BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **XX**, No X

Sofia • 201X

Semantic Classification and Indexing of Open Educational Resources with Word Embeddings and Ontologies

Dimitrios A. Koutsomitropoulos, Andreas D. Andriopoulos, Spiridon D. Likothanassis

Computer Engineering and Informatics Dpt. University of Patras, Patra, Greece
Emails: koutsomi@ceid.upatras.gr a.andriopoulos@upatras.gr likothan@ceid.upatras.gr

Abstract: *The problem of thematic indexing of Open Educational Resources (OERs) is often a time-consuming and costly manual task, relying on expert knowledge. In addition, a lot of online resources may be poorly annotated with arbitrary, ad-hoc keywords instead of standard, controlled vocabularies, a fact that stretches up the search space and hampers interoperability. In this paper, we propose an approach that facilitates curators and instructors to thematically annotate educational content. To achieve this, we combine explicit knowledge graph representations with vector-based learning of formal thesaurus terms. We apply this technique in the domain of biomedical literature and show that it is possible to produce a reasonable set of thematic suggestions which exceed a certain similarity threshold. Our method yields acceptable levels for precision and recall against corpora already indexed by human experts. Ordering of recommendations is significant and this approach can also have satisfactory results for the ranking problem. However, traditional IR metrics may not be adequate due to semantic relations amongst recommended terms being underutilized.*

Keywords: *learning objects, OERs, classification, word embeddings, thesauri, ontologies, doc2vec, federated search, MeSH*

1. Introduction

Open Educational Resources (OERs) are becoming turnkey learning objects (LOs) during the last few years and their massive availability is rapidly adopting characteristics of Big Data [1]. As the volume and complexity of OERs grows the task of discovering the most appropriate resources to glean together for e-learning purposes becomes cumbersome and error-prone. To this end, the use of knowledge

organization systems such as thesauri maintained by independent bodies and institutions is well-acknowledged, but not as frequently implemented. However, adequate content characterization using authority vocabularies requires elaborate, time-consuming, manual efforts, often involving field experts. Even when such thesauri are used, efficient and accurate content indexing is currently an open research problem, especially in rich and complex fields like biomedical literature [2].

In earlier work we have shown that it is possible to harvest OERs from disparate providers in a federated manner and to fetch a least common subset of metadata elements based on a Learning Object Metadata (LOM) schema [3]. User keywords are matched against well-structured thematic thesauri, expressed in the Web Ontology Language (OWL) and then expanded based on their structure/semantic relations to boost recall of the search process. Selected resources can be kept in a local institutional infrastructure known as the Learning Object Ontology Repository (LOOR) for further reuse [4].

In this paper, we propose an approach that can help the LOOR users, that is, curators and instructors to thematically annotate educational content. First, we reuse discovered thesaurus terms to thematically annotate selected OERs. Second, we further verify and amend these semantic matches with additional thematic suggestions coming from a machine learning process that employs the doc2vec algorithm: thesauri terms tagging OERs are automatically learned using word embeddings of their title and abstract. By combining these two approaches we demonstrate that it is possible to produce a reasonable set of thematic suggestions which exceed a certain similarity threshold. The added benefit of the collaboration between the logical formalism of web ontologies and non-symbolic inference for subject classification of OERs lies in the heart of this contribution and, to our knowledge, it has been seldom investigated before.

Our prototype is applied and tested by considering biomedical literature as OERs that we harvest from sources such as PubMed [5] and MERLOT [6]. However, the proposed approach can be applied to any other domain for which a structured thematic vocabulary exists, like what Medical Subject Headings (MeSH) is for the biomedical domain [7]. By reusing seed keywords and assigning subject labels to OERs, our method also achieves to tackle with the open problem of semantic classification or indexing of resources. Such educated subject suggestions would be finally amendable by instructors; therefore, their order of presentation is significant. It is shown that our approach can also improve on the ranking problem, putting relevant terms closer to the top of the list.

This paper is an extended version of work published in [8]. We extend our previous work with new results, more thorough description of the research contribution, including a comparison to related work closest to our approach, an extended training and test set, a larger scale validation, as well as new experiments and metrics about retrieval effectiveness and ranking performance of term recommendations.

In the following, we first review related work in the field of Natural Language Processing (NLP) and machine learning methods for text classification and examine advancements in our work relative to the state-of-the-art. Next, in section 3 we

present our methodology and architectural details for federated search and subject classification. Section 4 describes the design of our experiments, datasets and thesauri used for evaluation, as well as the baseline for comparison. A procedure for dataset construction and preparation is introduced. Section 5 discusses results in terms of average similarity scores, precision, recall and ranking effectiveness by testing our approach on a medical dataset using MeSH and comparing against manually recommended terms. Our conclusions and future work are summarized in the last section 6. Detailed evaluation scores are given in the Appendix.

2. Background and Related Work

2.1. Word embeddings

To represent text data, the word embeddings technique can be used: words and phrases are mapped into vectors of real numbers. This encoding may be simple, e.g. every word from a sentence is represented by a different number. A more intelligent approach, known as word2vec, was proposed by Mikolov [9]. In this method, vectors come as a result of training a shallow neural network, and it is possible to examine syntactic and semantic similarities by vector comparison. In a similar manner, doc2vec computes vectors for entire documents or paragraphs rather than mere keywords [10]. Related studies use word embedding techniques independently, as well as in combination with others in various text classification tasks. These include word2vec [11][12][13], GloVe [14], GloVe, word2vec and fastText [15], sentiment analysis with word2vec and fastText [16], information retrieval using a linear classifier and word2vec from a large body of text [17], even using word2vec with Principal Component Analysis (PCA) [18].

2.2. Topic and entity recognition

A demanding task is to assign a subject to a collection of OERs. So far, there have been relevant studies which, with the use of word embeddings and the PageRank algorithm [19][20][21], present a framework for automatic extraction and ranking of keywords. Also, another study extends word embedding models and employs the simple k-Nearest Neighbor (k-NN) search to predict tags for unseen documents [22]. Similarly to our work, these methods use doc2vec word embeddings and associate them with tags. However, they are mostly applied for generic topic recognition over web sources and do not employ any formal thesaurus or vocabulary.

The named entity recognition problem is related to subject assignment. Particularly, in biomedical studies, the ELMo model [23] which manages to improve the detection, for instance of ex genes, is utilized achieving high F-score values provided that the model is trained with biomedical data from repositories such as PubMed [24]. Finally, a new version of an NCBO Ontology Recommender 2.0 system which provides high quality recommendations in biomedical data, is proposed [25]. It extends the original version, which is a service that receives a biomedical text corpus or list of keywords and suggests ontologies (terms), according to new criteria. The criteria are four predefined questions which need to be answered.

2.3. MeSH indexing

A considerable part of relevant literature deals with the task of MeSH indexing in biomedical data. MeSH indexing is the task of assigning relevant MeSH terms based on a manual reading of scholarly publications by human indexers. Not only is this manual approach a time-consuming process (it takes 2-3 months for new articles to be incorporated) but also a costly one (approximately $10 per article) [2]. The main tools for implementing this task are machine learning classifiers, used on a case-by-case basis.

