaral. Extracting the hierarchical organization of complex systems. *Proceedings of the National Academy of Sciences*, 104(39):15224–15229, 2007.

[159] Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. Mixed membership stochastic blockmodels. *Journal of machine learning research : JMLR*, 9:1981–2014, 09 2008.

[160] Yong-Yeol Ahn, James P. Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 08 2010.

[161] Rasha Elhesha and Tamer Kahveci. Identification of large disjoint motifs in biological networks. *BMC bioinformatics*, 17(1):408, 2016.

[162] Falk Schreiber and Henning Schwobbermeyer. Frequency concepts and pattern detection for the analysis of motifs in networks. *Lecture Notes in Computer Science*, 3737:89–104, 2005.

[163] Marc Legeay, Béatrice Duval, and Jean-Pierre Renou. Differential functional analysis and change motifs in gene networks to explore the role of anti-sense transcription. In *International Symposium on Bioinformatics Research and Applications*, pages 117–126. Springer, 2016.

[164] Benjamin Baiser, Rasha Elhesha, and Tamer Kahveci. Motifs in the assembly of food web networks. *Oikos*, 125(4):480–491, 2016.

[165] Angelo B Monteiro and Lucas Del Bianco Faria. The interplay between population stability and food-web topology predicts the occurrence of motifs in complex food-webs. *Journal of theoretical biology*, 409:165–171, 2016.

[166] E McDonald-Madden, R Sabbadin, ET Game, PWJ Baxter, I Chadès, and HP Possingham. Using food-web theory to conserve ecosystems. *Nature communications*, 7:10245, 2016.

[167] Rahmtin Rotabi, Krishna Kamath, Jon Kleinberg, and Aneesh Sharma. Detecting strong ties using network motifs. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 983–992. International World Wide Web Conferences Steering Committee, 2017.

[168] Shanjia Wang, Yuhua Zhang, Hui Wang, Zihan Huang, Xiaofei Wang, and Tianpeng Jiang. Large scale measurement and analytics on social groups

of device-to-device sharing in mobile social networks. *Mobile Networks and Applications*, pages 1–13, 2017.

[169] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 555–564. ACM, 2017.

[170] Austin R Benson, David F Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016.

[171] Lawrence B Holder, Diane J Cook, Surnjani Djoko, et al. Substucture discovery in the subdue system. In *KDD workshop*, pages 169–180, 1994.

[172] Sebastian Wernicke and Florian Rasche. Fanmod: a tool for fast network motif detection. *Bioinformatics*, 22(9):1152–1153, 2006.

[173] Falk Schreiber and Henning Schwöbbermeyer. Mavisto: a tool for the exploration of network motifs. *Bioinformatics*, 21(17):3572–3574, 2005.

[174] Jin Chen, Wynne Hsu, Mong Li Lee, and See-Kiong Ng. Nemofinder: Dissecting genome-wide protein-protein interactions with meso-scale network motifs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 106–115. ACM, 2006.

[175] Zahra Razaghi Moghadam Kashani, Hayedeh Ahrabian, Elahe Elahi, Abbas Nowzari-Dalini, Elnaz Saberi Ansari, Sahar Asadi, Shahin Mohammadi, Falk Schreiber, and Ali Masoudi-Nejad. Kavosh: a new algorithm for finding network motifs. *BMC bioinformatics*, 10(1):318, 2009.

[176] Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, 2004.

[177] Saeed Omidi, Falk Schreiber, and Ali Masoudi-Nejad. Moda: an efficient algorithm for network motif discovery in biological networks. *Genes & genetic systems*, 84(5):385–395, 2009.

[178] Ali Masoudi-Nejad, Falk Schreiber, and Zahra Razaghi Moghadam Kashani. Building blocks of biological networks: a review on major network motif discovery algorithms. *IET systems biology*, 6(5):164–174, 2012.

[179] Cristina Bondavalli Robert E. Ulanowicz and Michael S. Egnotovich. *Network analysis of trophic dynamics in South Florida ecosystems–The Florida Bay Ecosystem: Annual Report to the U.S. Geological Survey.* 1997.

[180] Dongbo Bu, Yi Zhao, Lun Cai, Hong Xue, Xiaopeng Zhu, Hongchao Lu, Jingfen Zhang, Shiwei Sun, Lunjiang Ling, Nan Zhang, et al. Topological structure analysis of the protein–protein interaction network in budding yeast. *Nucleic acids research*, 31(9):2443–2450, 2003.

[181] Arabidopsis Interactome Mapping Consortium et al. Evidence for network evolution in an arabidopsis interactome map. *Science*, 333(6042):601–607, 2011.

[182] Luigi Pietro Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. An improved algorithm for matching large graphs. In *3rd IAPR-TC15 workshop on graph-based representations in pattern recognition*, pages 149–159, 2001.

[183] Ravi Boppana and Magnús M Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT Numerical Mathematics*, 32(2):180–196, 1992.

[184] Stefan Wuchty and Eivind Almaas. Peeling the yeast protein network. *Proteomics*, 5(2):444–449, 2005.