Bioinformatics, 36(5), 2020, 1533–1541
doi: 10.1093/bioinformatics/btz756
Advance Access Publication Date: 9 October 2019
Original Paper

Data and text mining

FullMeSH: improving large-scale MeSH indexing with full text

Suyang Dai[1], Ronghui You[1], Zhiyong Lu [2], Xiaodi Huang[3], Hiroshi Mamitsuka[4,5] and Shanfeng Zhu[1,6,7,*]

[1]School of Computer Science and Shanghai Key Lab of Intelligent Information Processing, Fudan University, Shanghai 200433, China, [2]National Center for Biotechnology Information (NCBI), National Library of Medicine (NLM), National Institutes of Health (NIH), Bethesda, MD, USA, [3]School of Computing and Mathematics, Charles Sturt University, Albury, NSW 2640, Australia, [4]Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji, Kyoto Prefecture, Japan, [5]Department of Computer Science, Aalto University, Espoo, Finland, [6]Shanghai Institute of Artificial Intelligence Algorithms and Institute of Science and Technology for Brain-Inspired Intelligence, Fudan University, Shanghai 200433, China and [7]Ministry of Education, Key Laboratory of Computational Neuroscience and Brain-Inspired Intelligence (Fudan University), China

*To whom correspondence should be addressed.

Associate Editor: Robert Murphy

Received on April 29, 2019; revised on August 29, 2019; editorial decision on September 26, 2019; accepted on October 3, 2019

Abstract

Motivation: With the rapidly growing biomedical literature, automatically indexing biomedical articles by Medical Subject Heading (MeSH), namely MeSH indexing, has become increasingly important for facilitating hypothesis generation and knowledge discovery. Over the past years, many large-scale MeSH indexing approaches have been proposed, such as Medical Text Indexer, MeSHLabeler, DeepMeSH and MeSHProbeNet. However, the performance of these methods is hampered by using limited information, i.e. only the title and abstract of biomedical articles.

Results: We propose FullMeSH, a large-scale MeSH indexing method taking advantage of the recent increase in the availability of full text articles. Compared to DeepMeSH and other state-of-the-art methods, FullMeSH has three novelties: (i) Instead of using a full text as a whole, FullMeSH segments it into several sections with their normalized titles in order to distinguish their contributions to the overall performance. (ii) FullMeSH integrates the evidence from different sections in a 'learning to rank' framework by combining the sparse and deep semantic representations. (iii) FullMeSH trains an Attention-based Convolutional Neural Network for each section, which achieves better performance on infrequent MeSH headings. FullMeSH has been developed and empirically trained on the entire set of 1.4 million full-text articles in the PubMed Central Open Access subset. It achieved a Micro F-measure of 66.76% on a test set of 10 000 articles, which was 3.3% and 6.4% higher than DeepMeSH and MeSHLabeler, respectively. Furthermore, FullMeSH demonstrated an average improvement of 4.7% over DeepMeSH for indexing Check Tags, a set of most frequently indexed MeSH headings.

Availability and implementation: The software is available upon request.

Contact: zhusf@fudan.edu.cn

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

As the largest bibliographic database of biomedical literature, MEDLINE (Sayers *et al.*, 2019), maintained by US National Library of Medicine (NLM), covers more than 5200 world-wide journals and 25 million citations (https://www.nlm.nih.gov/bsd/medline_pubmed_production_stats.html). Compared with other scientific and bibliographic databases, a distinct feature of MEDLINE is that the citations (records) are indexed with a controlled vocabulary, called Medical Subject Heading (MeSH). As of 2018, the vocabulary contains 28 939 MeSH main headings (MHs) (https://www.nlm.nih.gov/mesh/filelist.html) and each citation is indexed by 13 MHs on average. The MeSH indexing of MEDLINE citations has facilitated a wide range of applications in biomedical text mining and information retrieval, such as query expansion (Lu *et al.*, 2009; Stokes *et al.*, 2009) and document clustering (Gu *et al.*, 2013; Zhu *et al.*, 2009). Therefore, MeSH indexing is of great significance to biomedical researchers for hypothesis generation and knowledge discovery.

MEDLINE citations are mostly indexed by human curators at NLM who review the full text of each article and assign most suitable MHs to the article. The average cost of annotating one MEDLINE citation was estimated to be around $9.4 (Mork *et al.*, 2013), As the number of MEDLINE citations is rapidly increasing in the last few years, this labor-intensive process becomes highly costly. In fact, 904 636 citations were added to MEDLINE in 2018, which is around 11% increase over 2017 (813 598). To deal with the ever-expanding MEDLINE citations, NLM has used automated and semi-automated methods to improve the efficiency of MeSH indexing. Specifically, NLM has developed a software, Medical Text Indexer (MTI), to recommend MHs based on the title and abstract of MEDLINE citations (Aronson *et al.*, 2004; Mork *et al.*, 2014). In the automated mode, MHs are provided by MTI directly, while in the semi-automated mode, MHs are first provided by MTI and then reviewed (and possibly adjusted) by curators. These two types of methods are responsible for indexing 5% and 18% newly completed MEDLINE citations, respectively (https://www.nlm.nih.gov/pubs/techbull/ja18/ja18_indexing_method.html). All these highlight the importance of developing accurate and automatic MeSH indexing methods to keep up with the rapidly growing MEDLINE citations.

Automatic MeSH indexing can be regarded as a large-scale multi-label classification problem (Liu *et al.*, 2015), where each MH is a class label and each instance (citation) is associated with multiple MHs. The challenge of large-scale MeSH indexing comes from both the label and instance sides. For the label side, the number of distinct MHs is close to 30 000 with a highly biased distribution. On the other hand, for the instance side, traditional BOW (Bag of Words) representations cannot capture complicated semantics of the biomedical literature, which has many domain-specific concepts, phrases and abbreviations. These two challenging problems have been addressed by many studies. For example, MeSHLabeler (Liu *et al.*, 2015) used 'learning to rank' (LTR) to solve the challenge in the label side by integrating multiple types of evidence with different types of classifiers. Under the framework of MeSHLabeler, DeepMeSH (Peng *et al.*, 2016) further addressed the challenge in the instance side by using deep semantic representation.

Although these state-of-the-art methods have made significant progress over MTI, there is a clear gap between these automatic methods and manual indexing, in terms of data used. Specifically, the NLM curators review full text for assigning MHs, while the existing methods use only title and abstract, which may not have important and sufficient information for MeSH indexing. Mork *et al.* (2017) indicated that the main reason why MTI misses some most frequently occurring MHs (also known as Check Tags) is that the relevant information appears only in full text but not in title or abstract. In the meantime, the amount of available full text for text mining in PubMed Central (PMC) database is rapidly growing in recent years. Thus a key issue is how to use such a large number of articles with full text for improving the predictive performance of MeSH indexing.

