Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Deep neural network for hierarchical extreme multi-label text classification[☆]

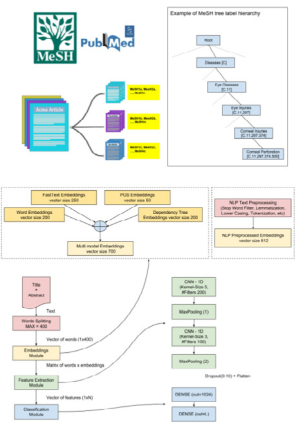
Francesco Gargiulo [a,*], Stefano Silvestri [a,b], Mario Ciampi [a], Giuseppe De Pietro [a]

[a] *Institute for High Performance Computing and Networking of National Research Council, ICAR-CNR, Via Pietro Castellino 111 - 80131, Naples, Italy*
[b] *Department of Engineering, University of Naples "Parthenope", Centro Direzionale di Napoli, Isola C4 - 80143, Naples, Italy*

HIGHLIGHTS

- Deep Neural Network architecture for extreme multilabel text classification.
- Multi-label classification problem with a huge label space hierarchically organized.
- Comparison among different word-embeddings methods for text representation.
- Definition of a method for label set expansion exploiting the label hierarchy.
- Experimental assessment based on flat and hierarchical measures.

GRAPHICAL ABSTRACT


MeSH
PubMed
Example of MeSH tree label hierarchy
Root
Diseases [C]
Eye Diseases [C.11]
Eye Injuries [C.11.297]
Corneal Injuries [C.11.297.374]
Corneal Perforation [C.11.297.374.500]
Acme Article
FastText Embeddings vector size 250
POS Embeddings vector size 50
Word Embeddings vector size 200
Dependency Tree Embeddings vector size 200
Multi model Embeddings vector size 700
NLP Text Preprocessing (Stop Word Filter, Lemmatization, Lower Casing, Tokenization, etc)
NLP Preprocessed Embeddings vector size 512
Title + Abstract
Text
Words Splitting MAX = 400
Vector of words (1x400)
Embeddings Module
Matrix of words x embeddings
Feature Extraction Module
Vector of features (1xN)
Classification Module
CNN - 1D (Kernel-Size 5, #Filters 200)
MaxPooling (1)
CNN - 1D (Kernel-Size 3, #Filters 100)
MaxPooling (2)
Dropout(0.10) + Flatten
DENSE (out=1024)
DENSE (out=L)

ARTICLE INFO

Article history:
Received 29 June 2018
Received in revised form 28 December 2018
Accepted 18 March 2019
Available online 29 March 2019

Keywords:
Extreme multi-label text classification
Semantic indexing
Deep learning
MeSH
Semi-supervised word embeddings

ABSTRACT

The classification of natural language texts has gained a growing importance in many real world applications due to its significant implications in relation to crucial tasks, such as Information Retrieval, Question Answering, Text Summarization, Natural Language Understanding. In this paper we present an analysis of a Deep Learning architecture devoted to text classification, considering the extreme multi-class and multi-label text classification problem, when a hierarchical label set is defined. The paper presents a methodology named Hierarchical Label Set Expansion (HLSE), used to regularize the data labels, and an analysis of the impact of different Word Embedding (WE) models that explicitly incorporate grammatical and syntactic features. We evaluate the aforementioned methodologies on the PubMed scientific articles collection, where a multi-class and multi-label text classification problem is defined with the Medical Subject Headings (MeSH) label set, a hierarchical set of 27,775 classes. The experimental assessment proves the usefulness of the proposed HLSE methodology and also provides some interesting results relating to the impact of different uses and combinations of WE models as input to the neural network in this kind of application.

1. Introduction

The classification of natural language texts is a key aspect in many tasks and in different domains. This kind of classification problem consists in applying one or more labels to each

[☆] This paper is an extended and improved version of the paper: Francesco Gargiulo, Stefano Silvestri and Mario Ciampi, Deep Convolution Neural Network for Extreme Multi-label Text Classification, presented at the AI4Health 2018 workshop and published in: BIOSTEC 2018, Proceedings of the 11th International Joint Conference on Biomedical Engineering Systems and Technologies, Volume 5: HEALTHINF, Funchal, Madeira, Portugal, 19-21 January, 2018, pp. 641-650, ISBN: 978-989-758-281-3, INSTICC, 2018
[*] Corresponding author.

E-mail address: francesco.gargiulo@icar.cnr.it (F. Gargiulo).

https://doi.org/10.1016/j.asoc.2019.03.041
1568-4946/© 2019 Elsevier B.V. All rights reserved.

document of a text collection. In literature this task has been approached by means of several different techniques, ranging from ontology-based methods to Machine Learning (ML) systems, or through the adoption of hybrid approaches integrating ontological knowledge and ML [1,2]. The increase of computational power and the availability of huge amounts of data, along with the active research and developments in the field of Deep Neural Networks (DNN), have recently led to the definition of DNN models able to outperform the previous state of the art systems.

In accordance with the literature, it is possible to identify a taxonomy of Natural Language text classification problems, composed of the following four classes:

- **Binary Classification**, where the labels belong to a binary set (Positive and Negative, True and False, etc.);
- **Multi-Class Classification**, where the single classification label belongs to a set with more than two elements;
- **Multi-Label Classification**, when the labels belong to a multi-class domain, but differently from the previous case, each document can be tagged with a variable number of labels, ranging from one to a total class number; and
- **Extreme Multi-Label Text Classification (XMTC)** [3] refers to the automatic assignment of the most relevant subset of labels to a text document, but differently from the classic multi-label problem where the label set size is usually in the order of ten, in this case the labels belong to an extremely large set, in the order of thousands, or ten of thousands of elements. If the label set is hierarchically organized, a hierarchical XMTC problem is defined.

The huge XMTC label space raises many research challenges, such as data sparsity and scalability. The availability of Big Data and the application of XMTC to real world problems have attracted a growing attention of researchers from ML and Deep Learning (DL) fields. Significant advances in multi-label classification methodologies have been made in recent years, thanks to the development of specific ML methods, although DL methods have not yet been widely explored on account of this particular problem.

In this paper we analyze a DL approach based on a Convolutional Neural Network (CNN), devoted to the hierarchical XMTC problem. We define a methodology that expands the label set of each document integrating all the missing labels along the label hierarchy. This operation is necessary because usually only the leaves of the tree and a few labels along the hierarchy are considered for indexing purposes by human experts who manually label the documents. The lack of all the classes along the hierarchy can lead to an incorrect training of the DNN, due to label inconsistencies.

We also analyze the impact of the use and combination of different types of embeddings for the representation of the input training text. In more detail, we evaluate the impact of semi-supervised embedding models. These latter models are able to explicitly infer grammatical and syntactic information in the obtained word vectors and can provide a performance boost in other tasks, such as Word Analogy/Similarity Querying, Named Entity Recognition (NER), Relation Extraction and Sentence Classification [4–7].

All the results have been evaluated using the PubMed[1] scientific articles collection as a test case. PubMed is a search engine maintained by the US National Library of Medicine (NLM), specifically devoted to medical and biological scientific papers. We have considered only the text of the title and the abstract of each paper, along with the corresponding labels, due to their free availability in PubMed. Each paper has been manually tagged by domain experts with a variable number of classes from the (MeSH) set, a hierarchical label set characterized by a total number of labels equal to 27,775. For these reasons, the automatic classification of PubMed papers with MeSH belongs to the hierarchical XMTC case.

The automatic classification of PubMed papers is also a task required by the NLM in order to help the domain experts in their tedious and time-consuming work. To achieve this objective, the NLM supports BioASQ,[2] a distributed challenge for the research community; one of the aims of BioASQ is the advance of the state of the art systems devoted to the automatic application of MeSH to PubMed indexed articles [8].

The XMTC problem is involved in many real world applications, such as the one above described, confirming the utility of any efforts focused on searching for new solutions. The results of our experiments prove the usefulness of the proposed HLSE method and provide many interesting findings resulting from the analysis of the different performances of the neural network in relation to the embedding models used. This analysis could also be considered as a starting point for an emerging problem, which may be addressed by what is called explainable-AI [9,10], namely that of correlating the input data representation and the label structure with their impact on the DL model performance.

The paper is organized as follows: in the next section an overview of the current state of the art is presented; then, all the details of the DNN used and the proposed methodologies are explained, followed by the experimental results, where the dataset details, a description of the evaluation measures and the instruments used to implement the whole architecture are also included; and finally, the obtained results are discussed and analyzed, in comparison with the state of the art.

2. Related works

The text classification problem has been addressed in the literature with many different approaches [1]. Some of them are based on ML methods with manual feature engineering, such as Latent Dirichlet Allocation (LDA) or K-Nearest Neighborhood (K-NN) [11,12]. More recently, various DNN approaches have been proposed, obtaining very promising results. In [13] a simple Neural Network (NN) approach for large-scale multi-label text classification has been presented, evidencing the usefulness of the cross entropy error function. The author herein demonstrated that DL can obtain an optimal performance in this task and proved that simple NN models equipped with advanced techniques such as Rectified Linear Units, Dropout, and AdaGrad outperform not-NN approaches on six large-scale textual datasets with different characteristics. Another DL solution to multi-label text classification issues has been proposed in [14]. In this case, a CNN has been applied to sentences of NL clinical texts, in order to obtain sentence level classification, the results obtained that this approach outperform the WE based methods. In [15] the performances of discriminative and generative Long Short Term Memory (LSTM) models for text classification have been evaluated, revealing that generative models substantially outperform discriminative models. The authors of [16] have explored the use of Very Deep Convolutional Neural Networks (VDCNN) in multi-class text classification, proving the effectiveness of their methodology with a large scale training set.

In [17] an innovative Recurrent Neural Network (RNN) is described. The presented architecture is a multi-label ranking model

1 https://www.ncbi.nlm.nih.gov/pubmed/.

2 http://bioasq.org/.

based on a LSTM NN, consisting of two LSTMs used respectively for an adaptive data representation process and a unified learning-ranking process. The first LSTM is used to learn document representation by incorporating the document labels, while in the latter ranking LSTM the order of the document labels is rearranged in accordance with a semantic tree, in which the semantics is compatible with and appropriate to the sequential learning of the LSTM. Connectionist Temporal Classification is performed in the ranking LSTM to address the error propagation for a variable number of labels in each document. Experiments with document classification conducted on three typical datasets revealed an impressive performance. In [18] the use of a Residual Network (ResNet) to improve the performance of a LSTM RNN in a multi-label text classification task is analyzed. The authors showed that the direct adaptation of ResNet performs well in sequence classification and that, when combined with the gating mechanism in LSTM, significantly improves residual learning and the LSTM performance. In [19] a model based on both CNN and RNN has been successfully applied to multi-label text classification. The proposed approach was able to capture both the global and the local textual semantics and to model high-order label correlations, having at the same time a tractable computational complexity. The experimental assessment showed that it achieves the state of the art performance when the CNN-RNN model is trained using a large size dataset.

