Automated MeSH Indexing of Biomedical Literature Using Contextualized Word Representations

Dimitrios A. Koutsomitropoulos(✉) and Andreas D. Andriopoulos

Computer Engineering and Informatics Department, School of Engineering,
University of Patras, 26504 Patras, Greece
koutsomi@ceid.upatras.gr, a.andriopoulos@upatras.gr

Abstract. Appropriate indexing of resources is necessary for their efficient search, discovery and utilization. Relying solely on manual effort is time-consuming, costly and error prone. On the other hand, the special nature, volume and broadness of biomedical literature pose barriers for automated methods. We argue that current word embedding algorithms can be efficiently used to support the task of biomedical text classification. Both deep- and shallow network approaches are implemented and evaluated. Large datasets of biomedical citations and full texts are harvested for their metadata and used for training and testing. The ontology representation of Medical Subject Headings provides machine-readable labels and specifies the dimensionality of the problem space. These automated approaches are still far from entirely substituting human experts, yet they can be useful as a mechanism for validation and recommendation. Dataset balancing, distributed processing and training parallelization in GPUs, all play an important part regarding the effectiveness and performance of proposed methods.

Keywords: Classification · Indexing · Word embeddings · Thesauri · Ontologies · Doc2Vec · ELMo · MeSH · Deep learning

1 Introduction

Digital biomedical assets include a variety of information ranging from medical records to equipment measurements to clinical trials and research outcomes. The digitization and availability of biomedical literature is important at least in two aspects: first, this information is a valuable source for Open Education Resources (OERs) that can be used in distance training and e-learning scenarios; second, future research advancements can stem from the careful examination and synthesis of past results.

For both these directions to take effect, it is critical to consider automatic classification and indexing as a means to enable efficient knowledge management and discovery for these assets. In addition, the sheer volume of biomedical literature is continuously increasing and puts excessive strain on manual cataloguing processes: For example, the

I. Maglogiannis et al. (Eds.): AIAI 2020, IFIP AICT 583, pp. 343–354, 2020.
https://doi.org/10.1007/978-3-030-49161-1_29

US National Library of Medicine experiences daily a workload of approximately 7,000 articles for processing [12].

Research in the automatic indexing of literature is constantly advancing and various approaches are recently proposed, a fact that indicates this is still an open problem. These approaches include multi-label classification using machine learning techniques, training methods and models from large lexical corpora as well as semantic classification approaches using existing thematic vocabularies. To this end, the Medical Subject Headings (MeSH) is the de-facto standard for thematically annotating biomedical resources [18].

In this paper we propose and evaluate an approach for automatically annotating biomedical articles with MeSH terms. While such efforts have been investigated before, in this work we are interested in the performance of current state-of-the-art algorithms based on contextualized word representations or word embeddings. We suggest producing vectorized word and paragraph representations of articles based on context and existing thematic annotations (labels). Consequently, we seek to infer the most similar terms stored by the model without the need and overhead of a separate classifier. Moreover, we combine these algorithms with structured semantic representations in Web Ontology Language format (OWL), such as the implementation of the MeSH thesaurus in OWL Simple Knowledge Organization Systems (SKOS) [20]. Finally, we investigate the effect and feasibility of employing distributed data manipulation and file system techniques for dataset preprocessing and training.

The rest of this paper is organized as follows: in Sect. 2 we summarize current word embedding approaches as the main background and identify the problem of automated indexing; in Sect. 3 we review relevant literature in the field of biomedical text classification; Sect. 4 presents our methodology and approach, by outlining the indexing procedure designed, describing the algorithms used and discussing optimizations regarding dataset balancing, distributed processing and training parallelization. Section 5 contains the results of the various experiments and their analysis, while Sect. 6 outlines our conclusions and future work.

2 Background

Word embedding techniques [9] convert words into word vectors. The following approaches have emerged in recent years with the performance of text recognition as the primary objective. The start has been made with the Word2Vec algorithm [11] where unique vector word representations are generated by means of shallow neural networks and the prediction method as well as by the explicit extension Doc2Vec (Document to Vector) [8], where a unique vector representation can also be given for whole texts. Next, the Global Vectors (GloVe) algorithm [14] manages to transfer the words into a vector space by making use of the enumeration method. Then, the FastText algorithm [4] achieves not only the management of a large bulk of data in optimal time but also better word embedding due to the use of syllables. In addition, the ELMo algorithm [15] uses deep neural networks, LSTMs, and a different vector representation for a word the meaning of which differentiates. Lastly, the BERT algorithm [2] also generates different representations of a word according to its meaning, but instead of LSTMs it uses transformer elements [21].