In this work, we propose a novel method called FullMeSH that takes advantage of full text to improve the performance of large-scale MeSH indexing. The main contributions of FullMeSH are as follows:

i. To the best of our knowledge, FullMeSH is the first automatic MeSH indexing method that mines the full text articles (>1.4 M) for assigning the entire 28 000 MHs. Existing methods are incapable of dealing with a large amount of full text and noise. As such, we segment full text into several sections with normalized titles so that their different information can be integrated effectively.
ii. With various representations of different sections, FullMeSH makes full use of deep learning and other traditional models to generate multiple types of evidence. Inspired by recent work on medical codes classification (Mullenbach *et al.*, 2018), we use a label-wise Attention-based Convolution Neural Network (AttentionCNN) as a new component for utilizing new evidence, which performs well on infrequent MHs. All evidence from full text are finally fused into an LTR framework to make the final recommendation.
iii. We conducted a thorough experiment for validating FullMeSH by using the whole PMC Open Access Subset (>1.4M) with 10 000 test articles. FullMeSH achieved a Micro F-measure of 66.76%, which is 3.3% and 6.4% higher than that of our previous abstract-based methods, DeepMeSH and MeSHLabeler, respectively. FullMeSH improves around 4.7% in Micro F-measure over DeepMeSH for indexing Check Tags on average. Moreover, we analyzed the contribution of each section to the overall indexing result.

2 Related work

2.1 Large-scale MeSH indexing based on title and abstract

Since 2013, BioASQ challenge has provided practical and realistic benchmark opportunities under the cooperation with NLM (Tsatsaronis *et al.*, 2015). Many effective algorithms for large-scale MeSH indexing have been developed through the BioASQ challenge, such as MetaLabeler (Tsoumakas *et al.*, 2013), L2R (Huang *et al.*, 2011; Mao and Lu, 2013), MeSHNow (Mao and Lu, 2017), MeSHLabeler (Liu *et al.*, 2015) and DeepMeSH (Peng *et al.*, 2016), in which DeepMeSH achieved the best results of Batches 1 and 2 and second best results of Batch 3 in BioASQ 6. Most recently, deep learning models have also been adopted for large-scale MeSH indexing such as AttentionMeSH (Jin *et al.*, 2018) and MeSHProbeNet (Xun *et al.*, 2019), which, respectively, had the best and third best results of Batch 3 in BioASQ 6. Specifically, both AttentionMeSH and MeSHProbeNet utilized deep recursive neural network and attention mechanism for large-scale MeSH indexing. This inspired us to incorporate a deep learning based model to generate global evidence in FullMeSH. The main difference between AttentionMeSH and MeSHProbeNet is that AttentionMeSH uses a multi-label attention for classification while MeSHProbeNet employs a multi-view framework to integrate multiple self-attentive MeSH probes, where each probe could extract different aspect of biomedical knowledge. Note that the training process of Bi-GRU network in AttentionMeSH and MeSHProbeNet is time-consuming because the GRU must be calculated sequentially. Inspired by the recent work of Mullenbach *et al.* (2018), we use CNN to train deep models for all sections, which is much faster than Bi-GRU in terms of training speed. Compared to Bi-GRU, CNN can naturally encode the k-gram features for a document/documents. In addition, by incorporating with attention mechanism, the network can choose the most important phrases for each MH. Therefore, we implement AttentionCNN (See Step 2 of the Section 3 for detail), in FullMeSH to improve the predictive performance. In particular, since the attention mechanism can use the prior information of labels and label-specific features, AttentionCNN can properly deal with the problem of infrequent MHs, for which traditional support vector machine (SVM) cannot perform well.

Note that all existing top-performing methods use title and abstract only, which cannot enjoy the rich information of full text.

2.2 MeSH indexing based on full text

So far MeSH indexing with full text has been studied using a relatively small set of biomedical articles or a small number of specific MHs. For example, Gay *et al.* (2005) found that using full text as the input to MTI could increase recall, but with a trade-off in precision, after conducting experiments on a collection of 17 journal issues containing 500 biomedical articles. Jimeno-Yepes *et al.* (2013) randomly selected 1413 articles from PMC Open Access Subset and examined the performance of automatically summarizing full text for MeSH indexing. Experimental results demonstrated that using the summary of full text would achieve higher MiF (Micro F-measure) than using full text as a whole, but lower MiF than using title and abstract only. In addition, Demner-Fushman and Mork (2015) developed a rule-based algorithm that extracts characteristics of subjects from full text to supplement title and abstract, and used the extracted results as the input to MTI. By expanding abstract

with sentences from the Section 3, their approach achieved an improvement of 1% on F1-score during the evaluation of 29 Check Tags assignments. All these studies suggest that improving the performance of large-scale MeSH indexing with full text information remains highly challenging. In contrast to previous work, FullMeSH uses the full text of all available biomedical articles in PMC Open Access Subset. Furthermore, we thoroughly examined the performance of FullMeSH over all MHs.

3 Materials and methods

3.1 Overview

Given the full text of a new biomedical article, MeSH indexing is the problem of assigning a certain number of relevant MHs from the whole MeSH vocabulary (>28 000 MHs). Figure 1 shows the entire workflow of FullMeSH, which consists of four steps, including two components: MeSHRanker and MeSHNumber for Steps 3 and 4, respectively. Given an article, MeSHRanker generates a ranked list of candidate MHs from multiple types of evidence in the 'LTR' framework. On the other hand, for each article, MeSHNumber predicts the number of MHs by considering multiple features of full text rather than only title and abstract. That is, for a new article, MeSHNumber predicts the number m of MHs to be returned as the final recommendation, out of the ranked MH list by MeSHRanker.

3.2 Competing methods: MeSHLabeler and DeepMeSH

MeSHLabeler and DeepMeSH tackle the MeSH Indexing problem by using only the title and abstract of each article. Specifically, MeSHLabeler integrates multiple types of evidence generated from the BOW representation in the LTR framework, with different types of classifiers, such as SVM for global evidence from the whole MEDLINE, k-nearest neighbor (KNN) for local evidence from similar citations and pattern matching for self evidence from the test citation itself. On the other hand, DeepMeSH further integrates additional types of strong evidence generated from a dense semantic representation named D2V-TFIDF, which is obtained by combining the power of both sparse representation, TFIDF and dense representation, D2V (Le and Mikolov, 2014). FullMeSH shares the same framework as MeSHLabeler and DeepMeSH in general but differs in its unique ability to make use of information in full text. To assess this particular effect, we directly compare FullMeSH with MeSHLabeler and DeepMeSH in this work, both of which have achieved the state-of-the-art performance in recent BioASQ challenges. MeSHLabeler achieved the first place in BioASQ 2 and 3, and DeepMeSH achieved the first place in BioASQ 4, 5 and Batches 1 and 2 of BioASQ 6. Note that both MeSHLabeler and DeepMeSH outperformed MTI significantly in all these BioASQ challenges, which is the production system used by the NLM indexers in their routine MeSH indexing.