The XMTC task (Section 1) is certainly challenging, due to the very large label set involved, with the result that currently there are still open issues. In [3] a DL approach for XMTC is described, presenting a family of new CNN models tailored for this specific application. The proposed models have been tested with different datasets, producing very good results in all cases. In [20] a comparison between ML and DL approaches for XMTC has been presented, with various approaches tested in the assignment of ICD-9 codes to Electronic Health Records. Support Vector Machines, Continuous Bag of Words, CNNs and GRUs have been considered, the results demonstrating that the latter two DL models provide the best results. The ICD-9 labeling of textual medical notes, has also been studied also in [21], the authors finding that a RNN and a RNN with LSTM units show an improvement over the Binary Relevance Logistic Regression model.

In many cases the label set of the XMTC problem is hierarchically organized. The authors of [22] have proposed a graph-CNN based DL model for this particular case. The method first converts texts to graph-of-words, and then uses graph convolution operations to convolve the word graph. This graph representation has the advantage of capturing non-consecutive and long-distance semantics, while the CNN is able to learn different levels of semantics. The authors also regularized the DNN architecture with the dependency among labels, in order to leverage the hierarchy of labels. The results on large scale datasets proved that this method can significantly improve performance, if compared with traditional hierarchical text classification approaches and previously developed DNNs.

The labeling of the PubMed scientific articles collection is a real world hierarchical XMTC application, very often considered in the literature. Some of the state of the art methods devoted to PubMed papers classification have been developed at the BioASQ distributed challenges (see Section 1), the latest results of which are described in [8].[3] The baseline architecture for MeSH labeling in the latest BioASQ challenge is the NLM Medical Text Indexer (MTI) [23] and its evolution is described in [24]. The MTI system uses two different methods to identify potential labels: the MetaMap algorithm [25] and PubMed Related Citations (PRC) [26]. The MetaMap algorithm maps the word phrases of a text into MeSH terms. In order to include all possible variants of the phrases, the algorithm generates for each noun phrase its spelling variants, abbreviations, synonyms, acronyms and inflectional and derivational variants, and all meaningful combination of these, using knowledge bases and NLP tools; then it searches among MeSH for all the possible candidate results, estimating the quality of each mapping using an evaluation function and selecting the results with the highest score. The PRC method uses a probabilistic topic-based model based on KNN to identify citations and to calculate a content similarity index. The MTI combines and ranks the lists of labels obtained with both methods, includes recommendations based on various lookup lists and filters the resulting MeSH classes according to the NLM indexing rules. The authors of [24] have enhanced this MTI system, by first extracting various specific features (PRC based features, Text based features, Vocabulary Density based features, MTI based features, Journal Descriptor Indexing features and MeSH Similarity based features) from the MTI results, and then training a Learning to Rank (L2R) model, in order to improve the initial MTI results.

The best performance in the latest BioASQ shared task has been obtained by the *DeepMeSH* system, described in [27] and in [28]. DeepMeSH is based on a new deep semantic representation named D2V-TFIDF, which concatenates both the sparse Term Frequency-Inverse Document Frequency (TF-IDF) and dense (doc2vec D2V [29]) semantic representations of words and documents. This vector representation is used as the input for two different pipelines. In the first, a set of binary classifiers is trained to predict a single MeSH; then a regression model is trained to predict the number of MeSHs, obtaining in this way a set of ranked MeSHs on the base of the relevance scores predicted by the binary classifiers. The second pipeline uses a K-NN to select the k-nearest indexed neighbors, based on the cosine similarity of their feature vectors; then a Support Vector Regression (SVR) is used to obtain the MeSH number. The results of both methods are combined to obtain the final result. Another system able to outperform the baselines approaches is the *AUTH* system [8,30]. In this case, the authors have used a vector representation similar to one previously described, based on TF-IDF and D2V. This feature representation is sent as input to a set of Binary Relevance models based on Support Vector Machines (SVMs) and to a variant of the LDA model. All the models were combined in an ensemble using a statistical significance multi-label ensemble to perform the classifier selection. Interesting performances in the latest BioASQ challenge have also been obtained by *MULE* [31], where the authors have used an ensemble of machine learning algorithms (SVM, SVR and Labeled LDA), achieving good results. Another approach to address the BioASQ task similar to those previously described has been adopted by means of *MZ* system [8], where Binary Relevance classification and LDA models with label frequencies per journal as prior frequencies is used, exploiting the features of TF-IDF; in the end, a regression for threshold prediction has been adopted. In [32] the *MeSH Now* system has been described, where a novel learning-to-rank framework is presented. The method retrieves the first 20 k-NN articles for each new PubMed, exploiting the assumption that documents similar in content would share a similar MeSH labeling. Then, a binary classifier based on a SVM is built to classify the most frequently indexed MeSHs, not considering the rarest cases. The results are combined with those obtained by the MTI baseline system and a Learning to Rank algorithm is applied in order to rank them all. Finally, the results are improved with some post-processing steps. Despite the optimal performances obtained in the BioASQ task by all the above described methods, they all exploit a large number of ML models, that have drawback of requiring a manual feature engineering phase; in addition, many different ML classifiers and models need to be trained and evaluated.

[3] The detailed results of are available at http://participants-area.bioasq.org/results/5a/.

Table 1
Systems addressing the BioASQ MeSH XMTC task on the PubMed dataset and the respective approaches used.

System	Approach
Default MTI [23]	UMLS MetaMap, K-NN
MTI First Line Index [24]	Learning to Rank, MTI
Search System [8]	Search Engine, UIMA ConceptMapper
MZ [8]	TF-IDF, LDA, BR classification
Sequencer [8]	Word Embeddings, RNN
MeSH Labeler [33]	LogReg, KNN, SVM, MH, pattern matching, MTI, correlation
Deep Mesh [27], [28]	d2v, TF-IDF, MESH Labeler
AUTH [30], [8]	d2v, TF-IDF, LLDA, SVM, ensembles
Iria [34], [8]	bigrams, Luchene Index, k-NN, ensembles, UIMA Concept Mapper
MeSH Now [32]	K-NN, SVM, MTI, Learning to Rank, Post-Processing
MULE [31]	Unibram, Bigram, TF-IDF, SVM, SVR, LDA ensembles
STML [35]	Word Embeddings, bi-GRU

Other approaches to solve the BioASQ task are rule-based, such as the one implemented by *Search system* [8]. The authors have exploited UIMA ConceptMapper [36] and defined a UIMA-based text and data mining pipeline. The *IRIA* system [8,34] has adopted a hybrid approach, where the rules are used at the end of a ML-based methodology. In detail, a textual representation of the articles to be stored in an Apache Lucene textual index is created, obtaining indexed representations that are queried by using the contents of the article to be annotated. The similarity-based descriptor selection is obtained with a *k*-NN algorithm. Pointwise Mutual Information (PMI) scores have been incorporated, in order to capture the most relevant multiword terms through a voting ensemble scheme; finally, a UIMA ConceptMapper annotator matches subject headings with the abstract text of the corresponding citation. The rule based approaches suffer form a lack of generalization, relying on fixed rules.

Additionally, some DL-based approaches have been adopted to meet the BioASQ challenge. The *Sequencer* system [8] is based on a RNN that takes as input WEs trained on PubMed articles. The *STML* [35] system also addresses the BioASQ task, proposing a DNN architecture formed by a WE layer, a bidirectional Gated Recurrent Unit (GRU) layer and a fully connected layer. This architecture is able to provide an improvement in the precision of the results if compared with the state of art approaches. The DL-based methods have the advantages of not requiring any manual engineering of the characteristics and of being based on a single DNN architecture, requiring in this way a less complex training phase.

The following Table 1 summarizes the above described systems specifically devoted to BioASQ MeSH XMTC, highlighting the corresponding methodologies used. The details of the results will be shown, discussed and compared with our proposed methodology in the subsequent Section 4.

3. Methodology

In this Section we first provide a brief overview of the DNN architecture that, for the sake of clarity, we have divided into three main modules: an *Embeddings module* for text encoding, a *Feature Extraction module* implemented through CNNs and a *Classification module* composed of fully connected neural networks. We also highlight the details of the loss function and the hyper-parameters used to train the network. Next, we describe the details of the proposed Hierarchical Label Set Expansion (HLSE) method, used to regularize the training set labels. Finally, we show the various word embedding models considered for the representation of the input text, describing how each of them has been obtained and combined.

3.1. DNN architecture

We have adopted a simplified version of the DNN architecture proposed in [16] based on a CNN chain, given our main aim, that of testing and analyzing the effects of the HLSE algorithm and different combinations of embedding models.

The proposed DNN for XMTC is composed of the following three main modules:

- Embedding module;
- Feature extraction module;
- Classification module.

The Embedding module produces a suitable representation of the input text, converting each word into a numeric vector. Various methodologies have been proposed in literature able to obtain Word Embeddings (WEs) from a large natural language text corpus with unsupervised training, starting from the WE the baseline presented in [37]. Recently, some authors proposed new embeddings models based on a semi-supervised training, exploiting an annotation of the training corpus with grammatical and syntactic features. In order to investigate on the impact of different WE models on the XMTC task, we have trained different models and combined them as described in the next Section 3.3.

The second module is devoted to feature extraction and is implemented with a Convolutional Neural Network (CNN). In detail, it is composed of two CNNs. In the first, a 1D-Convolution with a kernel size equal to 5, 200 filters and a *ReLU* activation function is followed by a Max Pooling (MP) layer whose pool size is 1. The second CNN has a 1D-Convolution with a kernel size equal to 3, 100 filters and a *ReLU* activation function, followed by a Max Pooling layer with a pool size equal to 2. A Dropout of 0.1 is applied on the output in order to prevent overfitting during the training phase. The inputs of the CNN are the WEs vectors corresponding to each word of the title and abstract of each paper from the PubMed dataset. The convolution input needs a fixed size, but the total word number of each sample is not constant. To overcome this problem, a maximum number of words has been chosen equals to 400, considering an average word number of the dataset equal to 127 after the NLP pre-processing for text normalization and equal to 167 without applying the normalization NLP pipeline (see Section 3.3, Table 2 in Section 4.1 for details). In the case of a sample with a lower word number, a zero padding is applied, while in the case of samples with more than 400 words, last words of the text are discarded.

The classification module, whose task is to associate one or more labels (classes) to each input text, is composed of a cascade of two different fully connected dense layers: the first has 1024 outputs, while the second and final layer of the whole DNN has an output number equal to the number of the graph nodes that represent the hierarchical label set L (see Section 4.1).