Specifically, a related system called *MeSH Now* [2] classifies candidate terms based on their relevance to the target article and selects the one with the highest ranking, thus achieving a 0.61 F-score. To achieve this, the authors use k-NN and Support Vector Machine (SVM) algorithms.

Another system is called *DeepMeSH* [26] and attempts to consider both the frequency characteristics of MeSH labels, as well as the semantics and ambiguity of the citations themselves. For the former, a deep semantic representation, called D2V-TFIDF, is proposed and combines features both from doc2vec and tf-idf. The latter is solved by using a learning to rank framework. A k-NN classifier is used to score the candidate MeSH headings. This system achieves an F-score of 0.63.

Another study [27] uses word2vec and applies it to all of the abstracts of the PubMed repository. It also considers the use of these vectors as a method to reduce dimensionality by allowing greater scalability for hierarchical text classification algorithms such as k-NN. Selecting a skip-gram neural network model (fasttext) produces 300-sized vectors with various windows from 2 to 25 which researchers call MeSH-gram and which yields an F-score of 0.64 [28].

Taking into account the assumption that similar documents are classified with similar MeSH terms, authors in [29] have proceeded to an implementation with an F-score of 0.69. This work starts with the conversion of documents into vectors by search engine indexing (Elastic Search) and the identification of the most similar documents based on cosine similarity. Then, by extracting the terms from these documents and by calculating their occurrence frequency in conjunction with their similarity, the authors define a scoring function which ranks these MeSH terms. Finally, a graph database of the MeSH thesaurus is used to discover hierarchical relationships among the terms. Given an unknown text, items from the text corpus that are most similar are found and only their terms are retrieved. These terms have already been assigned by experts. Thus, candidate terms come from a limited pool of recommendations. However, in our approach suggestions are sought within the entire set of MeSH terms and the model is responsible for identifying any potential structural and semantic similarities with the source text.

2.4. Discussion

To the best of our knowledge, the combination of word embedding techniques, in particular doc2vec, with ontology-based semantic matching and expansion for subject classification of OERs has not been proposed before.

Several of the related studies ([11][12][13][15][16][17][18]) attempt and eventually classify the data uniquely into a finite and quite small number of

categories, also possessing a large number of samples per category. In the present study, however, the subject assignment is done with the help of a thesaurus, which contains several thousands of labels. Furthermore, each sample is categorized by assigning more than one label to it (multi-label classification).

Also, most research that deals with the classification problem of a large number of classes ([2][24][25][26][27][28][29]) appears to rely on machine learning classifiers or deep neural networks. In our work, each thesaurus term has been moved to a vector space after training a doc2vec model with some corpus and using MeSH headings as tags. Once the model is trained, this allows us to quickly query the model for existing tags and perform feature extraction in unknown texts. Consequently, classification and indexing of items can be achieved by retrieving the most similar terms stored by the model, without the need and overhead of a separate classifier.

3. Design and Methodology

3.1. Federated Search

To address metadata incompatibilities among OER repositories, the creation and maintenance of a semantics-aware Learning Object Ontology Repository (LOOR) has been proposed in [4]. Such a repository can tap into ontologies and thesauri and allow LO metadata instances to be assigned machine-understandable semantic annotations. Building on this premise, LOs can be ingested and different schemata can be aligned within the LOOR into a common LO ontology. An outline of this ontology is shown in Fig. 1, combining terminology from the LOM standard and Dublin Core.

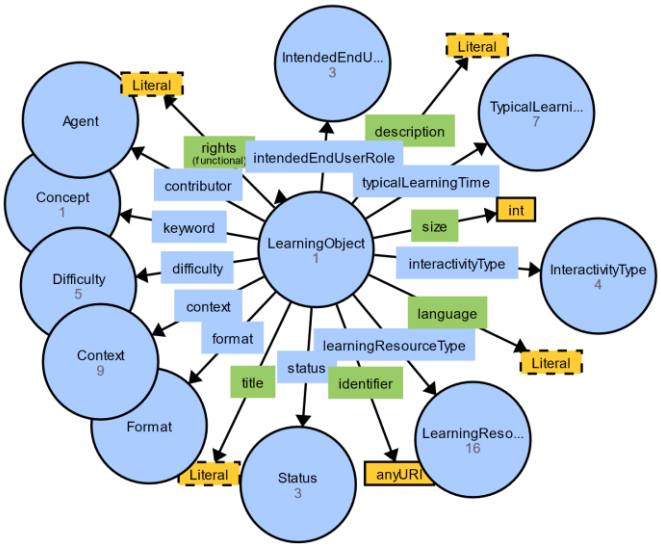

Fig. 1. Visualization of the LO Ontology Schema classes and properties.

First, a federated query is initiated towards the various repositories. Next, the metadata of items returned as responses to the query are harvested and aligned to a

unified LO Ontology Schema, using a common set of elements and mapping rules. The subject of these items is then automatically populated taking as basis the initial query keywords in accordance with thematic thesauri for specific knowledge domains. At this point, a curator or instructor may review the LO and decide to incorporate it into the LO ontology, thus making it available for others to reuse.

Currently, data sources include MERLOT II, a large archive of OERs [6], Europe PubMed Central, a major repository of biomedical literature [5], ARIADNE finder, a European infrastructure for accessing and sharing learning resources [30] and openarchives.gr, the entry point for Greek scholarly content.

3.2. Match SKOS Thesauri

The purpose of query expansion when harvesting OERs has been investigated before by the authors [3]. In essence, keywords that initiate harvesting are matched against expert terminological knowledge expressed in the form of term thesauri following the SKOS model in OWL format [31]. Each keyword can thus be expanded into several narrower keywords which refine the former by performing reasoning about the semantic relationships of the matching terms in the thesaurus hierarchy. For example, the SKOS relations *skos:broader / skos:narrower* are used to connect a concept with its refinements. Further, a thesaurus term can be comprised of multiple lexical representations, including alternative labels and translations in different languages, as represented by the properties *skos:preflabel* and *skos:altlabel*.

In this paper, we maintain and reuse the information caused by the exploration of the thesauri term hierarchy in order to thematically annotate an item, when it is selected for addition into the LOOR. As a result, original search keywords are being repurposed to provide subject annotations for selected LOs. Merely suppling arbitrary keywords as subjects would not make much sense; rather, these keywords are first matched and refined against formal thematic thesauri and the matches are injected as semantic subject annotations into the selected OERs, using the *lom:keyword* property of the LO Ontology Schema.

As an example, consider the seed keyword “medicine”. This is matched by a homonym term in the thesaurus and refined, for example, by the term *pathology*. Results matching the various labels of *pathology* will be automatically classified with the concept *pathology*, as well as the concept *medicine*, since this is the topmost parent matching term for the initial keyword.

3.3. Word Embeddings

In our work we employ the doc2vec model, namely the Distributed Memory architecture, which achieves the prediction task of the next word in a context to create a dictionary, trained with material from the OER corpora. Our aim is to create a model that would learn as many terms as possible from a vocabulary that represents a formal domain of knowledge, i.e. a thesaurus. Then, given the title and the abstract of a learning resource, the model would be able to predict those thesaurus terms that most closely represent the subject of the resource. The title and abstract are the two annotations of the least common subset of elements stored in the LOOR with the richest semantic meaning about the resource. The other would be the resource itself

(i.e. the full text). Keywords are also capable of conveying semantics but, as discussed, they are optional, frequently ad-hoc and are not guaranteed to be precise or to reference standard vocabularies. Moreover, the doc2vec algorithm operates on entire paragraphs or phrases instead of words. However, proper classification keywords (thesaurus subject annotations) do exist in our training set and they are used to tag paragraphs during training.