3.3 Proposed method: fullMeSH

FullMeSH, MeSHLabeler and DeepMeSH share the same workflow. Figure 1 shows the workflow of FullMeSH as well as the differences among the three methods. Originally for MeSH indexing, MeSHLabeler had a workflow (shown in white in Fig. 1), which was modified by DeepMeSH [which added Document2Vec (D2V) and D2V-TFIDF (shown in green)]. Then FullMeSH further adds three points: (i) all sections of full text, (ii) a new type of text representation, i.e. word2vec (W2V) and (iii) a new type of machine learning model, AttentionCNN (all new points shown in pink in Fig. 1). We explain the procedure of FullMeSH, following Figure 1.

3.3.1 Step 1: Generating representation of a document

1. Extracting and mapping of sections of full text
   A naive approach for incorporating full text is to treat all different sections in one article as one instance. However, this approach has two serious drawbacks: (i) long vector problem: practically one citation (title and abstract) has around 229 words on average, while the full text has around 5000 words on average. Thus, a long feature vector of full text would cause difficulties when training a model; and (ii) losing information: several studies have pointed out that information from different sections plays different roles for indexing, implying that merging different sections together might lose some important information. Thus, we divide one full text article into five sections: *Title and Abstract*, *Introduction*, *Methods and Materials*, *Result and Experiment*, *Conclusion and Summary*.In preprocessing data, section names are not consistent in given articles (We get the section names and its corresponding text by parse the tags of XML files). For example, *Methods and Materials* might vary slightly, such as *methods*, *method*, *Methodology*, *Methods*, *Methods and materials* and *Methods/Design*, etc. To tackle this problem of data cleaning, we use pre-defined pattern sets to link a section title in one article to the normalized section title (see Supplementary Material for the pre-defined patterns).
2. Generating multiple representations of sections
   After extracting the sections of an article, FullMeSH represents the

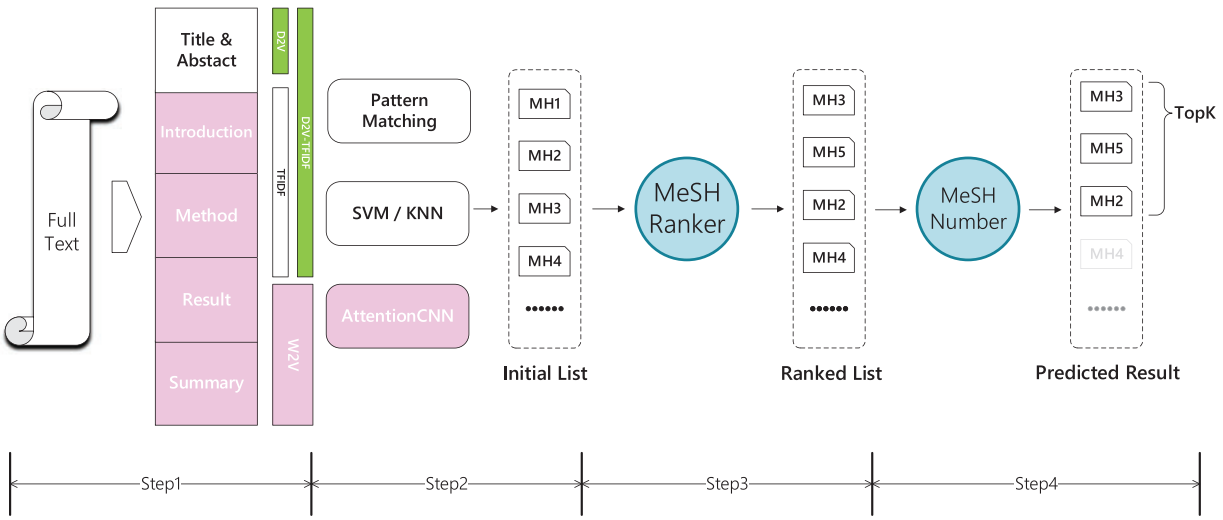

Fig. 1. The workflow of FullMeSH: Step 1. Multiple sections of full text are extracted, mapped and shown by representation; Step 2. Candidate MHs are generated; Step 3. 'LTR' is used to rank candidate MHs and Step 4. Top ranked MHs are selected

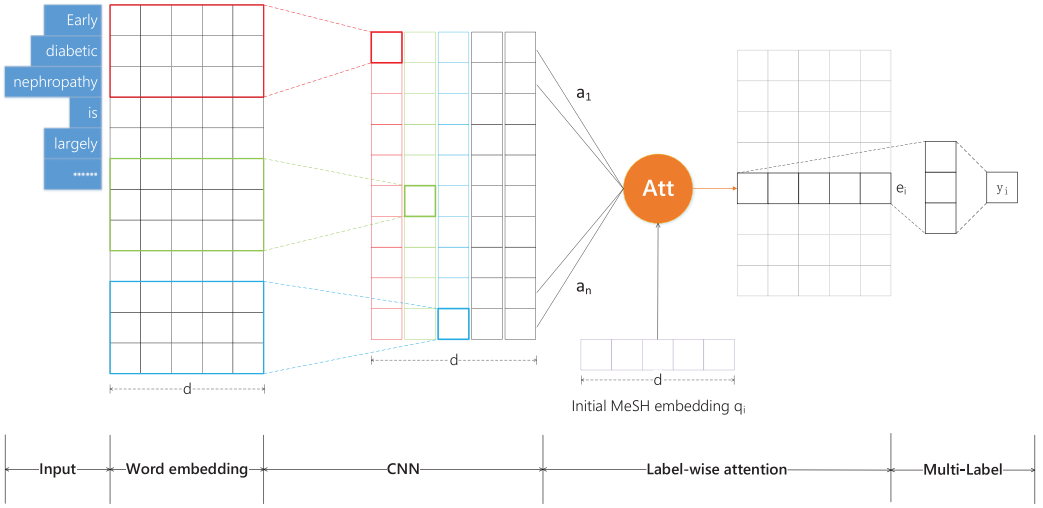

Fig. 2. The architecture of AttentionCNN

text of each section by using four representations: (i) TFIDF, a classical BOW representation, which is a sparse and long vector; (ii) D2V (Le and Mikolov, 2014), a widely used deep representation, which is a dense continuous vector; (iii) D2V-TFIDF, a concatenation of D2V and TFIDF and (iv) word2vec (W2V) (Mikolov *et al.*, 2013), a well known word deep representation, which is also a dense vector.