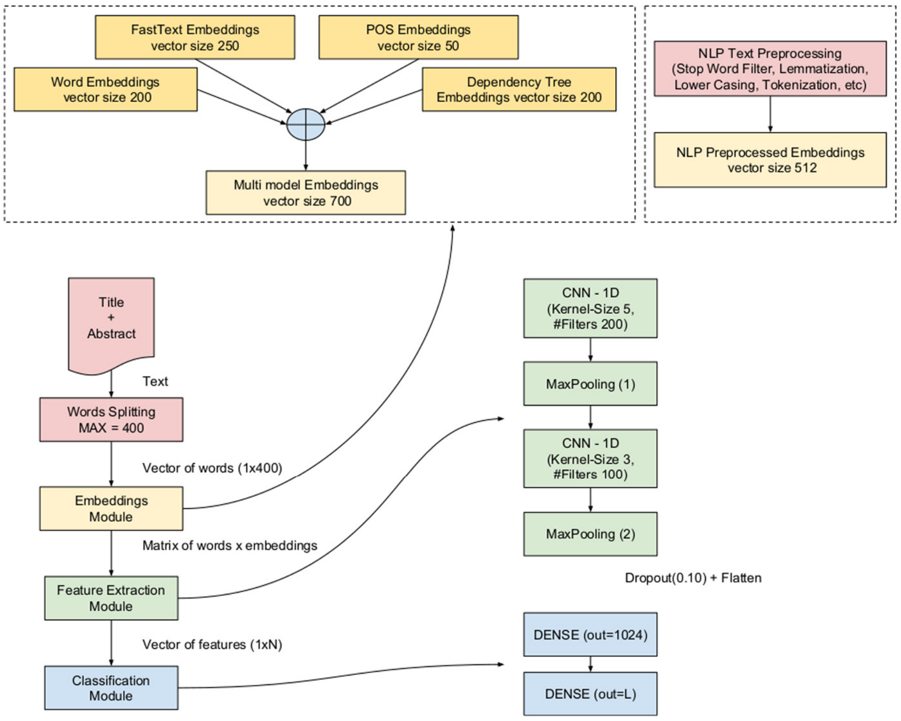

FastText Embeddings vector size 250
POS Embeddings vector size 50
Word Embeddings vector size 200
Dependency Tree Embeddings vector size 200
Multi model Embeddings vector size 700
NLP Text Preprocessing (Stop Word Filter, Lemmatization, Lower Casing, Tokenization, etc)
NLP Preprocessed Embeddings vector size 512
Title + Abstract
Text
Words Splitting MAX = 400
Vector of words (1x400)
Embeddings Module
Matrix of words x embeddings
Feature Extraction Module
Vector of features (1xN)
Classification Module
CNN - 1D (Kernel-Size 5, #Filters 200)
MaxPooling (1)
CNN - 1D (Kernel-Size 3, #Filters 100)
MaxPooling (2)
Dropout(0.10) + Flatten
DENSE (out=1024)
DENSE (out=L)

Fig. 1. Graphical representation of the DNN models used. The yellow blocks represent the WE module, the green blocks are the *feature extraction* layers and the blue blocks represent the classification layers. The pink blocks are related to the pre-processing phases.

Table 2
Overall statistics of the dataset used, with and without considering the NLP preprocessing. N is the total document number, $Avg(words)$ and $Std(words)$ are respectively the average and the standard deviation of the number of words per document (title and abstract) and $\#classes$ is the total number of classes.

Dataset	N	$Avg(words)$	$Std(words)$	$\#classes$
Non-NLP Preprocessed	11, 075, 577	209	78	27,755
NLP Preprocessed	11, 075, 577	127	48	27,755

Fig. 1 shows the architecture and the details of the corresponding NN layers of these modules.

We have analyzed the contribution of different embedding models, also considering semi-supervised techniques, as described in the next Section 3.3, in order to understand which input model better performs when applied in the XMTC task.

3.1.1. Loss function and SGD hyperparameters

The loss function used during the training phase of the DNN is the sigmoid cross entropy (Eq. (1)). In this case the loss computed for each vector component is not affected by the other component values and thus it is used for the multi-label classification [13]. The sigmoid cross entropy function is equal to:

$$loss(x,y) = -\sum_{l\in L}\left[\left(y_l \cdot \log\frac{1}{1+\exp(-x_l)}\right) + \left((1-y_l)\cdot\log\frac{\exp(-x_l)}{1+\exp(-x_l)}\right)\right] \quad (1)$$

where y_l and x_l are respectively the prediction and the target for each label $l \in L$.

The correct setting of the hyperparameters of the DNN is another crucial aspect [38]. Some of the hyperparameters are directly related to the network structure and topology, such as the number of hidden layers, the number of hidden units and the choice of activation function, while others influence the training phase, because they are directly involved in the Stochastic Gradient Descent (*SGD*). The *SGD* algorithm updates the parameters θ of the objective function $J(\theta)$, following Eq. (2):

$$\theta = \theta - l_r \nabla_\theta J(\theta, x_i, y_i) \quad (2)$$

where x_i, y_i is a sample/label pair from the training set and l_r is the learning rate. The *SGD* is noisy, due to the update frequency of the weights performed at each sample. In order to reduce this effect and to obtain a better functional point, the parameter update in the *SGD* is usually computed with a variation of this method called *Mini-batch gradient descendent*, where the cost function is computed starting from a batch of samples:

$$\theta = \theta - l_r \nabla_\theta J(\theta, x_{(i:i+bs)}, y_{(i:i+bs)}) \quad (3)$$

where *bs* is the batch size. This approach provides many advantages: first, it smooths out some of the noise in the *SGD*, although it still allows the stuck of the function caused by local minimums; secondly, the mini-batch size is still small, thereby keeping the performance benefits of *SGD*. Another problem of the *SGD* is related to local optima. *SGD* cannot easily solve ravines, namely the areas where the function surface curves much more steeply in one dimension than in another [39], causing oscillations and a very slow convergence. Momentum [40] can be used to accelerate the function along the shallow ravine . The Momentum *SGD* update (4) is equal to:

$$\begin{aligned} v_t &= \mu v_{t-1} + l_r \nabla_\theta J(\theta, x_i, y_i) \\ \theta &= \theta - v_t \end{aligned} \quad (4)$$

where v_t is the *velocity* at iteration t and μ is the momentum coefficient. In this way, the SGD can gain a faster convergence and reduce its oscillations. A further improvement to the SGD momentum has been made by the Nesterov accelerated gradient [41], which is equal to:

$$\begin{aligned} v_t &= \mu v_{t-1} + l_r \nabla_\theta J(\theta - \mu v_{t-1}, x_i, y_i) \\ \theta &= \theta - v_t \end{aligned} \quad (5)$$

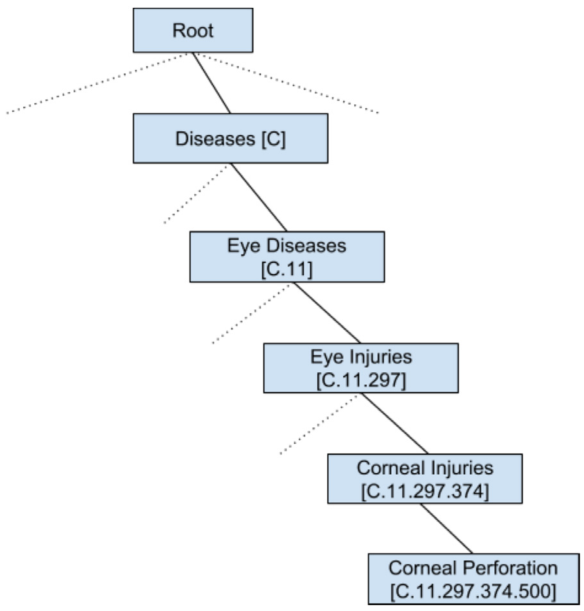

Root
Diseases [C]
Eye Diseases [C.11]
Eye Injuries [C.11.297]
Corneal Injuries [C.11.297.374]
Corneal Perforation [C.11.297.374.500]

Fig. 2. A part of the MeSH tree label hierarchy, showing the full path from the root to the leaf *Corneal Perforation*.

Table 3
Dataset statistics for the Original MeSH graph and the Expanded MeSH graph. L represents the total number of MeSH graph nodes number, L^* is the average number of labels per document and L^o is the average number of documents per label.

MeSH graph	L	L^*	L^o
Original MeSH graph	57,842	12.91	5, 188.15
Expanded MeSH graph	57,859	50.57	19, 614.76

The term $\theta - \mu v_{t-1}$ in Eq. (5) approximates the next parameters update (the gradient calculation is missing, but in the case of small differences this approximation is good). Thus, the gradient is calculated not on the current parameters θ, as in the previous Eq. (4), but on an approximation of their future position. The effects of this modification result in an increased responsiveness, because the velocity is corrected in order to take into account the next position. In this way, the momentum increases or decreases adjusting itself to the future variations, improving the convergence of the *SGD*.

The experiment previously performed on the same DNN architecture described in [42] produced, as a result, an optimal hyperparameters set, also used in this case. In detail, the batch size *bs* has been set equal to 10, the momentum is 0.5 and the learning rate l_r is 0.1.

3.2. Hierarchical label set expansion

When human domain experts perform the task of labeling the documents of a text collection with classes belonging to a hierarchy, it often happens that they do not consider the full path of the label hierarchy. In other words, when a label is manually assigned to a document, all its ancestors in the corresponding class tree should also be assigned, in order to produce a consistent dataset for the training of Machine Learning (ML) systems. A document tagged with a more compact set of labels, which does not includes all ancestor classes, can better summarize and focus the main topics of a document and is more suitable for indexing purposes. On the other hand, when this kind of documents are used as a training set for a ML automatic classification system, the label inconsistencies with respect to the class hierarchy could lead to a wrong behavior of the classifier, especially when the hierarchy itself must be learned.

The PubMed collection is an example of such documents: each label manually assigned by an expert belongs to the MeSH hierarchical set. However, very frequently the documents are not labeled with all the classes laying the hierarchy path formed by all corresponding ancestors. The following example can better clarify this problem. In Fig. 2 a part of the MeSH hierarchy tree is represented,[4] showing the full path from the root to the leaf *Corneal Perforation*, along with class names and MeSH codes (the code of the leaf is [C11.297.374.500]).

If one of the topics of a document is *Corneal Perforation*, a domain expert may tag it with, as a minimum, the class coded as ([C11.297.374.500]) and its second ancestor, *Eye Injuries* ([C11.297]), without including the other classes in the hierarchy path, namely the first ancestor and the other classes along the path of the graph in Fig. 2. Another document of the collection, whose topics are very similar to the one previously considered, could be tagged with only *Corneal Perforation* ([C11.297.374.500]), while another may have a larger the label set, formed by *Eye Diseases* ([C.11]), *Corneal Injuries* ([C.11.297.374]) and *Corneal Perforation* ([C11.297.374.500]). As explained above, the main purpose of MeSH labeling is document indexing, and to this end the exclusion of some labels is not wrong, but, due to the high variability of the labels of documents with a very similar topic, the missing classes make the training of a ML classifier very difficult and causes a very noisy training set.

With the objective of improving the performances of the proposed DNN architecture, we defined a methodology for the expansion of the labels of each sample document of the training set, aiming at including the complete hierarchy path for each class, naming this method as *Hierarchical Label Set Expansion* (HLSE). In this way, it is possible to overcome the previously described problems that led to a noisy training set. In detail, for each sample document a new label set is created, adding all the ancestors of the original labels, if not present in the original set. For example, if in a document label set there are the MeSHs *Corneal Perforation* ([C.11.297.374.500]) and *Eye Injuries* ([C.11.287]), following the hierarchy path depicted in Fig. 2, the following MeSHs will be added to the set: *Corneal Injuries* ([C.11.297.374]), *Eye Diseases* ([C.11]) and *Diseases* [(C)], completing in this way the whole label path from the root to the leaf class.

The above described task can be easily accomplished by finding all labels of the ancestor nodes of the original classes, using classic tree traversal algorithms. In this particular case, this process can also be performed exploiting the MeSH code structure, which is directly related to the hierarchical tree. In fact, looking at the codes of the classes in Fig. 2, it is possible to observe that when the level of the tree is increased, the MeSH code adds a new group of numbers, separated by a dot. In this way, it is possible to obtain all the ancestors of a label, iteratively removing from the code of the leaf each group of numbers separated by dot, until the root class is reached.