The title together with the abstract from each OER form a single *body of text*. Next, we convert this body into a list of lowercase tokens, while words with length shorter than two and special symbols are removed. For our dataset of short titles and abstracts, we have seen that retaining stop words generally improves similarity scores. Every title and abstract text are now a list of words. In our training dataset, every OER has been already annotated and classified by experts using terms from an appropriate thesaurus, e.g. MeSH for the biomedical field. Each such term in the thesaurus has a unique ID, so we select this ID instead of text to deal with multiple lexical representations of a term. As a result, each body text (title and abstract) is tagged by one or more term IDs, one for each term that occurs as its subject annotation in the dataset.

During training, the vector of each thesaurus term appearing in the dataset embeds information related to the entire abstract and title, thus each body text will be assigned a tag from one or more of these terms. However, it is unlikely that every possible term of the thesaurus would occur in the dataset, so there is a chance that several terms might be missing from the learned dictionary. To compensate for this loss, information from the thesaurus document is also integrated into the training phase. In particular, the text of the term description is also used for training. In this way, almost complete coverage of the dictionary words is achieved.

The trained model can answer queries about the similarity of whole texts and words. In addition, given a text, it detects a predetermined number of related words, calculates their similarity and sorts them in descending order.

3.4. Integrating semantic annotations and word embeddings

Both approaches described previously work collaboratively to provide subject classification suggestions, following the process flow shown in Fig. 2. First, metadata about OERs are harvested from the remote repositories into the LOOR and mapped to the unifying ontology schema. Then, semantic subject annotations are injected into these metadata, based on the seed keywords and term matching and expansion within the thesaurus ontology (*Semantic Matching*).

Next, the title and abstract of each OER metadata are fed into the trained doc2vec model and are corresponded to a single vector (*Doc2Vec*). The subject terms which have already been injected are searched for in the model dictionary, using the term IDs. Terms not occurring in the dictionary are ignored. Based on the model's output, each term is assigned a similarity score, as a means to assess the quality of the term suggestion.

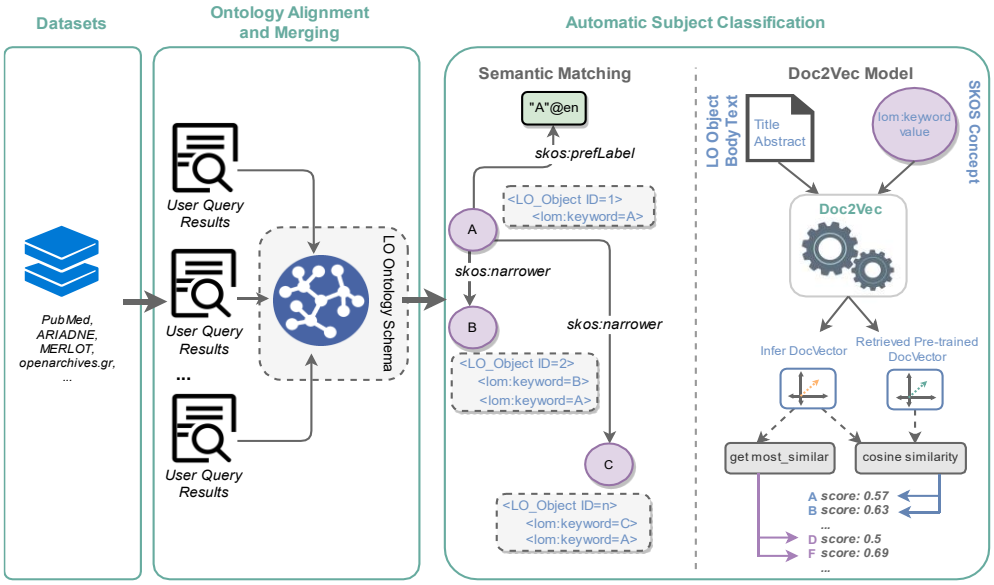

Fig. 2. Overall system process flow.

Furthermore, the trained model also seeks similarities between the OER and other thesaurus terms contained in the dictionary and outputs the top 10 terms with the highest similarity score. These terms can be further considered by a curator for inclusion when adding the OER into the LOOR. Naturally, the similarity of a proposed term to the body text of an item, no matter where it comes from (semantic matching or doc2vec itself), is a measure of the quality of this suggestion. Therefore, it might be useful to set a threshold above which suggestions are retained or discarded otherwise.

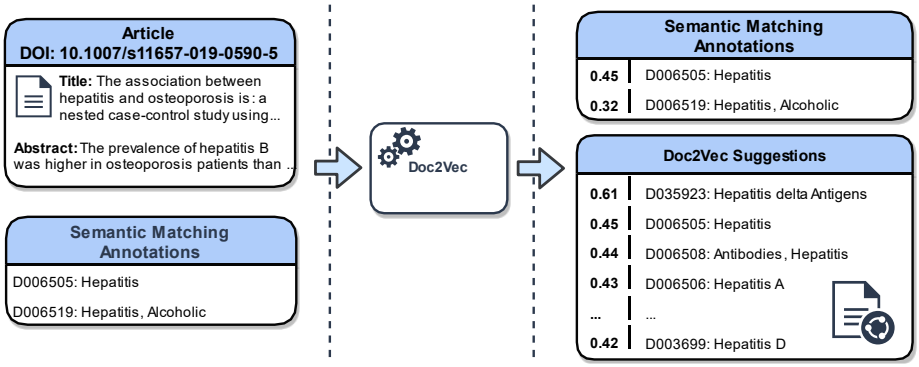

Fig. 3. A specific item gets subject annotations and their similarity scores are computed.

An example demonstrating the subject classification and scoring scheme discussed before is depicted in Fig. 3. A specific OER returned by federated search (https://doi.org/10.1007/s11657-019-0590-5) is annotated with two terms by the semantic matching process. Then, doc2vec computes their similarity scores and proposes another 10 subject terms along with their score.

4. Experimental Setup

4.1. Evaluation procedure

To evaluate our methodology, we initially conduct three representative experiments. First, we test the trained doc2vec model against part of the training set [32]. This is reasonable, since doc2vec can only perform well with texts and words already contained in its dictionary.

Second, we test the model with another, unknown test set and see how it performs when it is supplied with arbitrary titles and abstracts. In contrast to word2vec, doc2vec is capable of inferring vector representations of body texts not presented before to the model. The results of this second experiment are typical of a federated search scenario with arbitrary keyword seeds and, therefore, represent our baseline or threshold above which term suggestions can be retained.

Third, we evaluate the quality of the semantic matching suggestions by computing their average similarity. In addition, we present the average similarity score of the best suggestion made by doc2vec itself, for the purposes of comparison. The cosine similarity is computed for each sample of the test set used in the three experiments and the average performance of the model in all samples is reported.