3.3.2 Step 2: Generating candidate MHs

After generating representations (Step 1) for each section, we generate numerous evidence for each section of one article mainly using supervised learning, as follows:

1. Global evidence by the binary classifier SVM
   We train a binary classifier for each MH and each section, by using the whole training data, which can be regarded as *global* evidence (against *local* evidence. See local evidence for the reason why we say global evidence). Given an article, we can predict scores of all MHs using the binary classifiers of each section.
2. Global evidence by multi-label classifier (AttentionCNN)
   We train a multi-label classifier for all MHs using the whole training data. Again given an article, we can predict scores of all MHs using AttentionCNN for each section. We explain AttentionCNN more in detail below at the end of this step.
3. Local evidence KNN
   We use KNN to find articles with similar sections. The resultant articles are found just by similarity, meaning that the entire database is not necessarily used for each article. We call this evidence as KNN local evidence. In practice, for each section, we compute the cosine similarity score between two articles, by using only D2V and D2V-TFIDF in feature vectors and obtain k most similar articles. To obtain MHs, we take a weighted voting by k articles in each section, resulting in several groups of MHs, each from one section.
4. Self evidence (pattern matching in test citation)
   We use pattern matching for finding MHs or their concept terms in each section of test citation, where MHs are weighted by their number of appearances.
5. Other evidence (already used in DeepMeSH)

   (i) MeSH dependency that considers correlations between two MHs; and
   (ii) MeSH frequency that counts the probability of each MH in the target journal.

Using the above evidence, we generate a list of candidate MHs, where the list should not have irrelevant (or redundant) candidates, to reduce computational burden for training and prediction, for highly large-scale data (See Supplementary Materials for the detailed settings).

Below we will explain AttentionCNN in more detail, which is 'Global evidence by the multi-label classifier'.

3.3.3 Attention-based convolutional neural network: label-wise attention CNN

The mechanism called *attention* is the recent idea of selecting the most important part in image or text (Bahdanau *et al.*, 2014), and has been successfully applied in many problems in natural language processing (Mullenbach *et al.*, 2018; Yang *et al.*, 2016). This idea would be useful for our problem of multi-label classification in the following two ways: (i) in a long document, the attention mechanism can choose the most informative segment (K-gram) for each MH, and (ii) also the mechanism might be useful for solving the label imbalance problem of MeSH indexing (Peng *et al.*, 2016), by using pre-available information on MHs. In fact for this problem, traditional classifiers, such as SVM, has a limitation due to only few positive instances available.

As such, we propose Attention-based Convolutional Neural Network (AttentionCNN) or label-wise attention CNN. Figure 2 shows the architecture of AttentionCNN, which has four main layers: (i) Word embedding: we use W2V for document representation as the input of CNN. Let l and d be the document length and the dimension of word embedding, respectively. The output of this layer is matrix $\boldsymbol{X} \in \mathbb{R}^{l \times d}$. (ii) CNN: Given document representation $\boldsymbol{X}$, we use d 1-D convolution filters with the width of u to generate document feature matrix $\boldsymbol{C} \in \mathbb{R}^{(l-u+1) \times d}$. Note that d is also chosen as the number of filters for convenient computation and avoiding unnecessary dimension transformation. (iii) Multi-label attention: we use what we call *multi-label attention layer* instead of general so-called *max-pooling* to generate label-specific features from the document feature matrix. In this layer, we generate initial representation $\boldsymbol{q}_i$ for i-th MH, MH_i, which is obtained by the average over the words from the synonym terms (obtained from https://www.nlm.nih.gov/mesh/filelist.html) of each MH:

$$\boldsymbol{q}_i = \frac{1}{M}\sum_{j=1}^{M} \boldsymbol{w}_j, i = 1, \ldots, K$$

where M is the number of words in the synonym terms of MH_i, K is the number of all MHs and $\boldsymbol{w}_j$ is the word embedding of word j in the synonym terms. Note that the representation of MHs would be updated during the training process.

Given document feature matrix $\boldsymbol{C}$ and initial representation $\boldsymbol{q}_i \in \mathbb{R}^d$ for MH_i, label specific feature, e_i, of MH_i is obtained as follows:

$$\boldsymbol{e}_i = \boldsymbol{C}^T \boldsymbol{a}_i, \quad \text{where}$$
$$\boldsymbol{a}_i = \text{Softmax}(\boldsymbol{C}\boldsymbol{q}_i)$$

where $\boldsymbol{a}_i \in \mathbb{R}^{(l-u+1)}$ is called *attention vector* for MH_i. Label specific feature $\boldsymbol{e}_i$ is obtained by using $\boldsymbol{a}_i$ to have the weighted average over each row of document feature matrix $\boldsymbol{C}$. (iv) Multi-label output: after obtaining label specific feature $\boldsymbol{e}_i \in \mathbb{R}^d$, we obtain probability $\hat{y}_i$ for MH_i by connecting the feature to the output layer in a fully-connected manner:

$$\hat{y}_i = \text{Sigmoid}(\boldsymbol{W}_2^T f(\boldsymbol{W}_1 \boldsymbol{e}_i + \boldsymbol{b}_1) + \boldsymbol{b}_2)$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{h \times d}$, $\boldsymbol{b}_1 \in \mathbb{R}^h$, $\boldsymbol{W}_2 \in \mathbb{R}^h$ and $\boldsymbol{b}_2 \in \mathbb{R}^1$ are parameters to be trained, and f represents a non-linear activation function. We note a difference from Mullenbach *et al.* (2018): we use a fully-connected layer, which is shared by all MHs, to avoid overfitting in training, which might be caused by the limited training data of full text.

Finally, we use the binary cross entropy (BCE) loss, for the objective function to train AttentionCNN, where we note that the BCE loss is well used in multi-label neural networks (Liu *et al.*, 2017).

Overall, the multi-label deep learning architecture of AttentionCNN, attention mechanism, as well as the initialization using prior knowledge are all helpful for the indexing of infrequent MHs.

3.3.4 Step 3: MeSHRanker: 'LTR' to rank candidate MHs

The generated candidate MHs in Step 2 are ranked by LTR of MeSHRanker by using multiple evidence generated from all sections of the full text. The input feature vector of MeSHRanker is a concatenation of four types of evidence: (i) Scores by AttentionCNN and SVM (of using D2V – TFIDF) for all five sections, resulting in 10 features in total; (ii) Scores by KNN (of D2V – TFIDF) and KNN (of D2V) for all five sections, resulting in 10 features in total; (iii) Count, precision and recall (for training data) by pattern matching for all five sections, resulting in 15 features in total.

Specifically, given a candidate MeSH term, count refers to the number of exact matching in the target section. Precision and recall refer to the values of precision and recall of this matched MeSH term in the whole training data, respectively, if we use pattern matching to predict its annotation. (iv) MeSH dependence score and MeSH frequency in the entire corpus, and the corresponding journal of each instance, resulting in three features (elements). Thus, all in all, the input feature vector has 38 features.