As shown in the next Section 4, we have proved that the expansion of the label set provides a significant performance boost to the classification results. The only drawback is the increase in the average number of labels per document (as shown in the next Table 3), making the classification problem more complex. Despite the larger label set for each document, the contribution made by HLSE is indeed considerable, as proved by the experimental results.

[4] The full MeSH hierarchy tree can be browsed at https://meshb.nlm.nih.gov/treeView.

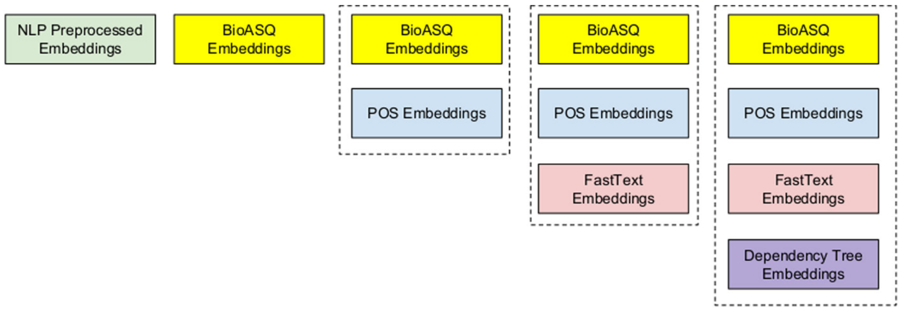


Fig. 3. Different embeddings models used as pre-trained input, obtained by concatenating the corresponding the word vectors from each single model.

3.3. Embedding models

Different methods to train the input word embeddings have been analyzed, in order to evaluate their effectiveness, considering them both alone or in combination with each other. We first considered a WE model trained on a normalized corpus, obtained through a NLP pipeline [43,44] and we compared it with a model trained on a non-normalized training set. Then, we considered some WE algorithms that modified the base word2vec [45] by adding some features, such as sub-word information [46], or explicit grammatical and syntactic features [4–7]. Those latter WE models made an impact on the performances of some DNNs devoted to specific tasks, such as Word Analogy/Similarity Querying, Named Entity Recognition (NER), Relation Extraction and Sentence Classification. Our aim was to verify the effectiveness of specific WE models in the considered XMTC task, combining different models, as depicted in Fig. 3, exploiting the compositional property of WE [37]. To achieve this objective we trained the following embedding models:

- **NLP preprocessed embedding model**, obtained with the application of a NLP pipeline to normalize the input training text.
- **BioASQ embedding model**,[5] that uses the WE model provided for the BioASQ distributed task. In this case the input training text has not been normalized with NLP techniques.
- **BioASQ + POS embedding model**, where the vectors of the previous model have been concatenated with POS embeddings [47].
- **BioASQ + POS + fastText embedding models**, obtained by concatenating a fastText model [46] to the previous model.
- **BioASQ + POS + fastText + Dependency Tree embedding models**, incorporating also the vectors from the Dependency Tree embedding models [6].

In the following paragraphs of this Section we describe the details of the aforementioned models, while in Section 4.2 we show all the information about their implementation and training phases.

Word Embeddings (WE) [37,45] is a shallow neural network that maps the input text into a continuous vector space, transforming each word of a training set into a fixed size vector to be used as input to a more complex DNN. The correct choice of the training corpus for the WE model is very important; it has also been shown that a closed domain document collection, such as PubMed, which is focused only on the biomedical field, can lead to optimal performances, if the obtained WE model is applied to the same domain [43]. Thus, we extracted the text of the titles and abstracts of the entire collection of papers indexed in PubMed as training corpus for WEs.

Another method to improve the WE model is the normalization of the training corpus by means of a NLP pipeline [44,48]. This operation aims at simplifying the NL text complexity. For this purpose we applied the following NLP operations to the text extracted from PubMed as training set. All the words from the original text were tokenized and lower cased and all punctuation, symbols and numbers were removed; a stop word filtering was applied, with the purpose of reducing the data sparsity caused by adverbs, articles, conjunctions, and other common stop words; finally, all the words were lemmatized. The resulting preprocessed text was used to train the WE model with the skip-gram algorithm and hierarchical softmax [37]. In order to assess the utility of this NLP preprocessed embedding model, we also tested the BioASQ embedding model [49], a vector space provided by BioASQ, trained on the abstracts of the PubMed papers with a very soft preprocessing applied (only tokenization, punctuation and symbol removal and lower casing).

An intrinsic limitation of the WE is the inability to explicitly encode any grammatical or syntactical information into the resulting word vectors. An aspect of grammatical information that WE is not able to fully capture is the Part Of Speech (POS). POS represents the category of a word, namely if that word is a noun, adjective, article, adverb, etc. In languages such as English, where the same word can have different POS depending on the context in which it is used, it could be useful to provide this explicit information to the training set of the embeddings. In [50] the authors proposed a model able to learn word embeddings with weighted contexts based on POS relevance weights, while in [47] they produced a POS embeddings model trained on the POS tag sequence obtained by substituting the POS tag for each word of the training set. We adopted the latter method, substituting the POS tag for each word in the original training text and then applying the word2vec algorithm to the sequences of POS obtained. Finally, we built a WE model concatenating the BioASQ model with the obtained POS embedding model.

Another considered WE model considered is fastText [46,51,52]. The main purpose of fastText is to add character n-grams information to the original WE model, training word vectors on both the word sequences and character sequences of each word, assuming a word to be formed by n-grams of character, providing in this way a different representation from word2vec including sub-words information. We trained a model using fastText and we concatenated the obtained word vectors with the BioASQ model and POS embedding model, creating a further WE model.

The syntactic dependencies of words in the sentences of the training text can also be used as additional information for the training of an embedding model, incorporating into the training corpus the Dependency Tree corresponding to the input sentences, as described in [53] and [6]. We used in our experiments the method from [6], implemented in word2vecf software [53],[6]

5 http://bioasq.lip6.fr/tools/BioASQword2vec/.

6 https://bitbucket.org/yoavgo/word2vecf/overview.

in order to obtain a Dependency Tree embedding model that we concatenated with the BioASQ, fastText and POS embedding models, obtaining the last WE model that we tested.

4. Experimental results

In this Section we first describe all the characteristics of the datasets used for the experimental assessment. Next, we report the details of the systems used for the implementation of the proposed architecture are described, listing and explaining all the corresponding parameter settings. We also describe the methods used to obtain the unsupervised and semi-supervised embedding models described in the previous Section 3. Next, we provide a complete overview of the evaluation metrics for the flat and hierarchical multi-label text classification used in the experimental assessment. A comparison of the performance of our approach with the state of the art methodologies devoted to MeSH labeling is also made. Finally, the results obtained are presented, analyzed, and discussed.

4.1. Dataset description

The experimental assessment has been performed on the PubMed paper collection.[7] PubMed is a free search engine maintained by the US NLM, specifically devoted to medical and biological scientific articles. The papers indexed by PubMed are manually labeled by domain experts, mainly for cataloging purpose. The label set used in PubMed is named *MeSH*. The release from year 2017, used in our case, is composed of 27,755 different classes hierarchically organized as a graph[8] with MeSH being located in one or more nodes. Each document from the dataset is labeled with a variable number of classes.

Our experimental dataset is composed of documents formed by only the text of the title and the abstract of each PubMed article, along with the corresponding labels, due to their free availability. It is worth noting that hereinafter we will denote as *document* the text obtained as above described. The text extraction from the PubMed repository has been performed with the Big Data architecture described in [54], obtaining a dataset composed of $11,075,577$ documents.

Table 2 summarizes the main features of the document collection above described used as the experimental dataset. The original dataset has an average word number per document equal to 209, with a standard deviation of 78, while the same dataset after the NLP preprocessing described in Section 3 has an average word number per document equal to 127, with a standard deviation of 48. Due to the high number of classes, the classification problem related to the automatic assignment of the correct MeSH set to each document of PubMed belongs to both multi-class and multi-label types. Due to the very high class number can be classified as XMTC, as described in Section 1.

The dataset obtained has been split into a training set and a test set, with 99% of the samples randomly selected for the training set and the remaining 1% for the test set. Following [55], the most correct way to compare multiple classification systems over multiple datasets is based on the average ranking obtained across all datasets. For this purpose, we divided the test set into ten smaller sets, each one composed of approximately 2500 samples.

Table 3 shows additional important features of the dataset: the total number of MeSH graph nodes, the average number of labels per document and the average number of documents per label, for both the original training dataset and the training set after the application of the proposed HLSE method described in Section 3.2.

As explained above, the total MeSH set includes 27,755 different classes, structured as a hierarchical graph. The same class could fall in several nodes of the graph, making the graph node number higher than the total class number. For example, the class *Retinoblastoma* is represented in the node coded as ([C11.768.717.760]), lying in the path of the graph that starts from the node *Eye Diseases* ([C11]); the same class is also present in the node coded as ([C04.557.465.625.600.725]), along the path from the parent node *Neoplasms* ([C04]). For this reason, the total graph node number is equal to 57,842. In addition, when the label set of each sample document is expanded following the HLSE methodology, some nodes are added to the graph, because the original node set misses some levels. In this way a graph formed of 57,859 nodes is obtained. Table 3 also shows that the classification problem becomes more difficult when HLSE is applied, due to the higher average number of labels per document, L^*. On the other hand, the training set after the HLSE increases the average number of documents per label, L^0, providing several samples of each class during the training phase of the DNN.

As explained above, the experimental results obtained, described in the following paragraphs of this Section, can add further details to the latest BioASQ results [8], helping the research community to solve this difficult XMTC problem.

4.2. Implementation and parameter settings

The whole DNN architecture described in Section 3 and depicted in Fig. 1 has been implemented using Keras,[9] a Python Deep Learning framework. The embedding models used in the first module of the architecture have been pre-trained using specific additional software, as described in the following paragraph of this Section, and then loaded into the Keras environment. Due to the average number of words per document, equal to 209 for the original dataset or to 127 in the case of the NLP preprocessed text, with corresponding standard deviations equal to 78 and 48 respectively (see Table 2), we chose to set the input document word number to 400, obtaining a matrix of 400 x the embedding dimension size for the representation of the text.

The embedding module of the proposed DNN uses different models, as described in Section 3.3 (see also Fig. 3). The first embedding model considered in our experimental assessment was trained on a normalized corpus, obtained using the word2vec [37,45] algorithm; we refer to this model as *NLP Preprocessed*. In detail, the training text was severely pre-processed, applying lower casing, tokenization and lemmatization, and removing punctuation, brackets, symbols, numbers and stop words. The normalized text was obtained using the Stanford NLP software tool [56] for the tokenization and lemmatization, and Python scripts for the removal of the punctuation, symbols, numbers and stop words, using the Onix Text Retrieval Toolkit stop word list.[10] The training was performed using Gensim [57] word2vec implementation, selecting the skip-gram algorithm with the hierarchical softmax method and setting a vector size equal to 512 and a context window equal to 10, following the successful results obtained in our previous experiments [42].

The second embedding model considered was the *BioASQ* pre-trained word2vec model,[11] which was trained on PubMed abstract plain texts applying a light pre-processing removing the punctuation, brackets and symbols and lower casing all words.

7 https://www.ncbi.nlm.nih.gov/pubmed/.