Doc2vec training has been performed using the following parameters: *train epochs* 10, *size vector* 100, *learning parameter* 0.025, and *min count* 10. A variety of tests were performed to select these specific parameter values. Tests have shown that for a large number of samples even a small number of epochs, such as 10, is sufficient for the model to learn. Additionally, removing words with less than 10 occurrences also creates better and faster vector representations for thesaurus terms. The created model is saved so that it can be called directly when appropriate.

To verify the suitability of the similarity threshold discussed before and get an estimate of user satisfaction by the system's suggestions, we also evaluate as per two other dimensions: a) examine retrieval effectiveness in terms of precision and recall b) rank effectiveness, to investigate if relevant terms are positioned high enough in the suggestion list.

4.2. Dataset

For the application of the doc2vec method, a dataset from the PubMed repository with records of biomedical citations and abstracts was used[1]. 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 update files that include new, revised and deleted citations. About 30M records, which are collected annually, can be accessed by researchers as of December 2018.

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.

[1] https://www.nlm.nih.gov/databases/download/pubmed_medline.html

MeSH is a formal, specialized thematic thesaurus that gives uniformity and consistency to the indexing and cataloging of biomedical literature. MeSH has been already implemented using the SKOS vocabulary specification into OWL format [33]. It is a relatively large and dense thesaurus, comprising 23,883 SKOS concepts (thesaurus terms).

The data is available in XML format [34]. The elements finally used for doc2vec training are *ArticleTitle*, *AbstractText* that represent the body text; and, from the *MeshHeadingList*, the *DescriptorName* with *MajorTopicYN = "Y"* or the *DescriptorName* that includes at least one *QualifierName* with *MajorTopicYN = "Y"*. The information contained in the DescriptorName is a unique ID of the format e.g. D007069 and it represents the labels of the text (thesaurus terms). Fig. 4 shows an example of an XML record representing a literature item with all the elements to be used for training.

```
<?xml version="1.0" encoding="UTF-8"?>
<ArticleTitle>Demonstration of tumor inhibiting
properties...</ArticleTitle>
<Abstract>
  <AbstractText>A report is given on the recent discovery of outstanding
   immunological properties...
  </AbstractText>
</Abstract>
<MeshHeadingList>
  <MeshHeading>
    <DescriptorName UI="D007069"
MajorTopicYN="Y">Ifosfamide</DescriptorName>
    <QualifierName UI="Q000494"
MajorTopicYN="N">pharmacology</QualifierName>
  </MeshHeading>
  <MeshHeading>
    <DescriptorName UI="D007109"
MajorTopicYN="N">Immunity</DescriptorName>
    <QualifierName UI="Q000187" MajorTopicYN="Y">drug
effects</QualifierName>
  </MeshHeading>
  <MeshHeading>
    <DescriptorName UI="D007165"
MajorTopicYN="N">Immunosuppression</DescriptorName>
  </MeshHeading>
</MeshHeadingList>
```

Fig. 4. Sample XML record from the PubMed repository

For the initial set of experiments a training set of 155,963 samples (bibliographic items) has been used. These samples contain a total of 420,165 MeSH terms, i.e. an item may be annotated with multiple terms, while the unique terms are 11,686, which cover 49% of the total vocabulary of 23,883 words. For the sake of completeness, 11,883 additional terms were selected from the thesaurus file. The selection criterion is for these terms to have descriptions, specifically the *scopeNote* field, roughly representing a brief definition of the term. In total we have covered 99% of the dictionary since we have 23,569 unique terms. However, for these additional terms, the model is trained using only a single body text which is the contents of the scopeNote field; therefore, such terms may not be adequately learned yet.

Additionally, an even larger dataset was adopted to perform experiments for the evaluation of the model with precision and recall metrics. This set contains 1M samples and 16,782 unique MeSH terms, which cover 70% of the total vocabulary. Another 10K items, not used during training and therefore unknown to the model, have been reserved as a test set for our experiments. However, it may contain some terms beyond the coverage of the training set. Such terms will never be proposed, therefore precision and recall values reported would always represent lower bounds. The quantitative characteristics of the datasets used for training and testing are summarized in Table 1.

Table 1. Details of dataset

	Initial dataset	Extended dateset	Test set
Total items	155,963	1,000,000	13,470
Total annotations	420,165	3,208,301	41,374
Average # terms per item	3	3	3
Thesaurus Terms Coverage Rate (%)	49%+50%	70%	25%

4.3. Dataset preparation

Data with the biomedical citations and abstracts were obtained from the PubMed repository through FTP. In the PubMed baseline folder there are 972 zip files, up to December 2018. Each of these files is individually downloaded locally, unzipped, and sent for parsing using a Python script. In this process, the useful information is isolated and stored in two lists for each file. Specifically, the abstract is placed immediately after the title and is added to the first list, while all its respective tags are inserted into a second list. Lastly, the file is deleted from the local disk for space saving. The process is repeated for all available files (Algorithm 1). In order to make it easier to manage the files that are generated, the process stops per 100 files and from the lists two csv files are created, one from each list. When the process is completed, the data are ready for the next stage of pre-processing to be ready for the training phase.

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. parse file - useful information is isolated
Step 5. store useful information in two lists
Step 6. delete file from local disk
Step 7. **end for**
Step 8. write lists to CSV files

5. Experiments and Results

5.1. Similarity Scores

Initially, we select the doc2vec model which was derived from the word embedding process and to which unsupervised training was applied. Evaluation is carried out by checking the similarity of text sentences among a dataset of 15,383 items, a subset of the training set. Specifically, we create the vector of the title and abstract by supplying this body text as an argument to the model, while we draw the vector which already exists in the model's dictionary using the term ID. These two elements normally have a high degree of correlation. Then, through the metric of the cosine similarity between two vectors, we calculate the similarity between the two. Because each item can be annotated with multiple terms, the repetitions executed reach the amount of 46,693. Results are depicted in Fig. 5 (left). The mean and the standard deviation of the results are 0.43 and 0.12 respectively.

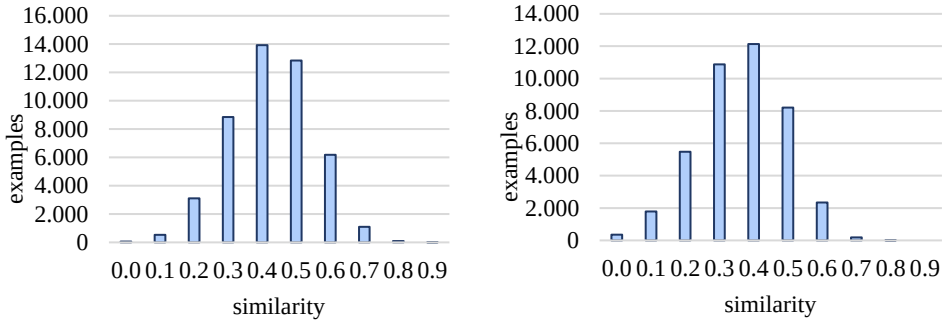

Fig. 5. Similarity distribution over a subset of the training set (left) and unseen text (right).