3.3.5 Step 4: MeSHNumber: selecting the top ranked MHs

MeSHNumber uses the following features as the input: (i) the number of annotated MHs of citations in the same published journal; (ii) the number of MHs of similar citations based on title and abstract; (iii) the highest scores of the MH by SVM and AttentionCNN based on title and abstract; (iv) the length of full text; and (v) the output of MeSHRanker. Again, note that (iv) and (v) are unique features of FullMeSH and not in DeepMeSH.

After we getting the number $\boldsymbol{N}$ of MHs for a citations, we would choose the top $\boldsymbol{N}$ ranked MHs outputed by MeSHRanker as the final prediction.

4 Results

4.1 Data collection

Using the PMC FTP service (https://www.ncbi.nlm.nih.gov/research/bionlp/APIs/BioC-PMC) (Comeau *et al.*, 2019), we downloaded the PMC Open Access Subset, which contains 2 589 238 articles (by October 2018). After matching with the citations in PubMed Annual Baseline Repository according to PMID, we obtained a set of 1 430 968 articles that have been indexed with MHs. We then normalized the data (as mentioned before) to have five corpora of different sections, and counted the total text length of each section in the corpus. We then used about 90% of all text in our experiments (since the remaining text are the description on, e.g. grant support, interest conflict or author contributions, etc., which are all less useful for MeSH indexing). For a fair comparison on the importance of each section and the LTR training process, we reserved the latest 50 000 articles with all sections.

Table 1. Statistics of the generated corpus for each of the five sections

Section	Dataset size #docs	Ave. length #words	Unigram (unique)	Bigram (unique)
Title and Abstract	1 430 968	229	100 621	1 649 861
Introduction	1 139 056	601	188 590	3 890 553
Method	1 091 326	1150	332 622	5 355 997
Result	1 051 289	1254	300 500	5 754 973
Conclusion	1 242 518	1177	296 136	6 571 810

Out of these 50 000 articles, the latest 10 000 articles were used as a test set for the performance evaluation of FullMeSH. We randomly divided the remaining 40 000 articles into 20 000 and 20 000 articles, which were used as the training set of MeSHRanker and the training set of MeSHNumber, respectively.

After removing the 50 000 articles above from the corpus of each section, the rest articles in the primary corpus were used as the training set of D2V, SVM, KNN and AttentionCNN. For generating classical TFIDF features, we used BioTokenizer to preprocess the raw text of MEDLINE (Jiang and Zhai, 2007). Both unigram and bigram features are employed to represent each citation. Similar to other work (Liu *et al.*, 2015; Peng *et al.*, 2016; Tsoumakas *et al.*, 2013), the features that appear less than 10 times were removed. As a result, we obtained unigram and bigram features of different sections with various lengths. Table 1 shows the statistics of the generated corpus for each of the five sections.

4.2 Implementation and parameter settings

FullMeSH was implemented by using several open source tools: XGBoost for the ranking model (Chen and Guestrin, 2016), and LibSVM for support vector regression (Chang and Lin, 2011). Linear SVM was implemented by using LIBLINEAR (Fan *et al.*, 2008). We used the python package of gensim (http://radimrehurek.com/gensim) to implement both D2V and W2V, with the dimensions of 200 and 300, respectively. The distributed bag of words was used to generate D2V. AttentionCNN was implemented by using TensorFlow (https://www.tensorflow.org/). AttentionCNN was trained by Adam with the default value of 0.001 as the learning rate. We utilized dropout of 0.5 and early stopping to avoid overfitting. The filter size of CNN was set to 10 and the number of filters was set to 300. For fairly evaluating the performances of the classifiers based on each section, the number of MHs was predicted by MetaLabeler using TFIDF representation with the same settings in (Liu *et al.*, 2015).

4.3 Performance evaluation measures

Denote K as the size of all labels (MHs), and N as the number of instances (articles). Let y_i and $\hat{y}_i \in \{0,1\}^K$ be the true and predicted labels for instance i, respectively. For evaluating the performance of different models, we use the three groups of main metrics: Micro (recall, precision and F-Measure), Macro (recall, precision and F-Measure) and Example Based (recall, precision and F-Measure). Due to the space limitation, the computation of Macro-based and Example-based metrics is presented in the Supplementary Materials.

In this work, we focus on MiF, which is also the main metric in the BioASQ challenge.

4.4 Experimental results

In this section, by using 10 000 benchmark test articles, we compare the performance of FullMeSH with MeSHLabeler and DeepMeSH,

in terms of three points: (i) generating candidate MeSH lists; (ii) ranking candidate MHs and (iii) predicting the number of MHs.

4.4.1 Generating candidate MHs by using full text

Due to the space limitation, we report a brief result only. We found AttentionCNN and pattern matching contributed to boosting the performance of FullMeSH. In detail, before using both, the average coverage of true MHs was 87.6%. After using AttentionCNN and also pattern matching, the coverage reaches 91.5% and 92.3%, respectively. These results imply that true positive MHs has been added by AttentionCNN and pattern matching to the initial candidate list.

4.4.2 Ranking candidate MHs (by MeSHRanker)

Table 2 shows the performance results obtained by adding diverse evidence from full text to a baseline approach, MeSHRanker. The baseline MiFs by MeSHLabeler and DeepMeSH using title and abstract only were 0.6179 and 0.6369, respectively. After the features from pattern matching were added to the LTR, the MiF score increased slightly to 0.6427. This indicates that the information through pattern matching from full text is valuable. After integrating the local evidence (KNN) into MeSHRanker, MiF was slightly improved to be 0.6446. This means that except for Title and Abstract, the improvement by local evidence from sections except Title and Abstract is limited. After adding global evidence (SVM and AttentionCNN for all sections) to MeSHRanker, a significant improvement on MiF, however, was achieved and the final MiF was 0.6570.

4.4.3 Predicting the number of MHs (by MeSHNumber)

To study the effectiveness of full text on predicting the number of MHs, we first generated the number of MHs by using MeSHNumber in DeepMeSH ($\text{MeSHNumber}_{\text{DeepMeSH}}$). Then we obtained the results by MeSHNumber in FullMeSH ($\text{MeSHNumber}_{\text{FullMeSH}}$, which is trained by adding the scores from $\text{MeSHRanker}_{\text{FullMeSH}}$ and full text). Table 2 reports a slight (0.65%) improvement of $\text{MeSHNumber}_{\text{FullMeSH}}$ over $\text{MeSHNumber}_{\text{DeepMeSH}}$ in terms of MiF, which suggests a small advantage of full text over title and abstract on predicting the number of MHs.