8 The MeSH hierarchical structure can be browsed at https://meshb.nlm.nih.gov/treeView.

9 https://github.com/fchollet/keras.

10 http://www.lextek.com/manuals/onix/stopwords1.html.

11 Available at http://bioasq.lip6.fr/tools/BioASQword2vec/.

Table 4
Training parameters of the embedding models: vector size, vocabulary size, window size and character n-grams.

Embedding models	*Vector size*	*Vocabulary size*	*Window size*	*Char n-grams*
NLP Preprocessed	512	1, 192, 810	10	/
BioASQ	200	1, 701, 632	5	/
fastText	200	1, 192, 810	5	3 to 6
POS	50	45	5	/
Dependency Tree	250	1, 701, 632	5	/

The training was performed with the *continuous bag of words* (cbow) method with negative sampling, using a context window equal to 5 and setting a vector size equal to 200 [49].

The third model was obtained by concatenating the POS embedding (see Section 3.3) with the previous BioASQ embedding, obtaining the final BioASQ + POS embedding model. In order to create the training set for the POS embedding, we added the POS tags for each word of the original corpus, using the NLTK Python library [58]. Next, we trained a WE model using as the training set only the POS tag sequences, obtained by substituting the words of the text with the corresponding POS tag. For this purpose, we used the Gensim framework, adopting the skip-gram methodology and setting a context window size equal to 5 and a vector size equal to 50, due to the shorter POS vocabulary size, composed of the whole POS tag set of the Penn Treebank.[12]

A further embedding model considered was fastText [46,52], which adds sub-word information to the basic WE models, analyzing also n-grams of characters in the training corpus words. In this case, we trained the model on the training set described in the previous Section 4.1, using the skip-gram method with a word context window equal to 5, the character n-grams varying in range from 3 to 6 and a vector size equal to 200. In this case also, we concatenated the model obtained with the previous one, obtaining the *BioASQ + POS + fastText* model.

Finally, we trained a Dependency-based WE model, following the methodology proposed in [6]. To achieve this objective, we first produced the dependency trees of the training text using the malt-parser[13] through the Python NLTK libraries. We selected in malt-parser the pre-trained English language model.[14] provided, based on linear SVM trained on the Wall Street Journal section of the Penn Treebank, converted into dependency trees using the Stanford Parser and producing a CoNNL-U standard file[15] Next, thanks to the word2vecf software,[16] [53] we trained the Dependency-based WE model using the skip-gram methodology and hierarchical softmax, setting the word vector size equal to 250 and the window size equal to 5. We concatenated the latter model with the BioASQ, the POS and fastText obtaining the *BioASQ + POS + fastText + Dependency Tree* model.

Table 4 summarizes each embedding model showing vector size, vocabulary size, window size and character n-grams for each. We tested how these models contribute to the DNN classification performance; the results of the experiments are described and discussed in Section 4.4.

The second module of the architecture used is devoted to feature extraction and it is formed of two convolutional layers, each followed by a max-pooling layer. The first convolutional layer has 200 filters and a kernel size equal to 5 with a stride of 1 and the corresponding max pooling layer has a pool size equal to 1. The second convolutional layer has 100 filters and a kernel size equal to 3 with a single stride and a max pooling layer whose size is equal to 2. In both cases the activation function is a ReLU and the padding option is activated. This group of layers is followed by a dropout unit with a discard probability set of 0.1%, in order to prevent overfitting.

The classification module is composed of two fully connected dense layers: the first has 1024 neurons with a ReLU activation function, while the second has a number of neurons equal to the total number of hierarchical label graph nodes. The activation function of the final layer is a sigmoid, obtaining the multi-label classification as output.

The training hyperparameters have been set following the results of the previous experiments described in [42]. In detail, we chose a batch size *bs* equal to 10, a learning rate equal to 0.1 and a momentum equal to 0.5 and we used the Nesterov accelerated gradient (see Eq. (5)).

The system used for the training was a dual CPU Intel Xeon E5-2630, clocked at 2.2 GHz and with 256 GB of RAM, equipped with a Nvidia Titan X 1080 GPU with 11 GB of dedicated VRAM. The training phase required about two hours for each model.

4.3. Evaluation metrics

In literature, various evaluation measures for methodologies that learn from multi-labeled data have been presented. In the case of *flat measures* [59], where the label hierarchy is not considered, it is possible to use *label-based* methods. These methods decompose the evaluation process into separate evaluations for each label, and then perform an average over all labels. The label-based metrics calculation starts from the Precision P_i, Recall R_i and F1-score F_i for each class c_i, equal to:

$$P_i = \frac{tp_{c_i}}{tp_{c_i} + fp_{c_i}} \tag{6}$$

$$R_i = \frac{tp_{c_i}}{tp_{c_i} + fn_{c_i}} \tag{7}$$

$$F_i = \frac{2 \cdot P_i \cdot R_i}{P_i + R_i} \tag{8}$$

where tp_{c_i}, fp_{c_i} and fn_{c_i} are respectively the true positives, false positives and false negatives for the class c_i. Two different averages can be performed: the micro-average and macro-average. In the first case, it is possible to obtain the micro-Precision *MiP*, micro-Recall *MiR* and micro-F1 *MiF* metrics [60], which are calculated as:

$$MiP = \frac{\sum_{i=1}^{L} tp_{c_i}}{\sum_{i=1}^{L}(tp_{c_i} + fp_{c_i})} \tag{9}$$

$$MiR = \frac{\sum_{i=1}^{L} tp_{c_i}}{\sum_{i=1}^{L}(tp_{c_i} + fn_{c_i})} \tag{10}$$

$$MiF = \frac{2 \cdot MiP \cdot MiR}{MiP + MiR} \tag{11}$$

where L is the total class number. In the case of the macro-averaged measures, the P_i, R_i and F_i are previously calculated for each single class, and then are averaged over the total number of classes L, obtaining the macro-Precision *MaP*, macro-Recall *MaR* and macro-F1 *MaF* [59,60]:

$$MaP = \frac{1}{L}\sum_{i=1}^{L} P_i \tag{12}$$

12 https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html.
13 http://www.maltparser.org/mco/mco.html.
14 http://www.maltparser.org/mco/english_parser/engmalt.html.
15 http://universaldependencies.org/format.html.
16 https://bitbucket.org/yoavgo/word2vecf.

$$MaR = \frac{1}{L}\sum_{i=1}^{L} R_i \tag{13}$$

$$MaF = \frac{1}{L}\sum_{i=1}^{L} F_i \tag{14}$$

The macro-average based methods can produce to very different results if compared with micro-average based metrics. In fact, macro-averaging gives an equal weight to each class, while micro-averaging gives an equal weight to each per-document classification result [61]. The F1 measure does not take into account true negatives and its value is heavily influenced by the number of true positives. This causes the domination of large classes over small classes in micro-averaging. For this reason, micro-averaged results are more effective on the large classes in a test collection, while macro-averaged results are better suited to small classes.

In addition to the Precision, Recall and F1, it is possible to evaluate the average Accuracy *Acc* [62]. In a label-based evaluation the *Acc* does not depend on the micro or macro-average [59] and it is equal to:

$$Acc = \frac{\sum_{i=i}^{M} \frac{t_{p_i}+t_{n_i}}{t_{p_i}+t_{f_i}+f_{p_i}+f_{n_i}}}{M} \tag{15}$$

where t_{n_i} are the number of true negatives for the class c_i.

Another approach to obtain a flat measure in a multi-label classification problem is the *example-based* approach [59]. The example-based metrics evaluates bipartitions calculated on the average differences of the true label and predicted label sets over all examples of the evaluation data set, differently from the label-based approach, where separate evaluations for each class are performed. In detail, having a dataset with M multi-label examples and being $Y_i, i = 1..M$ being the set of true labels for each example and Z_i the set of predicted labels for the ith example, it is possible to define the example-based Precision *EBP*, example-based Recall *EBR* and example-based F1 score *EBF* [59,63] as:

$$EBP = \frac{1}{M}\sum_{i=i}^{M} \frac{|Y_i \cap Z_i|}{|Z_i|} \tag{16}$$

$$EBR = \frac{1}{M}\sum_{i=i}^{M} \frac{|Y_i \cap Z_i|}{|Y_i|} \tag{17}$$

$$EBF = \frac{1}{M}\sum_{i=i}^{M} \frac{|Y_i \cap Z_i|}{|Y_i| + |Z_i|} \tag{18}$$

The *EBP* metrics estimates how many predicted labels for the ith example are correct, while the *EBR* estimates how many assigned labels are retrieved; *EBF* combines both measures for a global evaluation. The example-based measures perform a per instance evaluation and the aggregate value is an average over all instances.

Among all the above described flat measures, *MiF* is the one mainly used in literature because it is able to summarize many aspects of the classification results. As explained above, the other flat measures can highlight different particular behaviors of the system.

When the labels lie in a hierarchical structure, such as in the case analyzed in this paper, *hierarchical measures* have been defined to take into account and evaluate the label hierarchy in the classification results. While the flat measures consider only the prediction of exact concepts, the hierarchical measures consider the full path of the exact concepts in the class hierarchy, measuring in this way whether part of the path has been correctly classified. To better understand the hierarchical measure, let us consider Fig. 4, where a simple example of a hierarchical class tree is depicted.

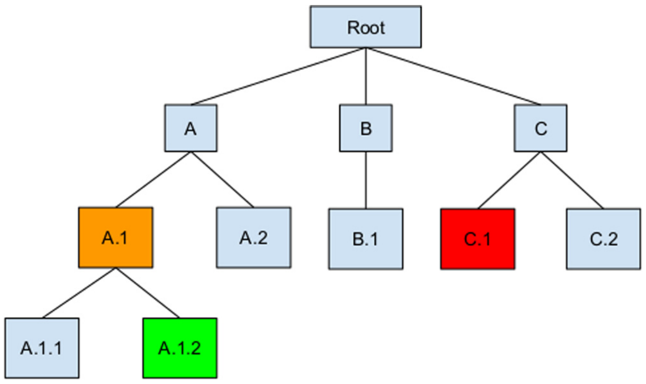


Fig. 4. An example of a class hierarchy tree. The green node represents the correct classification, while the red and orange nodes represent two misclassification results obtained by two different classifiers. We used a color code from green to red to indicate the error impact. Green means a perfect match, whereas red (the $C.1$ class) is the worst case among the classification results considered; orange (the $A.1$ class) has a minor impact on the red error, considering the hierarchical class structure.

Assuming that the true class for a test instance is the one lying in node A.1.2 (the green one in Fig. 4), if two different classifiers output respectively class A.1 and C.1 as the predicted class (in orange and red in the figure), a flat evaluation measure will equally penalize both wrong predictions. Actually, following the class hierarchy, the system whose prediction is on the same sub-tree of the true label (A.1) has produced a better result if compared with a prediction in a totally different and unrelated sub-tree (C.1). This behavior should be considered in the measure, in order to better compare the performances: the hierarchical measures have been defined to overcome this limitation of flat measures.