An additional test is performed to check the reliability of the model on a different dataset, unknown to the model. This dataset is comprised of another 13,470 items that have not been used for training. The terms that annotate these items are contained in the first 49% of the total vocabulary that has been learned from the PubMed training dataset. The total annotations count is 41,374. Results are of interest as, even with unseen body texts, the model responds satisfactorily regarding their similarity, as depicted in Fig. 5 (right). The mean and the standard deviation of the results are 0.36 and 0.12 respectively. We notice a slight drop in the average similarity score due to the dataset being unknown. However, we select this score as our threshold for term selection, considering the worst-case scenario where the model is oblivious to the body texts used as inputs.

Finally, another test is performed using a dataset of 1,405 items. These items have been specifically returned by the federated search procedure and their annotations are produced by Semantic Matching. The mean and the standard deviation of the average similarity scores for these annotations are 0.30 and 0.13 respectively (case 1). Out of 1,805 annotations, 596 (33%) pass the 0,36 threshold set before for arbitrary texts and can be ultimately retained when selecting the item for addition into the LOOR.

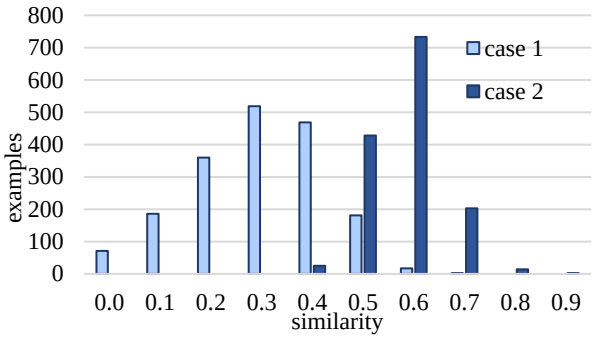

Fig. 6. Similarity distribution over semantic matching annotations (case 1) and doc2vec suggestions (case 2).

Next, we let the model produce its own suggestions asking for the top 10 terms with the highest similarity score (case 2). The best suggestion made has a mean of 0.58 and a standard deviation of 0.07 (case 2). The final results are depicted in Fig. 6. Additionally, the top 10 suggestions have a mean of 0.54 and a standard deviation of 0.07. In summary, the results of the above experiments are shown in Table 2.

A relatively increased similarity is noticed in comparison to all previous experiments. This is justified by the fact that the model is no longer limited to checking specific annotations. Now it is given the freedom to choose the most similar vector from a rather large repository of 23,569 unique terms contained in its dictionary. Therefore, on account of the reliability offered by the doc2vec model, as well as the data used for training, suggestions made by the model itself are determined by greater similarity.

Table 2. Experiment results for similarity scores

Metrics	Training subset	Test set (unseen)	Semantic matching annotations	doc2vec suggestions
Total annotations	46,693	41,374	1,805	1,405
Similarity Average	0.43	0.36	0.30	0.58
Similarity Stdev	0.12	0.12	0.13	0.07

5.2 Retrieval Effectiveness

Average similarity may be a relative performance measure but says little about the actual quality of the suggestions made by the system. To assess this, we also calculate precision and recall by comparing to the terms already inserted by the human indexers. These terms comprise the *ground truth* for these experiments and convey the meaning of the "gold standard" for user satisfaction, given that the system is intended for expert users (educators, instructors, curators). For this purpose, we consider a test set that is unknown to the model. We expect to derive conclusions about the suitability of the selected selection threshold (similarity > 0.36) and verify how close the system classification suggestions actually are to the experts' tags.

For precision, we check how many of the system proposed tags that exceed the threshold are correct, i.e. they coincide with expert tags.

$$P = \frac{Suggested\ terms\ \ that\ are\ relevant}{Suggested\ terms} \quad (1)$$

For recall, we measure how many system suggestions, again above the threshold, are contained within the list of expert tags, i.e. how many correct tags the system is able to retrieve.

$$R = \frac{Suggested\ terms\ that\ are\ relevant}{Relevant\ terms} \quad (2)$$

To appreciate the threshold level, we have conducted a set of experiments ranging over various values for the suggestion threshold (Appendix A).

The system's aim is to recommend appropriate thesaurus terms for educators to choose from and annotate items. They are presented in the order of their similarity score. Consequently, there is little point in assessing terms that come later on the list, since users are likely apt to consult only the first few recommendations and discard the rest.

To mitigate this effect, we report precision and recall at rank *k*: P@*k* is the precision achieved when examining only the first *k* terms suggested by the system, while R@*k* is the recall calculated in a similar fashion. Values reported represent mean values for precision and recall @*k*. This means that we treat the annotations of each item as a single information need and take the mean value of these metrics over all items of the test set, i.e. they are macroaveraged. Items in the test set for which there are no suggestions exceeding the specific threshold do not contribute negatively during metrics calculation.

As mentioned, in addition to the annotations proposed by semantic matching, the model proposes an additional of 10 terms at most, with the threshold acting as a hard cap. We measure P and R @1, @3 and @10. Rank 10 is of interest especially for recall, since it is usually the average of terms suggested by experts on the test set and coincides with the upper limit on the model's suggestions; rank 3 is important for precision, because it is the actual average of terms that are designated as major topics, which are the only ones considered for training.

It is important to note that it is impossible for precision to take maximum values unless the number of relevant terms equals or exceeds k. For example, if there is only one relevant term, even a perfect system could only achieve a p@3 of 0.33. For this reason, we also report *R-precision*, a metric that evaluates precision as an average of P@k_i over all test items *i*, where k_i is the number of relevant terms for *i*:

$$R\ precision = \frac{1}{n}\sum_{i=1}^{n} P@k_i \quad (3)$$

Vector inference for unknown texts by the model exhibits some degree of randomness, so the number of matching terms (relevant suggestions) may somewhat fluctuate between experiments but is statistically insignificant. The table below summarizes precision and recall at ranks 1, 3 and 10, as well as R-precision for 3 values of the threshold.

Table 3. Mean precision, recall and R-precision at different ranks and threshold values.

Threshold	# items with terms above threshold	Metric	Value	Metric	Value
0.36	13,470	P@1	0.30	R@1	0.12
0.48	11,887	P@1	0.32	R@1	0.13
0.60	2,379	P@1	0.46	R@1	0.21
0.36	13,470	P@3	0.19	R@3	0.22
0.48	11,880	P@3	0.19	R@3	0.21
0.60	2,411	P@3	0.17	R@3	0.23
0.36	13,469	P@10	0.10	R@10	0.36
0.48	11,895	P@10	0.07	R@10	0.27
0.60	2,374	P@10	0.05	R@10	0.23
0.36	13,470	R-Precision	0.21		
0.48	11,888	R-Precision	0.20		
0.60	2,375	R-Precision	0.23		

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 and R @1. Greater values for recall are met when allowing up to 10 suggestions (@10). 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. On the contrary, R@3 remains almost constant because this is the average number of terms actually contained in the test set.

Likewise, the threshold seems to have little effect for P@3, other than a small drop of 0.02. On average, 1 suggested term out of 3 would be relevant for each item more than 60% of the time. Again, for P@10, only few of the recommendations pass the higher values of the threshold. In addition, there is an average of 3 relevant terms per item, which justifies the low scores for P@10.