Assigning a topic to text data is a demanding process. Nevertheless, if approached correctly, it ensures easier and more accurate access for the end user. A typical subcase of the topic assignment problem is the attempt to create MeSH indexes in repositories with biomedical publications. This particular task, which improves the time and the quality of information retrieved from the repositories, is usually undertaken by field experts. However, this manual approach is a time consuming process (it takes two to three months to incorporate new articles), but also a costly one (the cost for each article is approximately $10) [10].

3 Related Work

A plethora of research approaches have attempted to tackle with the problem of indexing. To do this, they use word embeddings in combination with classifiers. Typical cases are discussed below.

MeSH Now [10] classifies the candidate terms based on the relevance of the target-article and selects the one with the highest ranking, thus achieving a 0.61 F-score. To do this, researchers are using k-NN and Support Vector Machine (SVM) algorithms. Another approach, named DeepMeSH [13], deals with two challenges, that is, it attempts to examine both the frequency characteristics of the MeSH tags and the semantics of the references themselves (citations). For the first it proposes a deep semantic representation called D2V-TFIDF and for the latter a classification framework. A k-NN classifier is used to rate the candidate MeSH headings. This system achieves an F-score of 0.63. Another study [5] uses the Word2Vec algorithm on all the abstracts of the PubMed repository, thereby generating a complete dictionary of 1,701,632 unique words. The use of these vectors as a method to reduce dimensionality is examined by allowing greater scaling in hierarchical text classification algorithms, such as k-NN. By selecting a skip-gram neural network model (FastText) vectors of size 300 are generated with different windows from 2 to 25, which the authors call MeSH-gram with an F-score of 0.64 [1].

Moreover, taking into consideration the assumption that similar documents are classified under similar MeSH terms, the cosine similarity metric and a representation of the thesaurus as a graph database, scientists proceed to an implementation with an F-score of 0.69 [16]. A problem-solving approach is considered starting from converting texts into vectors with the use of Elastic Search and identifying the most similar texts with the help of the cosine similarity metric. Then, by deriving the tags from these texts and calculating the frequency of occurrence in conjunction with similarity, an evaluation function is defined which classifies documents.

BioWordVec [22] is an open set of biomedical word vectors/embeddings which combines subword information from unlabeled biomedical text with MeSH. There are two steps in this method: first, constructing MeSH term graph based on its RDF data and sampling the MeSH term sequences and, second, employing the FastText subword embedding model to learn the distributed word embeddings based on text sequences and MeSH term sequences. In this way, the value of the F-score metric is improved to 0.69 and 0.72 for CNN and RNN models respectively.

4 Methodology

4.1 Automated Indexing Procedure

The proposed approach for the indexing of biomedical resources starts with assembling the datasets to be used for training. We then proceed by evaluating and reporting on two prominent embedding algorithms, namely Doc2Vec and ELMo. The models constructed with these algorithms, once trained, can be used to suggest thematic classification terms from the MeSH vocabulary.

In an earlier work we have shown how to glean together resources from various open repositories, including biomedical ones, in a federated manner. User query terms can be reverse-engineered to provide additional MeSH recommendations, based on query expansion [7]. Finally, we can combine and assess these semantic recommendations by virtue of a trained embeddings model [6].

Each item's metadata is scraped for the title and abstract of the item (*body of text*). This body of text is next fed into the model and its vector similarity score is computed against the list of MeSH terms available in the vocabulary. Training datasets comprise biomedical literature from open access repositories including PubMed [19], EuropePMC [3] and ClinicalTrials [17] along with their handpicked MeSH terms. For those terms that may not occur at all within the datasets, we fall back to their scopeNote annotations within the MeSH ontology. As a result, the model comes up with a set of suggestions together with their similarity score (Fig. 1).