The first class of hierarchical measure is called *set-based*, because these measures take into account the entire sets of the true and predicted classes Y and $\hat{Y}$, including also their ancestors in the tree. The definition of hierarchical measures is based on the augmented set of the true and predicted classes, respectively Y_{aug} and $\hat{Y}_{aug}$, which are equal to:

$$Y_{aug} = Y \cup An(y_1) \cup \cdots \cup An(y_N) \tag{19}$$

$$\hat{Y}_{aug} = \hat{Y} \cup An(\hat{y}_1) \cup \cdots \cup An(\hat{y}_M) \tag{20}$$

where $An(y_n)$ and $An(\hat{y}_m)$ are the ancestors of the true and predicted classes. The hierarchical Precision *HiP*, hierarchical Recall *HiR* and hierarchical F1 *HiF* [59], which consider the classification results of a single document as a sub-tree [64] including the whole path in the hierarchy, are defined as:

$$HiP = \frac{|\hat{Y}_{aug} \cap Y_{aug}|}{|\hat{Y}_{aug}|} \tag{21}$$

$$HiR = \frac{|\hat{Y}_{aug} \cap Y_{aug}|}{|Y_{aug}|} \tag{22}$$

$$HiF = \frac{2 \cdot HiP \cdot HiR}{HiP + HiR} \tag{23}$$

Table 5
Parameter number of the DNNs in function of the embedding model vector size.

Embedding model	*Vector size*	*DNN parameters #*
NLP Preprocessed	512	85,673,999
BioASQ	200	69,699,599
BioASQ + POS	$200 + 50$	72,259,599
BioASQ + POS + FT	$200 + 50 + 200$	82,499,599
BioASQ + POS + FT + DT	$200 + 50 + 200 + 250$	95,299,599

The Y_{aug} and $\hat{Y}_{aug}$ define two sub-trees that represent the true classification path and the classified path, respectively. The intersection between these two sub-trees, represents their similarity. The *HiP* is the ratio between the sub-tree intersection and the classified classification paths, while the *HiR* is the ratio between the intersection and the true classification paths.

Adding all the ancestors in the sets overestimates the error when the tree has nodes with many ancestors. In order to correct this behavior, in [65] different measures have been proposed: Lowest Common Ancestor Precision *LCaP*, Lowest Common Ancestor Recall *LCaR* and Lowest Common Ancestor F1 *LCaF*, based on the lowest common ancestor concept from graph theory [66]. The lowest common ancestor $LCA(n_1, n_2)$ of two nodes n_1 and n_2 of a tree T is defined as the node in T furthest from the root that is an ancestor of both n_1 and n_2. Following this definition, it is possible to construct the augmented sets of the true and predicted classes Y_{aug_L} and $\hat{Y}_{aug_L}$ in accordance with the following description. The first step needed to obtain this type of hierarchical measure is to calculate for each class y in the set of true classes Y the lowest common ancestor with respect to the set of predicted classes $\hat{Y}$ as:

$$LCA(y, \hat{Y}) = \underset{m}{argmin} \cdot \gamma(m, y) \tag{24}$$

where $\gamma(u, v)$ is the distance between the nodes u and v in the tree. In the same way, for each class $\hat{y}$ in the set of predicted classes $\hat{Y}$ the lowest common ancestor with respect to the set of true classes Y is calculated as:

$$LCA(\hat{y}, Y) = \underset{m}{argmin} \cdot \gamma(m, \hat{y}) \tag{25}$$

Afterwards, it is possible to define two graphs G_t and G_p containing respectively the shortest path from each $y \in Y$ to $LCA(y, \hat{Y})$, and the shortest path from each $\hat{y} \in \hat{Y}$ to $LCA(\hat{y}, Y)$. Using these graphs, a set Y_{aug_L} can be constructed with all the nodes in the graph G_t and, similarly, a set $\hat{Y_{aug_L}}$ will contain all the nodes in the graph G_p. In this way, *LCaP*, *LCaR* and *LCaF* are defined as:

$$LCaP = \frac{|\hat{Y}_{aug_L} \cap Y_{aug_L}|}{|\hat{Y}_{aug_L}|} \tag{26}$$

$$LCaR = \frac{|\hat{Y}_{aug_L} \cap Y_{aug_L}|}{|Y_{aug_L}|} \tag{27}$$

$$LCaF = \frac{2 \cdot LCaP \cdot LCaR}{LCaP + LCaR} \tag{28}$$

We have reported in our experimental assessment all the above described metrics, with the objective of highlight all aspects of the classification performances. In addition, this kind of evaluation follows the BioASQ shared task assessment methodology [8].

4.4. Experimental results and discussion

We have evaluated the performances of the DNNs described in Section 3, with the aim of analyzing two main aspects: to investigate the effects of different embedding models in the input layer and to assess the utility of the proposed HLSE methodology. To achieve this objective, we have evaluated the performances of the DNNs that differ simply in terms of the types of embedding, previously described in Section 3.3. To give an idea of the complexity of each model, Table 5 reports the number of parameters of the implemented DNNs in function of the embedding model used and, consequently, of the corresponding vector size. A direct correlation between the vector size and the corresponding number of network parameters is clear.

We first tested the WE model obtained using the NLP preprocessed training set, hereinafter *NLP Preprocessed* and the BioASQ pre-trained WE model, hereinafter *BioASQ*. Next, following the methodology explained in the previous Section 3.3, we obtained another embedding model by concatenating the word vectors from the BioASQ model with the POS embedding, naming this model *BioASQ+POS*. A further embedding model was obtained by concatenating the fastText word vectors with the previous BioASQ+POS model, creating in this way the *BioASQ+POS+FT* model. Finally, we concatenated the word vectors from the Dependency Tree embedding model with the latter, creating the *BioASQ+POS+FT+DT* model. All the aforementioned DNN models were tested by applying the HLSE methodology to both the training set and testing set, expanding the label set of each sample following the hierarchy class paths. In order to assess the performances of the proposed HLSE method, we tested the same DNN without applying HLSE with the *BioASQ+POS+FT+DT* model as input, referring to this test as *BioASQ+POS+FT+DT NoHLSE* and the NLP Preprocessed model without the application of HLSE.

Table 6 shows the performances of the considered DNN architecture obtained with all the above described models and tests, evaluated in terms of flat measures, while Table 7 reports the results in terms of the hierarchical measures, described in Section 4.3.

The results reported in Tables 6 and 7 show that the best performing model is the NLP Preprocessed with HLSE in terms of all measures, both flat and hierarchical, except for the *MaP* and *MiP* scores, as expected due to the higher Recall value. Focusing on the *MiF* score, we can see that the NLP Preprocessed model achieves a relative increment of the *MiF* score greater than 43%, if compared with the second best performing model in terms of *MiF* (the BioASQ+POS). On the other hand, we can also observe that the NLP Preprocessed NoHLSE model obtains the lowest scores, except in *MaP* and *EbP*, where it obtains the best result among all the tested models, due to the lower average label number per document (see 3) and to the lack of HLSE. We can also observe that in the case of the hierarchical measures the *hF* and the $LCA - F$ scores of the NLP preprocessed model obtain respectively a relative increment of 38.48% and 58.33% compared to the second best performing model. Considering also the previous results with the NLP Preprocessed NoHLSE model, the experiments confirm that the HLSE methodology improves the correct prediction on the same path of the hierarchy. Focusing on the semi-supervised embedding models, we can first observe that the BioASQ+POS model has a slight performance improvement if compared with the basic BioASQ model, while the concatenation with the other embedding models, such as fastText and Dependency Tree, produces worse performances if compared to those obtained with the BioASQ model. Analyzing the performances obtained with the various embedding models tested, the results prove that the use of those trained with additional information, such as POS or Dependency Tree, does not provide improvements if applied to this DNN architecture. On the contrary, the best WE input model is the one obtained by applying a NLP normalizing pipeline to the training text and using the HLSE methodology. This behavior is completely different from

Table 6
Performances obtained by the DNN classifiers using different embedding models with and without the HLSE application, in terms of flat measures.

TEST	FLAT measures									
	Acc	EbP	EbR	EbF	MaP	MaR	MaF	MiP	MiR	MiF
BioASQ	0.2484	0.6311	0.2847	0.3841	0.6408	0.0011	0.0009	0.6320	0.2777	0.3858
BioASQ+POS	0.2511	0.6221	0.2903	0.3874	0.6220	0.0011	0.0009	0.6221	0.2832	0.3892
BioASQ+POS+FT	0.2414	0.6448	0.2737	0.3763	0.6346	0.0010	0.008	0.6445	0.2662	0.3768
BioASQ+POS+FT+DT	0.2369	0.6287	0.2664	0.3667	0.6287	0.0010	0.0008	0.6294	0.2605	0.3685
BioASQ+POS+FT+DT NoHLSE	0.1857	0.7194	0.1939	0.3007	0.7136	0.0007	0.0006	0.7216	0.1890	0.2995
NLP Preprocessed	**0.3893**	0.6166	**0.5235**	**0.5496**	0.3779	**0.0474**	**0.0535**	0.6084	**0.5173**	**0.5592**
NLP Preprocessed NoHLSE	0.1690	**0.7676**	0.1728	0.2779	**0.7676**	0.0005	0.0005	**0.7676**	0.1669	0.2742

Table 7
Performances obtained by the DNN classifiers using different embedding models with and without the HLSE application, in terms of hierarchical measures.

TEST	Hierarchical measures					
	hP	hR	hF	LCA-P	LCA-R	LCA-F
BioASQ	0.6470	0.3004	0.4016	0.4490	0.1458	0.2099
BioASQ+POS	0.6378	0.3058	0.4046	0.4285	0.1484	0.2100
BioASQ+POS+FT	0.6611	0.2896	0.3943	0.4513	0.1346	0.1978
BioASQ+POS+FT+DT	0.6457	0.2825	0.3850	0.4585	0.1309	0.1943
BioASQ+POS+FT+DT NoHLSE	0.7393	0.2116	0.3236	**0.5141**	0.0731	0.1226
NLP Preprocessed	0.6265	**0.5339**	**0.5603**	0.4036	**0.3094**	**0.3325**
NLP Preprocessed NoHLSE	**0.7870**	0.1909	0.3023	0.1762	0.0533	0.0795

that observed in literature for other NLP tasks [47,67], such as Named Entity and Relation Extraction, where the use of semi-supervised WE models trained with POS, a representation of word grammatical classification, and Dependency Tree, an explicit representation of the relations between words in sentences, are able to provide a performance boost through information directly involved in the identification of relations and domain entities. In the considered XMTC task only the POS tags provide an improvement of the model, because the concatenation of the word vector and POS vector places the resulting vector from the BioASQ+POS model on the same hyperplane when words correspond to the same POS tag, simplifying the vector space model obtained. In the other cases, the most complex information provided by the other models adds noise to the resulting vector space. For the same reasons, the simplified training set obtained with NLP Preprocessing creates a simpler vector space in terms of dictionary length, making the classifier able to better extract the most important features from the text related to the output labels. In summary, the NLP normalization of the training texts aims at simplifying the complexity and variability of the natural language, whereas the semi-supervised techniques for POS tagging and Dependency Tree enhance the text with additional explicit grammatical and syntactic information.

The comparison of the results obtained with the HLSE application with the other experiments without HLSE (BioASQ+POS+FT+DT NoHLSE and NLP Preprocessed NoHLSE) show a performance boost. The only drawbacks of HLSE are lower *MaP*, *MiP*, *EbP*, *hP* and $LCA - P$ values, caused by the higher number of labels per document (see Table 3) and by the higher recall scores obtained. On the other hand, focusing on the hierarchical measures, we can observe high values that denote the correct classification of the test set samples in the same hierarchy tree path. These results confirm the effectiveness of the proposed HLSE methodology.