R-precision is not affected by the count of relevant items in the test set and appears slightly improved than, for example, P@3. It also avoids the detrimental effect noticed in P@10. Still, for higher threshold values, the terms recommended may be fewer than the actual relevant terms.

5.3 Ranking Effectiveness

In the context of the current evaluation, the ranking effectiveness of the model can be expressed via the mean average precision (MAP) [35]. To calculate this, we first measure the average precision for each item. The *average precision* (AP) of term suggestions for an item (body text) *s* can be expressed as follows:

$$AP_s = \frac{\sum_{t \in T_s} P@r(t)}{|T_s|} \quad (4)$$

Here, T_s denotes the set of suggested terms that are relevant for item *s*, based on the ground truth, *r(t)* denotes the rank of term *t* and $P@r(t)$ denotes the *precision at rank* r(t), i.e. the fraction of suggested items that are relevant up to the r(t)-th position that is:

(5) $$P@r(t) = \frac{\{relevant\ terms\ up\ to\ the\ r(t)\ th\ position\}}{r(t)}$$

Given that the number of our model suggestions is bounded, *r(t)* can range from one up to ten and can take any integer value in between, i.e. *r(t)* ∈ {1, 2, 3,…,10}. For example, if there are 3 relevant suggestions for an item *s*, and they are ranked at the first three positions of the suggestions list, then:

(6) $$AP_s = \frac{1 + 2/2 + 3/3}{3} = 1$$

If, however, there are 3 relevant suggestions and they are ranked at the first, second and sixth position respectively (other suggestions being irrelevant), then:

(7) $$AP_s = \frac{1 + 2/2 + 3/6}{3} = 0.83$$

Therefore, it can be seen that the closest to 1 the AP for an item is, the highest relevant terms are ranked by our model. The MAP on the test item set *S* is the mean of the AP for all items tested and can be computed as:

(8) $$MAP = \frac{1}{|S|}\sum_{s \in S} AP_s$$

The following table shows results for MAP@3 and MAP@10, by considering up to the third and up to the 10th recommended term respectively. MAP@1 would always equal 1, given that T_s only includes suggested terms that are relevant.

Table 4. MAP at different ranks and threshold values.

Threshold	# items with terms above threshold	Metric	Value
0.36	6,683	MAP@3	0.76
0.48	5,872	MAP@3	0.80
0.60	1,206	MAP@3	0.95
0.36	9,579	MAP@10	0.56
0.48	6,800	MAP@10	0.70
0.60	1,208	MAP@10	0.95

As discussed, the ranking of recommendations depends on their similarity score. Terms with higher similarity are positioned first in the list of suggestions. The results presented at Table 4 serve again as validation of the fact that the similarity measure produced by the model is highly relevant and correlated with the quality of suggestions. Therefore, it comes as no surprise that an increase in the threshold produces considerable increase in MAP at both ranks reported. When there are relevant terms recommended by the system, they are indeed placed in higher positions of the suggestion list most of the time. In addition, occurrence of relevant terms in early positions of the list becomes more frequent as the threshold increases.

For MAP@10 we look at the first 10 results suggested for each item instead of 3. For lower thresholds there will be suggestions for a greater number of items (there will be no items with zero suggestions due to threshold cut-off) , but they are of lower quality, i.e. some of them are not relevant. For MAP@3 and low thresholds there will

still be a lot of suggestions of lower quality, but this time they are cut-off by rank 3. This explains why MAP@10 would be less than MAP@3 for lower thresholds.

5.4 Discussion

Given the difficulty of the problem of assigning a subject to an unstructured body of text and, in fact, from a large vocabulary of unique terms, the results of the above experiments are considered satisfactory. Despite the scores being relatively not too high, exact values are not really critical. The model is able to assign similarity to annotations yielded by semantic matching and make its own suggestions with even greater certainty. Moreover, the definition of the 0.36 threshold will assist in selecting or rejecting suggestions.

It is also evident that the performance of the model depends greatly on the training set. We might have circumvented the dictionary sparsity by including thesaurus terms with their descriptions directly from the thesaurus document, but this is still just one annotation for each term. Better results could not be achieved by simply addressing an even larger dataset; rather, the latter needs to be broad enough to cover as many terms as possible and to contain adequate samples for each term. This is by no means straightforward, since literature tends to concentrate on limited sets of concepts during the years.

The increase of retrieval effectiveness @1 along with the threshold is an intuitively satisfying observation, since it confirms that the model is capable of making suggestions with a sensitivity directly depending on the specific threshold. In itself, this fact validates our overall approach and training methodology and suggests room for improvement in the characteristics of the training set (size, thesaurus coverage, balancing).

Ranking effectiveness solidifies with greater threshold values, as expected. Overall, these results confirm that work proposed in this paper can offer a plausible approach for the recommendation ranking problem, as the order of term suggestions affects the annotations finally made by the users: they tend to consider more (or even solely) results appearing first.

The inclusion of the thesaurus concept hierarchy in the training process and its consideration during evaluation can affect the effectiveness of the system on the positive side. For example, we have noticed cases where the system may miss some expert recommended terms, but suggests terms that are on the same concept hierarchy (*skos:broader*). Such suggestions are now deemed irrelevant because the thesaurus structure is not taken into account and human indexers tend to use only the most specific terms. Also, the system may propose additional terms that are semantically relevant, such as siblings, terms belonging in the same hierarchy tree and so on. Still, these are not exact matches and this semantic effect is not currently evaluated.

6. Conclusions and Future Directions

Subject classification of OERs is a highly involving task, as it depends on several parameters ranging from availability of resources to metadata incompatibilities to

intended OER use and synthesis. Reusing seed keywords can offer an alternative for missing or ad-hoc annotations; subject suggestions are authority controlled and refer to formal bodies of domain knowledge. In addition, these suggestions can be assessed through a threshold posed by computing similarity between thesaurus terms and OER metadata. Not only can the construction of word embeddings for these two validate subject annotations but it can also make additional proposals for thematic classification.

Further evaluation involving larger pre-indexed citations corpora indicates that our approach for similarity scoring and ranking of subject recommendations is reasonable. First, it is shown that the model is indeed capable of learning the semantic interpretation of MeSH terms by adjusting their vectors according to the documents fed during training. Next, terms with higher similarity tend to be closer to the actual subject pertaining to the content of the input document, even if it is unknown and its vector is only inferred by the model. Therefore, the selection of the cosine similarity measure as a ranking criterion is effective and validated by the relatively high MAP values.

In the near future, we intend to make available system features as a web service, so as to facilitate seamless integration with the LOOR as well as interoperability with other web-based learning management and research services. To this end, we consider a more robust approach for dataset preparation, update and maintenance, possibly involving text-oriented and noSQL databases and its coupling with an incremental training process. The use of a distributed infrastructure could also help with the increased needs for space and computational power that will be posed by such big data requirements.

There is evidence that the proposed system and methodology is affected by and therefore is capable of implicitly learning the semantic relationships among terms in the thesaurus, for example, by proposing terms that share common ancestors or are otherwise related. These terms have not been considered by indexers possibly because they are too far apart in the concept hierarchy or represent research by-products rather than the core topic of a publication. To the extent that these terms are indeed relevant, this fact can open a whole new set of possibilities other than efficient indexing, such as to identify potential new research directions and facilitate novel results in the field of interest.