4.4.4 Overall results of FullMeSH

From the test set of 10 000 articles, we generated 50 test datasets using bootstrap with replacement. We then examined paired *t*-test to evaluate the statistical significance of performance improvement between FullMeSH and competing methods, where the *P*-value of smaller than 0.05 is considered as statistically significant. Table 3 reports the final performance results of FullMeSH against MeSHlabeler and DeepMeSH. FullMeSH outperformed the two competing methods, and in particular, FullMeSH achieved MiF of 66.76%, which was 3.3% and 6.4% higher than the state-of-the-art DeepMeSH and MeSHLabeler, respectively. These results demonstrate that: (i) full text plays an important role in large-scale MeSH indexing; and (ii) separating the full text into multiple sections and generating a learning model, such as AttentionCNN, for each section, worked well for using the full text by LTR.

4.5 Result analysis and discussion

In this section, we further explore the reason why FullMeSH on full text performed better than existing cutting-edge methods. We first explored the importance of each section in FullMeSH, and then compared the performance of different component methods (SVM, KNN and AttentionCNN) in FullMeSH. Finally, we analyzed the respective result from the perspective of MHs (label) and article (instance) separately.

4.5.1 Title and abstract were more informative than using the entire article as a whole

We evaluated the performance of the binary classifier (SVM) with D2V-TFIDF on five sections and 'Entire Article', which stands for using the entire full text as a whole. We founded that SVM on Title and Abstract performed clearly best under all metrics, achieved Mif of 0.6238, indicating that Title and Abstract cover the most relevant information about the article subject (See Supplementary Materials for the detailed result).

Note that the MiF score of Entire Article was only 0.6091, much lower than the corresponding result of Title and Abstract. This likely because the full text of the article is too lengthy to generate a good representation and because it would contain more noise. Furthermore, a lengthy representation would take more training time. In summary, the naive approach to using full text does not work.

4.5.2 The method was the second most informative section

Figure 3 shows the performance result obtained by removing the evidence from one of the five sections (except Title and Abstract) to explore the contribution of each section in terms of the performance

Table 2. Performance comparison of MeSHRanker and MeSHNumber with different types of evidence incrementally added

Method	MiP	MiR	MiF	EBP	EBR	EBF	MaP	MaR	MaF
$\text{MeSHRanker}_{\text{MeSHLabeler}}$	0.6098	0.6261	0.6179	0.6119	0.6409	0.6118	0.4832	0.4560	0.4692
$\text{MeSHRanker}_{\text{DeepMeSH}}$	0.6287	0.6454	0.6369	0.6310	0.6614	0.6310	0.5337	0.5373	0.5355
$+\text{PM}_{\text{All Sections}}$	0.6344	0.6513	0.6427	0.6365	0.6673	0.6366	0.5443	0.5400	0.5471
$+\text{KNNs}_{\text{All Sections}}$	0.6362	0.6532	0.6446	0.6387	0.6697	0.6389	0.5476	0.5526	0.5500
$+\text{SVMs}_{\text{All Sections}}$	0.6437	0.6609	0.6522	0.6461	0.6775	0.6463	0.5544	0.5608	0.5576
$+\text{AttentionCNN}_{\text{All Sections}}$	0.6485	0.6658	0.6570	0.6511	0.6826	0.6513	0.5598	**0.5707**	**0.5652**
$+\text{MeSHNumber}_{\text{DeepMeSH}}$	0.6992	0.6312	0.6634	0.6985	0.6476	0.6585	0.5744	0.5385	0.5559
$+\text{MeSHNumber}_{\text{FullMeSH}}$	**0.7021**	**0.6366**	**0.6677**	**0.7017**	**0.6522**	**0.6630**	**0.5761**	0.5432	0.5592

Note: The best results are highlighted in bold.

Table 3. Overall results of FullMeSH

Method	MiP	MiR	MiF	EBP	EBR	EBF	MAP	MAR	MAF
MeSHLabeler	0.6636 (1.44e–70)	0.5946 (1.82e–72)	0.6273 (7.69e–76)	0.6624 (1.39e–66)	0.6096 (1.72e–71)	0.6214 (2.64e–73)	0.4935 (2.15e–75)	0.4420 (2.12e–83)	0.4663 (2.65e–82)
DeepMeSH	0.6808 (1.58e–66)	0.6142 (4.67e–75)	0.6458 (2.25e–74)	0.6793 (4.80e–65)	0.6302 (3.59e–73)	0.6407 (8.94e–72)	0.5484 (4.44e–59)	0.5171 (8.42e–66)	0.5323 (1.01e–64)
FullMeSH	**0.7021**	**0.6363**	**0.6676**	**0.7017**	**0.6521**	**0.6629**	**0.5739**	**0.5456**	**0.5594**

Note: The best results are highlighted in bold.

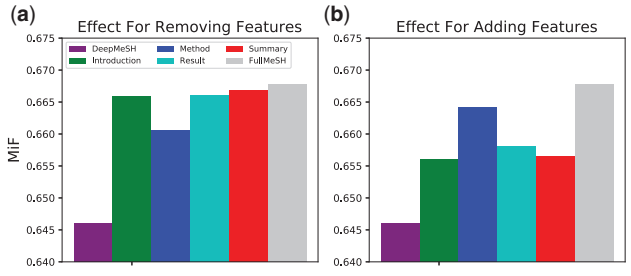

Fig. 3. Performance comparison among different sections (except Title and Abstract Sections)

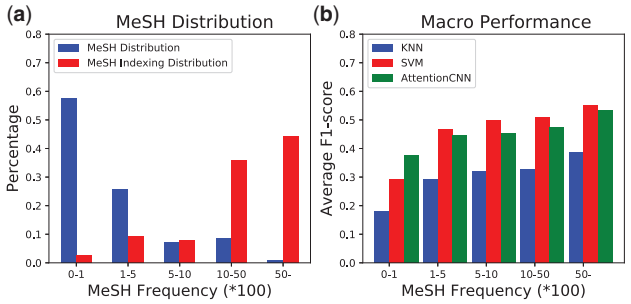

Fig. 4. Performance comparison of AttentionCNN with KNN and SVM, over groups of MHs with different frequencies

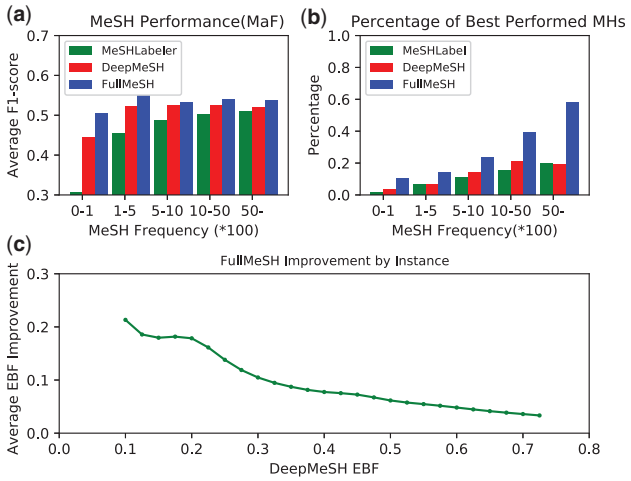

Fig. 5. Performance comparison from the viewpoints of MHs and articles

improvement. Note that the baseline is DeepMeSH which uses the information from Title and Abstract only. Figure 3a shows that the removal of Method (shown by blue) reduced MiF most, implying that Method plays an important factor for FullMeSH, while removing other sections from our model results in only a slight difference.