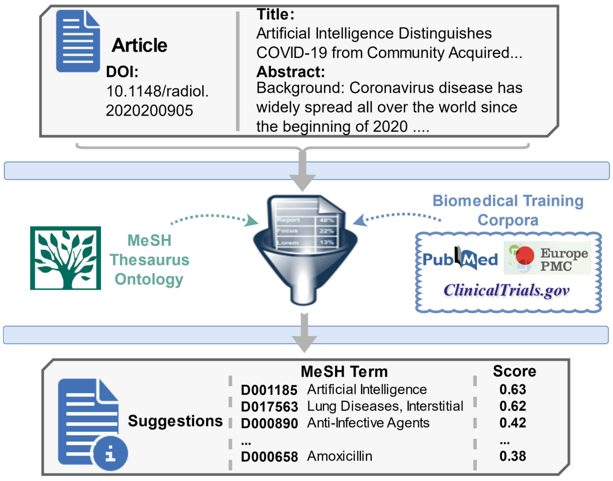
Article
DOI:
10.1148/radiol.
2020200905
Title:
Artificial Intelligence Distinguishes
COVID-19 from Community Acquired...
Abstract:
Background: Coronavirus disease has
widely spread all over the world since
the beginning of 2020 ....
MeSH
Thesaurus
Ontology
Biomedical Training
Corpora
PubMed
Europe PMC
ClinicalTrials.gov
Suggestions
MeSH Term
Score
D001185 Artificial Intelligence 0.63
D017563 Lung Diseases, Interstitial 0.62
D000890 Anti-Infective Agents 0.42
...
D000658 Amoxicillin 0.38

Fig. 1. A specific item gets subject annotations and their similarity scores are computed.

4.2 Datasets

For the application of the Doc2Vec and ELMo methods, a dataset from the PubMed repository with records of biomedical citations and abstracts was used. In December of every year, the core PubMed dataset integrates any updates that have occurred in the field. Each day, the National Library of Medicine produces updated files that include new, revised and deleted citations. About 30 M records, which are collected annually, can be accessed by researchers as of December 2018. Another source is EuropePMC, which is a European-based database that mirrors PubMed abstracts but also provides free access to full-texts and an additional 5 M other relevant resources.

Each entry in the dataset contains information, such as the title and abstract of the article, and the journal which the article was published in. It also includes a list of subject headings that follow the MeSH thesaurus. These headings are selected and inserted after manual reading of the publication by human indexers. Indexers typically select 10-12 MeSH terms to describe every indexed paper. MeSH is a specialized solution for achieving a uniform and consistent indexing of biomedical literature. In addition, it has already been implemented in SKOS [20]. It is a large and dense thesaurus consisting of 23,883 concepts.

The ClinicalTrials repository also provides medical data, that is, records with scientific research studies in XML. ClinicalTrials contains a total of 316,342 records and each record is also MeSH indexed. We include these records with the ones obtained from PubMed and EuropePMC for the purposes of variability and dataset diversity.

4.3 Using a Distributed File System

Our initial methodology for collecting data for training followed a serial approach [6]. Access to PubMed can be done easily via File Transfer Protocol (FTP). The baseline folder includes 972 zip files, up to a certain date (December 2018). Each file is managed individually with the information being serially extracted, thus rendering the completion of the entire effort a time costly process. In addition to delays, this problem makes the entire process susceptible to the need for constant internet connection to the repositories and any interruptions that may occur.

Algorithm 1: Dataset preparation procedure

```
         input: XML files from repository
         output: two CSV files
Step 1.  for each file ∈ repository do
Step 2.      connect to FTP server
Step 3.      get file to local disk
Step 4.      store file as line in RDD
Step 5.      delete file from local disk
Step 6.  end for
Step 7.  parse file - useful information is extracted
Step 8.  convert RDD to DataFrame
Step 9.  write useful information to CSV files
```

To solve this particular problem, we investigate the use of a distributed infrastructure at the initial stage of data collection. For this purpose, Apache Spark[1], a framework for parallel data management, was used. XML files are now stored as a whole in a DataFrame. In detail, all the information in an XML file is read as a line and then converted into Resilient Distributed Dataset (RDD). The useful information is then extracted as in the

[1] https://spark.apache.org/.

previous procedure. Finally, the RDD is converted into a DataFrame, from which the information is easily extracted, for example, into CSV files. With this process, although extracting data from repositories is still not avoided, parsing can be now performed on a distributed infrastructure.