References

1. Eichhorn S. and G. W. Matkin. Massive open online courses, big data, and education research, New Directions for Institutional Research, 2015 (167): 27-40. Wiley, 2016.
2. Mao Y. and Z. Lu, MeSH Now: automatic MeSH indexing at PubMed scale via learning to rank, in J Biomed Semantics. 17;8(1):15. doi: 10.1186/s13326-017-0123-3, April 2017.
3. Koutsomitropoulos D. A., G. D. Solomou, and A. K. Kalou. Federated Semantic Search Using Terminological Thesauri for Learning Object Discovery, International Journal of Enterprise Information Management 30 (5): 795-808. Emerald, 2017.
4. Koutsomitropoulos D. A. and G. D. Solomou. A Learning Object Ontology Repository to Support Annotation and Discovery of Educational Resources using Semantic Thesauri, IFLA Journal 44 (1): 4-24. SAGE, 2018.

5. Europe PMC Consortium. Europe PMC: A Full-Text Literature Database for the Life Sciences and Platform for Innovation. Nucleic Acids Research 43. Database issue (2015): D1042–D1048. PMC. Web. 11 Aug. 2017.
6. McMartin F., MERLOT: a model for user involvement in digital library design and implementation, Journal of Digital Information, 5 (3), 2006.
7. U.S. National Library of Medicine. Medical Subject Headings, 2019. [Online]. Available: https://www.nlm.nih.gov/mesh/meshhome.html
8. Koutsomitropoulos D., A. Andriopoulos and S. Likothanassis, ''Subject Classification of Learning Resources Using Word Embeddings and Semantic Thesauri'', In IEEE Innovations in Intelligent Systems and Applications 2019 (INISTA), Sofia, Bulgaria, July 3-5, 2019
9. Mikolov T., K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In ICLR Workshop, 2013.
10. Le Q.V. and T. Mikolov, Distributed Representations of Sentences and Documents, in 31st International Conference on Machine Learning, ICML, 2014.
11. Mandelbaum A. and A. Shalev, Word Embeddings and Their Use In Sentence Classification Tasks, in CoRR, vol. abs/1610.08229, 2016.
12. Turner C. A., A. D. Jacobs, C. K. Marques, J. C. Oates, D. L. Kamen, P. E. Anderson, and J. S. Obeid. Word2Vec inversion and traditional text classifiers for phenotyping lupus. BMC, in Medical Informatics and Decision Making, vol.17, pp. 126.-136, January 2017.
13. Liu Q., H. Huang, Y. Gao, X. Wei, Y. Tian, and L. Liu, Task-oriented Word Embedding for Text Classification. COLING, 2018.
14. S. Suraj and V. Deepali, Unsupervised Text Classification and Search using Word Embeddings on a Self-Organizing Map, in International Journal of Computer Applications. Volume 156, pp. 35-37, 10.5120/ijca2016912570, December 2016.
15. Stein R. A., P. A. Jaques, and J. F. Valiati, An analysis of hierarchical text classification using word embeddings, Information Sciences, Volume 471, pp. 216-232, 2019.
16. Petrolito R. and F. D. Orletta, Word Embeddings in Sentiment Analysis, in Proceedings of the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018), vol. 2253, Torino, Italy, 2018.
17. Petrolito R. and F. D. Orletta, Document retrieval and question answering in medical documents. A large-scale corpus challenge, in Proceedings of the Biomedical NLP Workshop associated with RANLP, Varna, Bulgaria, pp. 1-7, September 2017.
18. Meilin Z., Research on Text Classification Method Based on Multi-type Classifier Fusion, in 8th International Conference on Social Network, Communication and Education (SNCE 2018), Shenyang, China, vol. 83, pp. 798-805, May 2018.
19. Wang R., W. Liu, and C. McDonald, Corpus-independent generic keyphrase extraction using word embedding vectors, in Software Engineering Research Conference, vol. 39, 2014.
20. Wang R., W. Liu, and C. McDonald, Using word embeddings to enhance keyword identification for scientific publications, in Proceedings of the 26th Australasian Database Conference, ADC 2015, Melbourne, Australia. Springer, pp. 257–268, June 2015.
21. Mahata D., J. Kuriakose, R.R. Shah, R. Zimmermann, and J.R. Talburt, Theme-Weighted Ranking of Keywords from Text Documents Using Phrase Embeddings, in IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), Miami, USA, pp. 184-189, April 2018.
22. Chen S., A. Soni, A. Pappu, and Y. Mehdad, DocTag2Vec: An Embedding Based Multi-label Learning Approach for Document Tagging, in Proceedings of the 2nd Workshop on Representation Learning for NLP, Vancouver, Canada, pp. 111-120, August 2017.
23. Peters M. E., M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee and L. Zettlemoyer Deep contextualized word representations, arXiv:1802.05365v2 [cs.CL], NAACL March 2018.
24. Sheikhshabbafghi G., I. Birol, and A. Sarkar, ''In-domain Context-aware Token Embeddings Improve Biomedical Named Entity Recognition'', in Proceedings of the

9th International Workshop on Health Text Mining and Information Analysis (LOUHI 2018), Brussels, Belgium, pp. 160–164, doi: 10.18653/v1/W18-5618, October 2018
25. Martínez-Romero M., C. Jonquet, M.J. O'Connor, J. Graybeal, A. Pazos, and M. A. Musen, NCBO Ontology Recommender 2.0: an enhanced approach for biomedical ontology recommendation. Journal of biomedical semantics, 8(1), 21. doi:10.1186/s13326-017-0128-y, 2017
26. Peng S., R. You, H. Wang, C. Zhai, H. Mamitsuka and S. Zhu, DeepMeSH: deep semantic representation for improving large-scale MeSH indexing, in Bioinformatics. 15;32(12): i70-i79. doi: 10.1093/bioinformatics/btw294, June 2016.
27. Kosmopoulos A., I. Androutsopoulos and G. Paliouras, Biomedical Semantic Indexing using Dense Word Vectors in BioASQ, J BioMed Semant Suppl BioMedl Inf Retr, 2015
28. Abdeddaïm S., S. Vimard and L. F. Soualmia, 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], November 2018.
29. Segura B., P. Martínez and M. A. Carruan, Search and Graph Database Technologies for Biomedical Semantic Indexing: Experimental Analysis, in JMIR Med Inform. 1;5(4): e48. doi: 10.2196/medinform.7059, December 2017.
30. Ternier S., K. Verbert, G. Parra, B. Vandeputte, J. Klerkx, E. Duval, et al. The ariadne infrastructure for managing and storing metadata, IEEE Internet Computing, 13(4), 2009.
31. Miles A. and S. Bechhofer, eds. SKOS Simple Knowledge Organization System Reference. W3C Recommendation, 2009. Available: http://www.w3.org/TR/skos-reference
32. Schnabel T., I. Labutov, D. M. Mimno, and T. Joachims, Evaluation methods for unsupervised word embeddings, in Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP), Lisbon, Portugal, pp. 298-307, September 2015.
33. Assem Van M., V. Malaisé, A. Miles, and G. Schreiber. A Method to Convert Thesauri to SKOS, In The Semantic Web: Research and Applications: 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11-14, 2006, Proceedings (Vol. 4011, p. 95). Springer, 2006.
34. U.S. Department of Health & Human Services, MEDLINE®PubMed® XML Element Descriptions and their Attributes, 2018. [Online]. Available: https://www.nlm.nih.gov/bsd/licensee/elements_descriptions.html
35. Zhang E., and Y. Zhang (2009) Average Precision. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA

Appendix A.