The comparison between our best performing model (NLP Preprocessed) with the state of the art systems devoted to automatic MeSH labeling can provide additional important data about our analysis. Tables 8 and 9 compare the results obtained by the state of the art systems specifically devoted to the PubMed labeling task with the NLP Preprocessed model. It is important, when discussing these results, to take into account also the corresponding approaches, summarized in the previous Table 1.

Considering in particular the *MiF* measure, our model outperforms rule based systems, such as *Search* and *Iria*. This confirms the limitation of this kind of approach, which suffers from a lack of generalization, relying on fixed rules. The systems based on ML are able to produce better results, except for the *MZ*. On the other hand, ML models have some important drawbacks: they all need manual feature engineering and almost all of them need to train a very large number of classifiers, equal to the number of classes to be predicted, making the use of these approaches very complex. Our DL based model obtains slightly worse performances, but requires to training of only a single CNN-based model. Focusing on the comparison with other DL based approaches, we can see that our model outperforms the *Sequencer* system, but obtains worse results if compared to the *STML* system. It must be remarked that the performances achieved by our NLP Preprocessed model and by the *STML* system prove that DL architectures are actually comparable with the state of the art ML approaches for the XMTC task, with the advantage of not requiring manual feature engineering or the training of a very high number of different classifiers.

In summary, the HLSE methodology is able to provide a performance boost despite a more complex problem caused by the higher average label number to be predicted; this is confirmed also by the hierarchical measures scores. We observed that the application of NLP normalization to the embedding training corpus is another means to obtain a performance improvement, if compared with the embeddings obtained with a non-preprocessed training text. We also observe that the concatenation of additional embeddings such as fastText, POS embeddings and Dependency Tree embeddings does not provide a substantial performance improvement, producing in some cases even worse results.

5. Conclusions

In this paper we have presented an analysis of a Deep Learning architecture devoted to text classification, considering the extreme multi-class and multi-label text classification problem, when a hierarchical label set is defined. We have described a methodology named Hierarchical Label Set Expansion (HLSE) used to regularize the data labels and have reported an analysis of the impact of the use and combination of different Word Embedding (WE) models that explicitly incorporate grammatical and syntactic features.

We have evaluated the aforementioned methodologies on the PubMed scientific articles collection, a multi-class and multi-label text classification problem with a hierarchical set of 27,775 classes.

The experimental assessment results have proved the usefulness of the proposed HLSE methodology, despite there being a more complex problem due to the higher label set of each sample. In addition, the results obtained have highlighted some interesting aspects about the embedding models in terms of both their use and combination as input of neural networks in XMTC

Table 8
Performances of the state of the art systems devoted to PubMed labeling in terms of flat measures.

System	FLAT measures									
	MiF	EbP	EbR	EbF	MaP	MaR	MaF	MiP	MiR	Acc
Default MTI	0.6272	0.6390	0.6425	0.6208	0.5958	0.5802	0.5425	0.6315	0.6232	0.4676
MTI First Line Index	0.6136	0.6730	0.5872	0.6061	0.6298	0.5409	0.5210	0.6634	0.5708	0.4539
DeepMeSH	0.6558	0.7055	0.6315	0.6480	0.6654	0.5235	0.5205	0.7034	0.6144	0.4960
Sequencer	0.2785	0.2831	0.2853	0.2731	0.2061	0.1526	0.1460	0.2745	0.2829	0.1896
Search System	0.2541	0.2491	0.3096	0.2535	0.3674	0.4102	0.3409	0.2350	0.2970	0.1515
MZ	0.4733	0.5233	0.4602	0.4644	0.5708	0.2270	0.2304	0.5173	0.4428	0.3167
auth	0.6286	0.6334	0.6380	0.6176	0.6288	0.5022	0.4907	0.6373	0.6229	0.4622
Iria	0.4917	0.4425	0.5628	0.4822	0.4104	0.4176	0.3790	0.4469	0.5469	0.3312
SMTL	0.6220	//	//	//	//	//	//	0.7750	0.5199	//
MeSH NOW	0.6100	//	//	//	//	//	//	0.6120	0.6080	//
NLP Preprocessed	0.5592	0.6166	0.5235	0.5496	0.3779	0.0474	0.0535	0.6084	0.5173	0.3893

Table 9
Performances of the state of the art systems devoted to PubMed labeling in terms of hierarchical measures.

System	Hierarchical measures					
	LCA-F	HiP	HiR	HiF	LCA-P	LCA-R
Default MTI	0.5058	0.7620	0.7354	0.7297	0.5314	0.5097
MTI First Line Index	0.4911	0.7816	0.6966	0.7149	0.5406	0.4781
DeepMeSH	0.5228	0.8060	0.7269	0.7458	0.5688	0.5092
Sequencer	0.2867	0.5099	0.3941	0.4007	0.3509	0.2804
Search System	0.2863	0.4888	0.5274	0.4762	0.2873	0.3307
MZ	0.3840	0.6836	0.5727	0.5903	0.4467	0.3718
auth	0.5048	0.7656	0.7306	0.7285	0.5335	0.5066
Iria	0.4232	0.6015	0.6995	0.6309	0.4067	0.4661
SMTL	//	//	//	//	//	//
MeSH NOW	//	//	//	//	//	//
NLP Preprocessed	0.3325	0.6265	0.5339	0.5603	0.4036	0.3094

applications, suggesting the utility of NLP Preprocessed text for WE training and, differently from other tasks, the lack of any performance gain from the use of semi-supervised embedding models.

References

[1] M.M. Mirończuk, J. Protasiewicz, A recent overview of the state-of-the-art elements of text classification, Expert Syst. Appl. 106 (2018) 36–54, http://dx.doi.org/10.1016/j.eswa.2018.03.058.

[2] A. Alicante, M. Benerecetti, A. Corazza, S. Silvestri, A distributed architecture to integrate ontological knowledge into information extraction, Int. J. Grid Utility Comput. 7 (4) (2016) 245–256, http://dx.doi.org/10.1504/IJGUC.2016.10001945.

[3] J. Liu, W. Chang, Y. Wu, Y. Yang, Deep learning for extreme multi-label text classification, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, Shinjuku, Tokyo, Japan, 2017, pp. 115–124, http://dx.doi.org/10.1145/3077136.3080834.

[4] J. Wang, J. Zhang, Y. An, H. Lin, Z. Yang, Y. Zhang, Y. Sun, Biomedical event trigger detection by dependency-based word embedding, BMC Med. Genomics 9 (2) (2016) 45, http://dx.doi.org/10.1186/s12920-016-0203-8.

[5] A. Komninos, S. Manandhar, Dependency based embeddings for sentence classification tasks, in: NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, ACL, San Diego California, USA, 2016, pp. 1490–1500.

[6] O. Levy, Y. Goldberg, Dependency-based word embeddings, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, Volume 2: Short Papers, ACL, Baltimore, MD, USA, 2014, pp. 302–308, http://dx.doi.org/10.3115/v1/P14-2050.

[7] A. Trask, P. Michalak, J. Liu, sense2vec - A fast and accurate method for word sense disambiguation in neural word embeddings, arXiv preprint arXiv:1511.06388.

[8] A. Nentidis, K. Bougiatiotis, A. Krithara, G. Paliouras, I.A. Kakadiaris, Results of the fifth edition of the BioASQ Challenge, in: BioNLP 2017, Vancouver, Canada, August 4, 2017, ACL, Vancouver, Canada, 2017, pp. 48–57, http://dx.doi.org/10.18653/v1/W17-2306.

[9] A. Holzinger, P. Kieseberg, E.R. Weippl, A.M. Tjoa, Current advances, trends and challenges of machine learning and knowledge extraction: from machine learning to explainable AI, in: Machine Learning and Knowledge Extraction - Second IFIP TC 5, TC 8/WG 8.4, 8.9, TC 12/WG 12.9 International Cross-Domain Conference, CD-MAKE 2018, in: Lecture Notes in Computer Science, vol. 11015, Springer, Hamburg, Germany, 2018, pp. 1–8, http://dx.doi.org/10.1007/978-3-319-99740-7_1.

[10] R. Goebel, A. Chander, K. Holzinger, F. Lécué, Z. Akata, S. Stumpf, P. Kieseberg, A. Holzinger, Explainable AI: the new 42?, in: Machine Learning and Knowledge Extraction - Second IFIP TC 5, TC 8/WG 8.4, 8.9, TC 12/WG 12.9 International Cross-Domain Conference, CD-MAKE 2018, in: Lecture Notes in Computer Science, vol. 11015, Springer, Hamburg, Germany, 2018, pp. 295–303, http://dx.doi.org/10.1007/978-3-319-99740-7_21.

[11] Y. Zhang, J. Ma, Z. Wang, B. Chen, LF-LDA: a topic model for multi-label classification, in: Advances in Internetworking, Data & Web Technologies, The 5th International Conference on Emerging Internetworking, Data & Web Technologies, EIDWT-2017, Springer, Wuhan, China, 2017, pp. 618–628, http://dx.doi.org/10.1007/978-3-319-59463-7_62.

[12] M. Pavlinek, V. Podgorelec, Text classification method based on self-training and lda topic models, Expert Syst. Appl. 80 (Supplement C) (2017) 83–93, http://dx.doi.org/10.1016/j.eswa.2017.03.020.

[13] J. Nam, J. Kim, E. Loza Mencía, I. Gurevych, J. Fürnkranz, Large-scale multi-label text classification - revisiting neural networks, in: Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Springer, Nancy, France, 2014, pp. 437–452, http://dx.doi.org/10.1007/978-3-662-44851-9_28.

[14] M. Hughes, I. Li, S. Kotoulas, T. Suzumura, Medical text classification using convolutional neural networks, CoRR 235 (2017) 246–250.

[15] D. Yogatama, C. Dyer, W. Ling, P. Blunsom, Generative and Discriminative Text Classification with Recurrent Neural Networks, CoRR abs/1703.01898.

[16] H. Schwenk, L. Barrault, A. Conneau, Y. LeCun, Very deep convolutional networks for text classification, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Vol. 1, ACL, Valencia, Spain, 2017, pp. 1107–1116.

[17] Y. Yan, Y. Wang, W. Gao, B. Zhang, C. Yang, X. Yin, LSTM^2 : multi-label ranking for document classification, Neural Process. Lett. 47 (1) (2018) 117–138, http://dx.doi.org/10.1007/s11063-017-9636-0.

[18] Y. Wang, F. Tian, Recurrent residual learning for sequence classification, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, ACL, Austin, Texas, USA, 2016, pp. 938–943.

[19] G. Chen, D. Ye, Z. Xing, J. Chen, E. Cambria, Ensemble application of convolutional and recurrent neural networks for multi-label text categorization, in: 2017 International Joint Conference on Neural Networks, IJCNN 2017, IEEE, Anchorage, AK, USA, 2017, pp. 2377–2383, http://dx.doi.org/10.1109/IJCNN.2017.7966144.

[20] T. Baumel, J. Nassour-Kassis, R. Cohen, M. Elhadad, N. Elhadad, Multi-label classification of patient notes: case study on ICD code assignment, in: The Workshops of the The Thirty-Second AAAI Conference on Artificial Intelligence., AAAI, New Orleans, Louisiana, USA, 2018, pp. 409–416.