4.4 Balancing

An essential part of the dataset preparation process is to cover the whole thesaurus as thoroughly as possible. Thus, apart from the full coverage of the thesaurus terms, the model must also learn each term with an adequate number of examples (term annotations) and this number should be similar between the terms. Otherwise, there will be a bias towards specific terms that happen to have several training samples vs. others which might have only a few. Therefore, to achieve a balanced dataset for training, when a term is incorporated into the set, its number of samples is restricted by an upper limit (Algorithm 2). If there are fewer samples the term is ignored; if there are more, exceeding annotations are cut off. The final dataset, as shown in Table 1, does not fully cover the thesaurus, but the terms it contains are represented by an adequate and uniform number of samples.

Algorithm 2: Dataset balancing procedure

```
         input: two CSV files, thesaurus terms
         output: two CSV files with balanced dataset
Step 1.  for each term ∈ thesaurus do
Step 2.      compute number of term occurrences in CSV file
Step 3.      if number >= 100 then
Step 4.          collect first 100 samples
Step 5.          add samples in two lists
Step 6.      end if
Step 7.  end for
Step 8.  write lists to CSV files
```

Table 1. Details of dataset

	Balanced dataset	Test dataeset
Total items	100,000	10,000
Total annotations	140,000	14,000
Average # terms per item	1.4	1.4
Thesaurus terms coverage rate (%)	4%	4%

4.5 Experiments

Doc2Vec Model. To create the model, the input is formed from the "one-hot" vectors of the fixed-size body of text, which is equal to the dictionary size. The hidden plane includes 100 nodes with linear triggering functions and with the same size as the resulting vector dimensions. At the output level there will also be a vector equal to the dictionary size, while the activation function will be *softmax*.

The training of the Doc2Vec model, with the help of the Gensim library[2], is performed by using the following parameters: *train epochs* 100, *size vector* 100, *learning parameter* 0.025 and *min count* 10. A variety of tests were performed to estimate these values. Tests have shown that when there are only a few samples per term, a larger number of epochs can compensate for the sparsity of training samples. In addition, removing words with less than 10 occurrences also creates better and faster vector representations for thesaurus terms. The created model is stored so that it can be called directly when needed. This model, with the adopted weights of the synapses that have emerged, is in fact nothing more than a dictionary. The content of this dictionary is the set of words used in the training along with their vector representations as well as a vector representation for each complete body of text.

ELMo Model. Initially, all the vectors are extracted through a URL connection[3]. Then, all the words related to the topic and mentioned in labels, are converted into classes, that is, numeric values, e.g. 0, 1, 2, etc., which in turn become "one-hot" vectors. To create the model, we have used the Keras library[4]. The input layer receives one body of text (title and abstract) at a time. The next layer is the *Lambda*, which is supplied by the input layer, and uses the ELMo embeddings, with an output count of 1024. To create the ELMo embeddings, the vectors derived from the URL are used and the full body of text is converted into string format and compressed at the same time with the help of the tensorflow package. Selecting the default parameter in this process ensures that the vector representation of the body of text will be the average of the word vectors. The next layer will be the dense one, which in turn is powered by the Lambda layer and contains 256 nodes and the *ReLU* activation function. Finally, we have the output layer, which is also a dense layer with as many nodes as classes, and in this case the activation function will be *softmax*. In the final stage of the system (compile), the loss parameter is *categorical_crossentropy*, because of the categorization being attempted. The *ADAM* optimization, and the accuracy metric are chosen.

The training of the ELMo model is done by starting a tensorflow session with parameters: *epochs* 10 and *batch_size* 10. This ensures that training will not take longer than 10 epochs and the data will be transferred in packages of 10 samples. Upon completion, the finalized weights are stored, for the purpose of immediately creating a model either for evaluation or any other process required.

[2] https://radimrehurek.com/gensim/.
[3] https://tfhub.dev/google/elmo/3.
[4] https://keras.io/.

4.6 Scalability

In the context of the implementation of the above two models, the most significant stage is the execution of the algorithm responsible for their training. This process, depending on the architecture of the model, is particularly demanding on computing resources.