Table A1. Mean precision at different ranks and threshold values.

Rank	Sim	Precision				
		# items with terms above threshold	# items with terms below threshold	# annotations	# annotations that are relevant	Value
1	0.36	13,470	0	41,374	4,038	0.300
1	0.4	13,452	18	41,315	4,048	0.301
1	0.44	13,189	281	40,471	4,009	0.304
1	0.48	11,887	1,583	36,339	3,849	0.324
1	0.52	8,923	4,547	26,968	3,140	0.352
1	0.56	5,262	8,208	15,507	2,090	0.397
1	0.6	2,379	11,091	6,704	1,094	0.460

1	0.64	882	12,588	2,410	424	0.481
1	0.68	281	13,189	702	147	0.523
1	0.72	70	13,400	174	42	0.600
1	0.76	8	13,462	18	6	0.750
3	0.36	13,470	0	41,374	7,832	0.194
3	0.4	13,442	28	41,286	7,812	0.194
3	0.44	13,167	303	40,406	7,681	0.194
3	0.48	11,880	1,590	36,320	6,651	0.187
3	0.52	8,924	4,546	26,965	4,658	0.174
3	0.56	5,219	8,251	15,378	2,676	0.171
3	0.6	2,411	11,059	6,737	1,245	0.172
3	0.64	877	12,593	2,340	446	0.170
3	0.68	269	13,201	672	135	0.167
3	0.72	69	13,401	173	40	0.193
3	0.76	6	13,464	17	6	0.333
10	0.36	13,469	1	41,370	13,528	0.100
10	0.4	13,445	25	41,307	13,255	0.099
10	0.44	13,187	283	40,477	11,721	0.089
10	0.48	11,895	1,575	36,421	8,525	0.072
10	0.52	8,889	4,581	26,824	5,142	0.058
10	0.56	5,230	8,240	15,368	2,785	0.053
10	0.6	2,374	11,096	6,654	1,233	0.052
10	0.64	863	12,607	2,313	439	0.051
10	0.68	279	13,191	713	149	0.053
10	0.72	69	13,401	162	38	0.055
10	0.76	6	13,464	10	5	0.083

Table A2. Mean recall at different ranks and threshold values.

Rank	Sim	Recall				
		# items with terms above threshold	# items with terms below threshold	# annotations	# annotations that are relevant	Value
1	0.36	13,470	0	41,374	4,038	0.118
1	0.4	13,452	18	41,315	4,050	0.120
1	0.44	13,191	279	40,479	4,008	0.119
1	0.48	11,888	1,582	36,328	3,859	0.129
1	0.52	8,919	4,551	26,954	3,135	0.142
1	0.56	5,255	8,215	15,494	2,093	0.166
1	0.6	2,378	11,092	6,699	1,096	0.206
1	0.64	888	12,582	2,429	428	0.228
1	0.68	283	13,187	707	149	0.270
1	0.72	69	13,401	171	42	0.334
1	0.76	8	13,462	18	6	0.510
3	0.36	13,470	0	41,374	7,835	0.219
3	0.4	13,442	28	41,286	7,816	0.219
3	0.44	13,166	304	40,407	7,676	0.221

3	0.48	11,881	1,589	36,326	6,664	0.213
3	0.52	8,924	4,546	26,993	4,655	0.206
3	0.56	5,222	8,248	15,381	2,675	0.212
3	0.6	2,403	11,067	6,718	1,239	0.226
3	0.64	880	12,590	2,347	449	0.240
3	0.68	269	13,201	672	135	0.253
3	0.72	69	13,401	173	40	0.305
3	0.76	6	13,464	17	6	0.464
10	0.36	13,469	1	41,370	13,527	0.361
10	0.4	13,446	24	41,309	13,248	0.355
10	0.44	13,188	282	40,481	11,726	0.325
10	0.48	11,902	1,568	36,455	8,522	0.267
10	0.52	8,892	4,578	26,831	5,140	0.227
10	0.56	5,230	8,240	15,366	2,786	0.220
10	0.6	2,374	11,096	6,647	1,236	0.230
10	0.64	868	12,602	2,323	439	0.244
10	0.68	279	13,191	712	150	0.269
10	0.72	68	13,402	160	38	0.300
10	0.76	6	13,464	10	5	0.556

Table A3. MAP at different ranks and threshold values.

Rank	Sim	Ranking					
		# items with terms above threshold	# items with terms below threshold	# items with relevant terms	# annotations	# annotations that are relevant	Value
3	0.36	13,470	0	6,863	40,407	7,857	0.763
3	0.4	13,452	18	6,814	40,158	7,838	0.767
3	0.44	13,191	279	6,649	37,944	7,637	0.771
3	0.48	11,891	1,579	5,872	30,089	6,659	0.804
3	0.52	8,916	4,554	4,285	18,275	4,754	0.851
3	0.56	5,252	8,218	2,503	8,570	2,664	0.911
3	0.6	2,380	11,090	1,206	3,259	1,245	0.952
3	0.64	886	12,584	449	1,079	457	0.972
3	0.68	283	13,187	151	299	152	0.987
3	0.72	70	13,400	43	72	43	0.988
3	0.76	8	13,462	6	8	6	1.000
10	0.36	13,470	0	9,579	134,327	13,583	0.558
10	0.4	13,442	28	9,424	128,277	13,280	0.566
10	0.44	13,165	305	8,624	100,062	11,715	0.612
10	0.48	13,470	1,589	6,800	56,405	8,567	0.703
10	0.52	8,924	4,546	4,495	25,261	5,163	0.807
10	0.56	5,224	8,246	2,563	9,974	2,765	0.898
10	0.6	2,403	11,067	1,208	3,507	1,251	0.947
10	0.64	879	12,591	444	1,079	450	0.974
10	0.68	270	13,200	134	294	135	0.985

10	0.72	70	13,400	40	73	40	0.988
10	0.76	6	13,464	6	6	6	1.000

Table A4. Mean R-precision at different ranks and threshold values.

Sim	R-Precision				
	# items with terms above threshold	# items with terms below threshold	# annotations	# annotations that are relevant	Value
0.36	13,470	0	41,374	8,192	0.205
0.4	13,452	18	41,315	8,192	0.207
0.44	13,191	279	40,475	7,861	0.202
0.48	11,888	1,582	36,331	6,664	0.198
0.52	8,923	4,547	26,973	4,622	0.192
0.56	5,259	8,211	15,504	2,603	0.196
0.6	2,375	11,095	6,686	1,226	0.225
0.64	890	12,580	2,429	454	0.239
0.68	281	13,189	702	151	0.275
0.72	69	13,401	171	43	0.338
0.76	8	13,462	18	6	0.510