Figure 3b shows the performance obtained by adding the evidence (scores) generated from each section to our model individually. We can see that Method (shown in blue) achieved the best performance, implying that Method is the most important once again. However, this figure shows that every section contributed to some improved score of MiF, implying that every section is useful. Another finding is that our final model of FullMeSH with the information from all sections performed best.

4.5.3 AttentionCNN performed well on infrequent MHs

We first used Title and Abstract only to evaluate the performance of different models of FullMeSH in terms of MiF. The results show that AttentionCNN achieved MiF of 0.6015, which is slightly worse than $SVM_{D2V\text{-}TFIDF}$ of 0.6238 but much better than $KNN_{D2V\text{-}TFIDF}$ of 0.4815.

We examined the performance of AttentionCNN for MHs with different frequencies [frequency means the number of occurrences of MHs in the training corpus of abstracts (1 380 968 articles)].

We divided MHs into five groups: [0, 100), [100, 500), [500, 1000), [1000, 5000) and [5000, 1 380 968], where [0, 100) means that the number of occurrences of MHs in this group is between 0 and 100. Figure 4a shows the distributions of MHs and indexing in the training corpus. It is clear that the number of occurrences of MHs follows a long-tail distribution. That is, only 1% of all MHs which have more than 5000 occurrences contribute to more than 40% of indexing.

We then compared the performance of AttentionCNN with SVM (of using D2V-TFIDF: $SVM_{D2V\text{-}TFIDF}$) and KNN (of using D2V-TFIDF: $KNN_{D2V\text{-}TFIDF}$) in each group of different frequency. Figure 4b shows the result of Average F1-score for the three competing methods. Interestingly, this figure clearly shows that AttentionCNN outperformed $SVM_{D2V\text{-}TFIDF}$ and $KNN_{D2V\text{-}TFIDF}$ for the group of infrequent MHs (frequency ¡ 100). This result indicates that the attention mechanism based on prior label information works well for infrequent MHs. Another interesting finding is that although $SVM_{D2V\text{-}TFIDF}$ outperformed AttentionCNN for the group with frequent MHs (frequence > 100), the difference is not necessarily increasing clearly. In particular, for the top 4 most frequent MHs (*Human*, *Male*, *Female*, *Animals*), which appear more than 200 000 times in the training set, AttentionCNN outperformed SVM in all four MHs. This result highlights the advantage of deep learning-based methods, which is useful also for large training data. Also this result indicates the performance of AttenionCNN could be further improved as the data size of full text is increased.

4.5.4 Performance comparison over different frequencies in MeSHs and instances

Figure 5a shows the average F1-score of each of the three competing methods, according to the groups separated by the frequency of MHs. Moreover, Figure 5b shows the percentage of best performed MHs (removing ties) in each group by each method. These two figures show that FullMeSH outperformed DeepMeSH and MeSHLabeler in every MH group, meaning that the information from full-text is useful for indexing most of the MHs, regardless of the frequency of MHs.

Furthermore, we compared the performance with respect to instances (articles). Figure 5c shows the average EBF improvement of FullMeSH over DeepMeSH. As the EBF of DeepMeSH was smaller, FullMeSH showed more improvement. Even for the already-well predicted instances by DeepMeSH, FullMeSH could also improve the prediction slightly.

4.5.5 Performance comparison against check tags

We computed the F1-score of FullMeSH and DeepMeSH for all Check Tags which occur more than 50 times in our test set. We also computed that of $SVM_{D2V\text{-}TFIDF}$ for each of the five sections (provided in the Supplementary Materials). Out of 20 Check Tags, FullMeSH achieved the highest F1-scores in all cases, with an average score of 0.7791. This is 4.7% higher than that of DeepMeSH (0.7444). Due to the space limitation, we showed the top five Check Tags here and the complete and detailed result is given in the Supplementary Material.

Table 4 shows the results for five Check Tags, where FullMeSH outperformed significantly all other methods in all five Check Tags. Not only the performance but also the difference of the performance is high, implying the statistical significance in the performance advantage.

4.6 Computational efficiency

For training all classifiers and the LTR model, we used a cluster server with six nodes. Each node was equipped with two Intel

Table 4. Performance (F1-score) comparison over top five Check Tags. Asterisk denotes the highest F1-score by each section using SVM

Method	$SVM_{T\&A}$	SVM_{Method}	DeepMeSH	FullMeSH
Humans	0.9319	0.9332*	0.9448	**0.9581**
Male	0.8270	0.8408*	0.8453	**0.8833**
Female	0.8411	0.8571*	0.8630	**0.8938**
Animals	0.8855	0.8980*	0.9159	**0.9431**
Middle aged	0.7852*	0.7827	0.8035	**0.8259**
Average	0.8541	0.8624*	0.8745	**0.9008**

Note: The best results are highlighted in bold.

* denotes the highest F1-score by each section using SVM.

XEON E5-4650 2.7 GHz CPUs and 128 GB RAM. Learning all binary classifiers for MHs and FullMeSH took around 5 days. Moreover, we trained our AttentionCNN using 3 Nvidia Gerforce GTX 1080Ti GPU, spending about 10 h for each section. In contrast, both MeSHLabeler and DeepMeSH were trained with a server with four Intel XEON E5-4650 2.7 GHz CPUs and 128 GB RAM, which took around 5 and 7 days, respectively. We can see that FullMeSH needs much more computational resource to deal with full text. Note that the computational cost of FullMeSH can be significantly reduced if we focus on important sections, such as Abstract Section and Section 3.

5 Discussion

Automatic MeSH indexing is becoming increasingly important. Most of the existing approaches, including the state-of-the-art DeepMeSH, have used only the titles and abstracts of articles. In reality, manual indexing is, however, done by reading the full text of articles. This is because the full text of the article is too lengthy to generate a good representation and it would contain more noise. In particular, FullMeSH achieved 3.3% improvement over DeepMeSH in MiF using all data of PMC Open Access Subset data. Experimental results demonstrated that the use of information from full text is useful in every step by (i) increasing the coverage of a candidate list, which pushes the performance upper-bound; (ii) greatly improving the ranking performance of MeSHRanker by using information of each section and also a cutting-edge deep learning technique and (iii) improving MeSHNumber in predicting the number of MHs as output.