Most experiments are conducted on average commodity hardware (Intel i7, 2.6 GHz, 4-cores CPU with 16 GB of RAM). In a shallow neural network, such as Doc2Vec, execution can be completed relatively quickly without significant processing power demands. Our test configuration appears sufficient for at least 100 epochs to be achieved, so that the model can be trained properly. However, this is not the case for deep neural networks, such as ELMo. In these models, the multiple layers with many connections add complexity and increase computational power requirements in order to be adequately trained. Therefore, finding a way to parallelize the whole process is considered necessary, not only for the optimization of the training time, but also for the completion of the entire effort.

Specifically, options associated with increasing the size of the training set in combination with the number of epochs can well lead either to a collapse of the algorithm execution or to prohibitive completion times. As an example, training ELMo on a dataset of 1,000 samples (10 labels with 100 items each) took about 20 *min* in the above hardware configuration.

Based on this concern, we have conducted experiments on infrastructures with a large number of Graphics Processing Units (GPUs). GPUs owe their speed to high bandwidth and generally to hardware which enables them to perform calculations at a much higher rate than conventional CPUs. We experimented on high-performance hardware with two Intel Xeon CPUs including 12-cores, 32 GB of RAM and a Nvidia V100 GPU. The V100 has 32 GB of memory and 5,120 cores and supports CUDA v. 10.1, an API that allows the parallel use of GPUs by machine learning algorithms. In this configuration, training with the 1,000 samples dataset took only 30 s for the same model. This faster execution (40x) can help run experiments with larger datasets that would otherwise be prohibitive to achieve.

5 Results

5.1 Doc2Vec

To evaluate the Doc2Vec approach, a total of 10,000 samples was used. This test set is balanced with the same procedure as the 100 K dataset we have used previously for training. Therefore, each MeSH label occurring in the test set appears within the annotations of 10 bibliographic items vs. 100 in the training set. In both sets, a total of 1,000 distinct MeSH labels, out of the available 22 K, are considered. A body of text (title and abstract) is given as input to the model, which in turn generates a vector. The model then searches through the set of vectors, already incorporated from training, to find those that are close to the generated one. The process is quite difficult due to the plethora of labels in total, but also individually per sample as each one contains a finite non-constant number of labels. The threshold value reported is the similarity score above which suggestions are considered. At most 10 suggestions are produced by the model.

The following Fig. 2 plots precision (P) and recall (R) for various threshold values. For our purposes, a single information need is defined as a single-term match with an item (1-1) and we report the mean value of these metrics over all these matches, i.e. they are micro-averaged. Precision takes into account how many of these 1-1 suggestions are correct out of the total suggestions made, while recall considers correct suggestions out of the total number of suggestions contained in the ground truth.

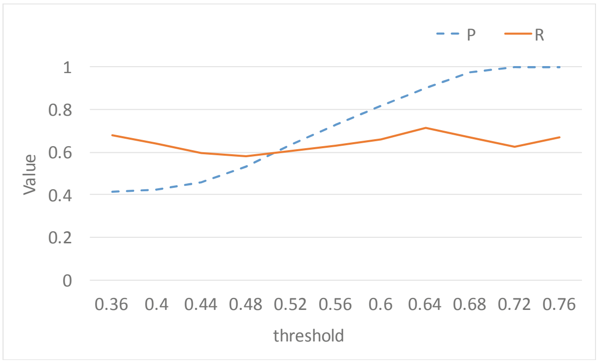

Fig. 2. Doc2Vec results

Other than the standard tradeoff between precision and recall, we first notice that the higher the threshold, the better the quality of results is for P. However, an increase in the threshold causes recall to drop. This makes sense, because there may be considerably fewer than 10 suggestions for higher thresholds, i.e. only few of the terms pass the similarity threshold, thus leaving out some relevant terms. For middle threshold values, more that 60% of predictions are correct and also cover over 60% of the ground truth.

5.2 ELMo

Similarly balanced test sets were also used to evaluate the ELMo model, with a varying number of labels and samples per label. The results obtained are shown in Fig. 3.