[21] P. Nigam, Applying deep learning to ICD-9 multi-label classification from medical records, Tech. rep., Stanford University, 2017.

[22] H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, Q. Yang, Large-scale hierarchical text classification with recursively regularized deep graph-cnn, in: Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, ACM, Lyon, France, 2018, pp. 1063–1072, http://dx.doi.org/10.1145/3178876.3186005.

[23] J.G. Mork, D. Demner-Fushman, S. Schmidt, A.R. Aronson, Recent enhancements to the NLM medical text indexer, in: Working Notes for CLEF 2014 Conference, Vol. 1180, CEUR-WS.org, Sheffield, UK, 2014, pp. 1328–1336, URL http://ceur-ws.org/Vol-1180/CLEF2014wn-QA-MorkEt2014.pdf.

[24] I. Zavorin, J. Mork, D. Demner-Fushman, Using learning-to-rank to enhance NLM medical text indexer results, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL, Berlin, Germany, 2016, pp. 8–15.

[25] A.R. Aronson, Effective mapping of biomedical text to the UMLS metathesaurus: the metamap program, in: AMIA 2001, American Medical Informatics Association Annual Symposium, AMIA, Washington, DC, USA, 2001, pp. 17–21.

[26] J. Lin, W.J. Wilbur, Pubmed related articles: a probabilistic topic-based model for content similarity, BMC Bioinformatics 8 (1) (2007) 423, http://dx.doi.org/10.1186/1471-2105-8-423.

[27] S. Peng, R. You, H. Wang, C. Zhai, H. Mamitsuka, S. Zhu, Deepmesh: deep semantic representation for improving large-scale mesh indexing, Bioinformatics 32 (12) (2016) 70–79, http://dx.doi.org/10.1093/bioinformatics/btw294.

[28] S. Peng, H. Mamitsuka, S. Zhu, Meshlabeler and DeepMeSH: recent progress in large-scale mesh indexing, in: Data Mining for Systems Biology. Methods in Molecular Biology, Vol. 1807, Springer, 2018, pp. 203–209, http://dx.doi.org/10.1007/978-1-4939-8561-6_15.

[29] Q.V. Le, T. Mikolov, Distributed representations of sentences and documents, in: Proceedings of the 31th International Conference on Machine Learning, ICML 2014, JMLR.org, Beijing, China, 2014, pp. 1188–1196.

[30] E. Papagiannopoulou, Y. Papanikolaou, D. Dimitriadis, S. Lagopoulos, G. Tsoumakas, M. Laliotis, N. Markantonatos, I. Vlahavas, Large-scale semantic indexing and question answering in biomedicine, in: Proceedings of the Fourth BioASQ workshop, ACL, 2016, pp. 50–54.

[31] Y. Papanikolaou, G. Tsoumakas, M. Laliotis, N. Markantonatos, I.P. Vlahavas, Large-scale online semantic indexing of biomedical articles via an ensemble of multi-label classification models, J. Biomed. Semant. 8 (1) (2017) 43:1–43:13, http://dx.doi.org/10.1186/s13326-017-0150-0.

[32] Y. Mao, Z. Lu, MeSH Now: automatic MeSH indexing at PubMed scale via learning to rank, J. Biomed. Semant. 8 (1) (2017) 15:1–15:9, http://dx.doi.org/10.1186/s13326-017-0123-3.

[33] K. Liu, S. Peng, J. Wu, C. Zhai, H. Mamitsuka, S. Zhu, MeSHLAbeler: improving the accuracy of large-scale MeSH indexing by integrating diverse evidence, Bioinformatics 31 (12) (2015) 339–347, http://dx.doi.org/10.1093/bioinformatics/btv237.

[34] F.J. Ribadas-Pena, L.M. de Campos, V.M.D. Bilbao, A.E. Romero, CoLe and UTAI at BioASQ 2015: experiments with similarity based descriptor assignment, in: Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, CEUR-WS.org, Toulouse, France, 2015, URL http://ceur-ws.org/Vol-1391/84-CR.pdf.

[35] Y. Du, Y. Pan, J. Ji, A novel serial deep multi-task learning model for large scale biomedical semantic indexing, in: 2017 IEEE International Conference on Bioinformatics and Biomedicine, BIBM, IEEE, Kansas City, MO, USA, 2017, pp. 533–537, http://dx.doi.org/10.1109/BIBM.2017.8217704.

[36] M.A. Tanenblatt, A. Coden, I.L. Sominsky, The conceptmapper approach to named entity recognition, in: Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, European Language Resources Association, Valletta, Malta, 2010, pp. 546–551.

[37] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013., Lake Tahoe, Nevada, USA, 2013, pp. 3111–3119.

[38] I. Ilievski, T. Akhtar, J. Feng, C.A. Shoemaker, Efficient hyperparameter optimization for deep learning algorithms using deterministic rbf surrogates, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17), AAAI, San Francisco, California, USA, 2017, pp. 822–829.

[39] R.S. Sutton, Two problems with backpropagation and other steepest-descent learning procedures for networks, in: Proceedings of 8th annual conference of cognitive science society, Erlbaum, 1986, pp. 823–831.

[40] B.T. Polyak, Some methods of speeding up the convergence of iteration methods, USSR Comput. Math. Math. Phys. 4 (5) (1964) 1–17.

[41] Y. Nesterov, A method for unconstrained convex minimization problem with the rate of convergence O(1/kˆ 2), in: Doklady AN USSR, Vol. 269, 1983, pp. 543–547.

[42] F. Gargiulo, S. Silvestri, M. Ciampi, Deep convolution neural network for extreme multi-label text classification, in: Proceedings of the 11th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2018) - Volume 5: HEALTHINF, Vol. 5, SCITEPRESS, Funchal, Madeira, Portugal, 2018, pp. 641–650, http://dx.doi.org/10.5220/0006730506410650.

[43] A. Alicante, A. Corazza, F. Isgrò, S. Silvestri, Semantic cluster labeling for medical relations, in: Innovation in Medicine and Healthcare 2016, Springer, 2016, pp. 183–193, http://dx.doi.org/10.1007/978-3-319-39687-3_18.

[44] F. Gargiulo, S. Silvestri, M. Ciampi, A clustering based methodology to support the translation of medical specifications to software models, Appl. Soft Comput. 71 (2018) 199–212, http://dx.doi.org/10.1016/j.asoc.2018.03.057, URL http://www.sciencedirect.com/science/article/pii/S1568494618303685.

[45] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: Proceedings of the International Conference on Learning Representations (ICLR 2013), 2013.

[46] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, T. Mikolov, Fast-Text.zip: Compressing text classification models, arXiv preprint arXiv:1612.03651.

[47] S.M. Rezaeinia, A. Ghodsi, R. Rahmani, Improving the Accuracy of Pre-trained Word Embeddings for Sentiment Analysis, CoRR abs/1711.08609.

[48] F. Gargiulo, S. Silvestri, M. Fontanella, M. Ciampi, G. De Pietro, A deep learning approach for scientific paper semantic ranking, in: International Conference on Intelligent Interactive Multimedia Systems and Services, Springer, Vilamoura, Portugal, 2017, pp. 471–481, http://dx.doi.org/10.1007/978-3-319-59480-447.

[49] I. Pavlopoulos, A. Kosmopoulos, I. Androutsopoulos, Continuous Space Word Vectors Obtained by Applying Word2Vec to Abstracts of Biomedical Articles, Tech. rep., NLP Group, Department of Informatics, Athens University of Economics and Business, Greece Institute of Informatics and Telecommunications, NCRS Demokritos, Greece, 2014.

[50] Q. Liu, Z. Ling, H. Jiang, Y. Hu, Part-of-Speech Relevance Weights for Learning Word Embeddings, CoRR abs/1603.07695.

[51] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Trans. Assoc. Comput. Linguist. 5 (2017) 135–146.

[52] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, Bag of tricks for efficient text classification, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, ACL, 2017, pp. 427–431, http://dx.doi.org/10.18653/v1/E17-2068.

[53] O. Melamud, O. Levy, I. Dagan, A simple word embedding model for lexical substitution, in: Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing, VS@NAACL-HLT 2015, ACL, Denver, Colorado, USA, 2015, pp. 1–7, http://dx.doi.org/10.3115/v1/W15-1501.

[54] F. Gargiulo, S. Silvestri, M. Ciampi, A big data architecture for knowledge discovery in PubMed articles, in: 2017 IEEE Symposium on Computers and Communications, ISCC 2017, IEEE, Heraklion, Greece, 2017, pp. 82–87, http://dx.doi.org/10.1109/ISCC.2017.8024509.

[55] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.

[56] C.D. Manning, M. Surdeanu, J. Bauer, J.R. Finkel, S. Bethard, D. McClosky, The stanford coreNLP natural language processing toolkit, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, ACL, Baltimore, MD, USA, 2014, pp. 55–60, http://dx.doi.org/10.3115/v1/P14-5010.

[57] R. Řehůřek, P. Sojka, Software framework for topic modelling with large corpora, in: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, ELRA, 2010, pp. 45–50, http://dx.doi.org/10.13140/2.1.2393.1847.

[58] S. Bird, E. Klein, E. Loper, Natural Language Processing with Python, O'Reilly, 2009.

[59] G. Tsoumakas, I. Katakis, I. Vlahavas, Mining multi-label data, in: Data Mining and Knowledge Discovery Handbook, 2nd ed., Springer, 2010, pp. 667–685, http://dx.doi.org/10.1007/978-0-387-09823-4_34.

[60] A. Özgür, L. Özgür, T. Güngör, Text categorization with class-based and corpus-based keyword selection, Comput. Inf. Sci.-ISCIS 2005 (2005) 606–615.

[61] C.D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2010, http://dx.doi.org/10.1007/s10791-009-9115-y.

[62] M. Sokolova, G. Lapalme, A systematic analysis of performance measures for classification tasks, Inf. Process. Manage. 45 (4) (2009) 427–437, http://dx.doi.org/10.1016/j.ipm.2009.03.002.

[63] S. Godbole, S. Sarawagi, Discriminative methods for multi-labeled classification, in: H. Dai, R. Srikant, C. Zhang (Eds.), Advances in Knowledge Discovery and Data Mining, Springer Berlin, Berlin, Heidelberg, 2004, pp. 22–30, http://dx.doi.org/10.1007/978-3-540-24775-3_5.

[64] R. Moskovitch, S. Cohen-Kashi, U. Dror, I. Levy, A. Maimon, Y. Shahar, Multiple hierarchical classification of free-text clinical guidelines, Artif. Intell. Med. 37 (3) (2006) 177–190, http://dx.doi.org/10.1016/j.artmed.2006.04.001.

[65] A. Kosmopoulos, I. Partalas, É. Gaussier, G. Paliouras, I. Androutsopoulos, Evaluation measures for hierarchical classification: a unified view and novel approaches, Data Min. Knowl. Discov. 29 (3) (2015) 820–865, http://dx.doi.org/10.1007/s10618-014-0382-x.

[66] A.V. Aho, J.E. Hopcroft, J.D. Ullman, On finding lowest common ancestors in trees, SIAM J. Comput. 5 (1) (1976) 115–132, http://dx.doi.org/10.1137/0205011.

[67] S. Ramamoorthy, S. Murugan, An Attentive Sequence Model for Adverse Drug Event Extraction from Biomedical Text, CoRR abs/1801.00625.