The superiority of FullMeSH also results from the learning-to-rank framework, which can efficiently and effectively integrate multiple types of data sources (such as various sections), models (such as KNN, SVM, pattern matching and AttentionCNN) and representations (such as TFIDF, D2V and W2V), for better performance. The diverse evidence generated from these data sources, models and representations is complementary to each other, and thus crucial to the good performance of FullMeSH (Table 2). An interesting future work would be estimating the upper limit of the accuracy by automatic MeSH indexing.

Funding

This research was supported by NIH Intramural Research Program, National Library of Medicine (ZL). S.Z. is supported by National Natural Science Foundation of China (No. 61572139 and No. 61872094) and Shanghai Municipal Science and Technology Major Project (No. 2017SHZDZX01). S.D. and R.Y. are supported by the 111 Project (NO. B18015), the key project of Shanghai Science and Technology (No. 16JC1420402), Shanghai Municipal Science and Technology Major Project (No. 2018SHZDZX01) and ZJLab. H.M. has been supported in part by JST ACCEL [grant number JPMJAC1503], MEXT Kakenhi [grant numbers 16H02868 and 19H04169], FiDiPro by Tekes (currently Business Finland) and AIPSE by Academy of Finland.

Conflict of Interest: none declared.

References

Aronson,A. *et al.* (2004) The NLM indexing initiative's medical text indexer. *Stud. Health Technol. Inform.*, **107**, 268–272.

Bahdanau,D. *et al.* (2014) Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv: 1409.0473*.

Chang,C.-C. and Lin,C.-J. (2011) LIBSVM: a library for support vector machines. *ACM Trans. Intel. Syst. Technol.*, **2**, 1–27.

Chen,T. and Guestrin,C. (2016) *XGBoost: A Scalable Tree Boosting System*. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, pp. 785–794.

Comeau,D.C. *et al.* (2019) PMC text mining subset in BioC: about three million full-text articles and growing. *Bioinformatics* **35**, 3533–3535.

Demner-Fushman,D. and Mork,J.G. (2015) Extracting characteristics of the study subjects from full-text articles. In: *American Medical Informatics Association Annual Symposium, San Francisco, CA, USA*, Vol. **2015**, p. 484.

Fan,R.-E. *et al.* (2008) LIBLINEAR: a library for large linear classification. *J. Mach. Learn. Res.*, **9**, 1871–1874.

Gay,C.W. *et al.* (2005) Semi-automatic indexing of full text biomedical articles. In: *American Medical Informatics Association Annual Symposium, Washington, DC, USA*, Vol. 2005, pp. 271–275.

Gu,J. *et al.* (2013) Efficient semisupervised MEDLINE document clustering with MeSH-semantic and global-content constraints. *IEEE Trans. Cybernetics*, **43**, 1265–1276.

Huang,M. *et al.* (2011) Recommending MeSH terms for annotating biomedical articles. *J. Am. Med. Inform. Assoc.*, **18**, 660–667.

Jiang,J. and Zhai,C. (2007) An empirical study of tokenization strategies for biomedical information retrieval. *Inform. Retrieval*, **10**, 341–363.

Jimeno-Yepes,A.J. *et al.* (2013) MeSH indexing based on automatically generated summaries. *BMC Bioinformatics*, **14**, 208.

Jin,Q. *et al.* (2018) AttentionMeSH: simple, effective and interpretable automatic MeSH indexer. In: *BioASQ@EMNLP*. Brussels, Belgium, pp. 47–56.

Le,Q. and Mikolov,T. (2014) *Distributed Representations of Sentences and Documents*. In: *Proceedings of the 31th International Conference on Machine Learning, Beijing, China*, pp. 1188–1196.

Liu,J. *et al.* (2017) *Deep Learning for Extreme Multi-Label Text Classification*. In: *SIGIR*. ACM, Shinjuku, Tokyo, Japan, pp. 115–124.

Liu,K. *et al.* (2015) MeSHLabeler: improving the accuracy of large-scale MeSH indexing by integrating diverse evidence. *Bioinformatics*, **31**, i339–i347.

Lu,Z. *et al.* (2009) Evaluation of query expansion using MeSH in PubMed. *Inform. Retrieval*, **12**, 69–80.

Mao,Y. and Lu,Z. (2013) Ncbi at the 2013 bioasq challenge task: Learning to rank for automatic mesh indexing. *Microsoft Research Technical Report MSR-TR-2010–82*.

Mao,Y. and Lu,Z. (2017) MeSH Now: automatic MeSH indexing at PubMed scale via learning to rank. *J. Biomed. Seman.*, **8**, 15.

Mikolov,T. *et al.* (2013) *Distributed Representations of Words and Phrases and Their Compositionality*. In: *NIPS*. Lake Tahoe, Nevada, USA, pp. 3111–3119.

Mork,J. *et al.* (2013) The NLM medical text indexer system for indexing biomedical literature. In: *Proceedings of the First Workshop on Bio-Medical Semantic Indexing and Question Answering, a Post-Conference Workshop of Conference and Labs of the Evaluation Forum 2013, Valencia, Spain.*. Volume 1094. Available at: http://ceur-ws.org/Vol-1094/bioasq2013_submission_3.pdf.

Mork,J. *et al.* (2014) Recent enhancements to the NLM medical text indexer. *Working Notes for CLEF 2014 Conference, Sheffield, UK*, Volume 1180, pp. 1328–1336.

Mork,J. *et al.* (2017) 12 years on–is the NLM medical text indexer still useful and relevant? *J. Biomed. Seman.*, **8**, 8.

Mullenbach,J. *et al.* (2018) *Explainable Prediction of Medical Codes from Clinical Text*. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, Louisiana, USA*, Vol. **1**, pp. 1101–1111.

Peng,S. *et al.* (2016) DeepMeSH: deep semantic representation for improving large-scale MeSH indexing. *Bioinformatics*, **32**, i70–i79.

Sayers,E.W. *et al.* (2019) Database resources of the national center for biotechnology information. *Nucleic Acids Res.*, **47**, D23–D28.

Stokes,N. *et al.* (2009) Exploring criteria for successful query expansion in the genomic domain. *Inform. Retrieval*, **12**, 17–50.

Tsatsaronis,G. *et al*. (2015) An overview of the BioASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, **16**, 138.

Tsoumakas,G. *et al*. (2013) Large-scale semantic indexing of biomedical publications. In: *Proceedings of the First Workshop on Bio-Medical Semantic Indexing and Question Answering, a Post-Conference Workshop of Conference and Labs of the Evaluation Forum 2013, Valencia, Spain*.

Xun,G. *et al*. (2019) Meshprobenet: a self-attentive probe net for mesh indexing. *Bioinformatics*, **35**, 3794.

Yang,Z. *et al*. (2016) *Hierarchical Attention Networks for Document Classification*. In: *The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA*, pp. 1480–1489.

Zhu,S. *et al*. (2009) Enhancing MEDLINE document clustering by incorporating MeSH semantic similarity. *Bioinformatics*, **25**, 1944–1951.