Initially, equally comparable or better results to previous related work are observed for precision and recall, when a small number of labels is selected. Given that the dataset is balanced, these considerably improve when allowing 10x more samples for each label, thus allowing for better embeddings to be learned by the model. Raising the number of labels to 100 increases the dimensionality of the problem; now, each item must be classified among 100 classes rather than simply 10. Consequently, we notice a decrease in both metrics which, however, is ameliorated with the use of more samples (1,000 per label), as expected. Nonetheless, further increasing the complexity of the dataset, by allowing 1,000 labels, causes absolute values to decrease considerably. Even with the better performing hardware, any further efforts to improve the results in the case of 1,000 labels, notably by increasing the number of samples, do not succeed, as the execution of the algorithm is interrupted each time due to the need for additional memory.

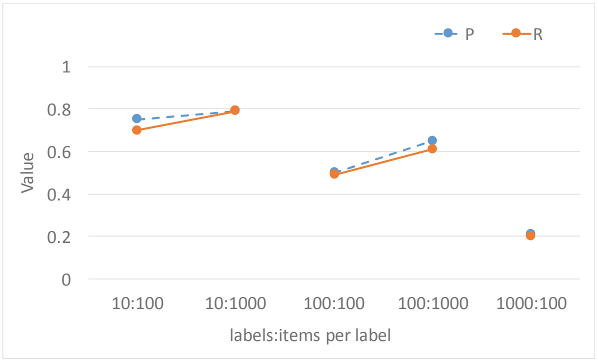

Fig. 3. ELMo results

Since our ELMo-based model performs multi-class classification (but not multi-label), micro-averaged precision and recall would be identical[5]. Now, because our test set is balanced, both micro- and macro-averaged recall will also be equal (the denominator is the fixed number of samples per label). Figure 3 plots *macro-averaged* precision i.e. averages precision equally for each class/label. This macro-averaged precision will always be higher than the micro averaged one, thus explaining why *P* appears better than *R*, but only slightly, because of the absence of imbalance. *P* and *R* are virtually the same.

5.3 Discussion

The Doc2Vec model is capable of making matching suggestions on its own, by employing similarity scores. A threshold value of 0.64 is where precision and recall acquire their maximum values. Certainly, fewer than 10 suggestions may pass this limit, but these are more accurate, apparently because of their higher score; they also occur more frequently within the terms suggested by experts in the ground truth: on average, each biomedical item in the test set hardly contains 2 MeSH terms, let alone 10 (see Table 1). This observation serves as validation of the fact that the similarity measure produced by the model is highly relevant and correlated with the quality of suggestions.

The ELMo model, in turn, has allowed us to construct a multi-class classification pipeline that is built around the ELMo embeddings. This manages to achieve very good results when dimensionality is kept low, i.e. a relatively small number of labels or classes is selected. It also seems to outperform the Doc2Vec approach, even though classes are fewer and the recommendation problem is reduced to multi-class classification. However, Doc2Vec surpasses the ELMo classification pipeline when there is a need to choose labels from a broader space. One the other hand, the threshold existence favors Doc2Vec, in the sense that fewer ground truth annotations pass its mark and are, therefore, considered when computing retrieval metrics. Further attempts for improvement where not possible for ELMo, a fact that confirms it is a very computationally expensive module compared to word embedding modules that only perform embedding lookups.

[5] https://scikit-learn.org/stable/modules/model_evaluation.html Sect. 3.3.2.8.2.

6 Conclusions and Future Work

Contextualized word representations have revolutionized the way traditional NLP used to operate and perform. Neural networks and deep learning techniques combined with evolving hardware configurations can offer efficient solutions to text processing tasks that would be otherwise impossible to perform on a large scale. We have shown that word embeddings can be a critical part and deserve careful consideration when approaching the problem of automated text indexing. Especially in the biomedical domain, the shifting nature of research trends and the complexity of authoritative controlled vocabularies still pose challenges for fully automated classification of biomedical literature.

To this end, we have investigated both deep- and shallow learning approaches and attempted comparison in terms of performance. Dimensionality reduction, either implied by setting a threshold or directly posing a hard cap over available classification choices, appears necessary for automated recommendations to be feasible and of any practical value. Still, a careful dataset balancing as well as the capability of deep networks to leverage distributed GPU architectures are demonstrated beneficial and should be exercised whenever possible.

As a next step, we intend to further evaluate our ELMo implementation and design a model that would perform multi-label classification using the ELMo embeddings. In addition, we are planning to release a web-based service offering access to the recommendations provided by the two models. This would facilitate interoperability, for example, with learning management systems and other repositories. Finally, we see improvements in the way classification suggestions are being offered, especially in view of the density of the thesaurus used: other ontological relations, such as generalization or specialization of concepts can be taken into account in order to discover and prune hierarchy trees appearing in the recommendations list.

References

1. Abdeddaïm, S., Vimard, S., Soualmia, L.F.: The MeSH-gram Neural Network Model: Extending Word Embedding Vectors with MeSH Concepts for UMLS Semantic Similarity and Relatedness in the Biomedical Domain, arXiv:1812.02309v1 [cs.CL] (2018)
2. Devlin, J., Chang, M. W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, arXiv:04805v2 [cs.CL] (2019)
3. Europe PMC Consortium. Metadata of all Full-Text Europe PMC articles. europepmc.org/ftp/pmclitemetadata/
4. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of Tricks for Efficient Text Classification, arXiv:1607.01759v3 [cs.CL] (2016)
5. Kosmopoulos, A., Androutsopoulos, I., Paliouras, G.: Biomedical semantic indexing using dense word vectors. In: BioASQ (2015)
6. Koutsomitropoulos, D., Andriopoulos, A., Likothanassis, S.: Subject classification of learning resources using word embeddings and semantic thesauri. In: IEEE Innovations in Intelligent Systems and Applications (INISTA), Sofia, Bulgaria (2019)
7. Koutsomitropoulos, D.: Semantic annotation and harvesting of federated scholarly data using ontologies. Digit. Libr. Perspect. **35**(3–4), 157–171 (2019)
8. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: 31st International Conference on Machine Learning, ICML, Beijing, China (2014)

9. Li, Y., Yang, T.: Word embedding for understanding natural language: a survey. In: Srinivasan, S. (ed.) Guide to Big Data Applications. SBD, vol. 26, pp. 83–104. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-53817-4_4
10. Mao, Y., Lu, Z.: MeSH now: automatic MeSH indexing at PubMed scale via learning to rank. J. Biomed. Semant. **8**(1), 15 (2017). https://doi.org/10.1186/s13326-017-0123-3
11. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: ICLR Workshop (2013)
12. Mork, J.G., Jimeno-Yepes, A., Aronson, A.R.: The NLM medical text indexer system for indexing biomedical literature. In: Conference and Labs of the Evaluation Forum 2013 (CLEF 2013), Valencia, Spain (2013)
13. Peng, S., You, R., Wang, H., Zhai, C., Mamitsuka, H., Zhu, S.: DeepMeSH: deep semantic representation for improving large-scale MeSH indexing. Bioinform. **32**(12), i70–i79 (2016). https://doi.org/10.1093/bioinformatics/btw294
14. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, pp. 1532–1543 (2014)
15. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations, arXiv:1802.05365v2 [cs.CL], NAACL (2018)
16. Segura, B., Martínez, P., Carruan, M.A.: Search and graph database technologies for biomedical semantic Indexing: experimental analysis. JMIR Med. Inform. **5**(4), e48 (2017). https://doi.org/10.2196/medinform.7059
17. U.S. National Library of Medicine. ClinicalTrials.gov. https://clinicaltrials.gov
18. U.S. National Library of Medicine. Medical Subject Headings, 2019. https://www.nlm.nih.gov/mesh/meshhome.html
19. U.S. National Library of Medicine. PubMed.gov. https://www.nlm.nih.gov/databases/download/pubmed_medline.html
20. van Assem, M., Malaisé, V., Miles, A., Schreiber, G.: A method to convert thesauri to SKOS. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 95–109. Springer, Heidelberg (2006). https://doi.org/10.1007/11762256_10
21. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 6000–6010 (2017)
22. Zhang, Y., Chen, Q., Yang, Z., et al.: BioWordVec, improving biomedical word embeddings with subword information and MeSH. Sci. Data **6**, 52 (2019). https://doi.org/10.1038/s41597-019-0055-0