Annotating PubMed Abstracts with MeSH Headings using Graph Neural Network

Faizan E Mustafa
QUIBIQ GmbH
faizan.e.mustafa@quibiq.de

Rafika Boutalbi
Universität Stuttgart
boutalbi.rafika@gmail.com

Anastasiia Iurshina
Universität Stuttgart
anastasiia.iurshina@ipvs.uni-stuttgart.de

Abstract

The number of scientific publications in the biomedical domain is continuously increasing with time. An efficient system for indexing these publications is required to make the information accessible according to the user's information needs. Task 10a of the BioASQ challenge aims to classify PubMed articles according to the MeSH ontology so that new publications can be grouped with similar preexisting publications in the field without the assistance of time-consuming and costly annotations by human annotators. In this work, we use Graph Neural Network (GNN) in the link prediction setting to exploit potential graph-structured information present in the dataset which could otherwise be neglected by transformer-based models. Additionally, we provide error analysis and a plausible reason for the substandard performance achieved by GNN. The source code is available on the GitHub.[1]

1 Introduction

Many scientific publications are available on the internet, and the number of publications is continuously increasing with time. The digital library PubMed[2] currently consists of 33 million citations and is based on the medical database MEDLINE. The articles available on PubMed are indexed with concepts that come from the Medical Subject Headings (MeSH) ontology. The human and financial effort needed to keep up with the rapid pace of development is steadily increasing (You et al., 2020). There was a 5% increase in the number of citations in 2018 for MEDLINE. Moreover, these citations are manually indexed with MeSH headings, which cost $9.4 per citation on average.

A large-scale biomedical semantic indexing task (10a) in the BioASQ[3] challenge is designed to help develop systems that can automatically index PubMed publications using headings from the MeSH ontology[4]. The fact that a publication can be assigned more than one MeSH heading makes it a multi-label classification task. Additionally, there are approximately 30k MeSH headings which makes it an extreme multi-label classification task.

GNN has been shown to achieve unprecedented performance on the benchmarks of link prediction and recommender systems (Ying et al.). A considerable amount of real-world datasets contains latent graph-structured information that could be effectively exploited to improve performance by modeling the task as a graph-related task. The models proposed in previous versions of the BioASQ challenge do not formulate the problem as GNN modeling, which can curtail the performance gain due to the omission of crucial graph-structured information present in the dataset.

Task 10a of the BioASQ challenge is to assign MeSH headings to PubMed articles based on the title and abstract of each article. In this work, we work on the following points to solve the problem.

- We formulate the problem as GNN link prediction task to improve the performance by utilizing the information present in the graph structure.
- We provide error analysis and highlight limitations of the GNN model in order to understand the potential reasons for its inability to perform better than the baseline.

2 Literature Review

The methods used for the task in the previous versions of the BioASQ challenge can be classified into three categories (You et al., 2020). The first category named *Binary relevance* consists of models such as MetaLabeler (Tsoumakas et al., 2013)

[1]GitHub Repository
[2]PubMed Website
[3]BioASQ Website
[4]MeSH Tree View

which uses linear SVMs in a one vs all multi-label classification setting to select the most probable MeSH headings. The second category consists of models like DeepMesh (Peng et al., 2016) and MeSH Now (Mao and Lu, 2017) which rely on the widely used Information Retrieval technique named *Learning to Rank* in order to obtain the most relevant MeSH headings. The final category is based on *Deep Learning* e.g. MeSHProbeNet (Xun et al., 2019) and AttentionMeSH (Jin et al., 2018) which uses RNN and attention mechanism to get the most probable MeSH headings.

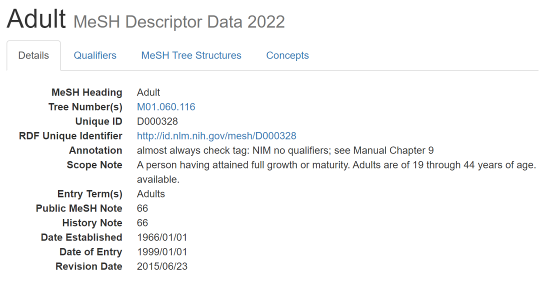
Most teams in the 2021 version of the BioASQ challenge relied on contextualized language models, such as BERT (Devlin et al., 2019). The top performing model BERTMeSH You et al. (2020) also uses BERT as a foundational model.

3 Methodolgy

Our proposed GNN model is implemented in the link prediction setting consisting of two modules, namely, *Document Embedding Module* and *Link Prediction Module*.

3.1 Document Embedding Module

We use Sentence-BERT Reimers and Gurevych (2019) to create embedding for the abstract of an article (article embedding). Sentence-BERT is used to make embeddings for the MeSH headings (heading embedding) also by using the "Scope Note". An example of the MeSH heading named *Adult* is shown in Figure 1. Both article and heading embeddings can then be used to initialize the GNN model nodes in the *Link Prediction Module*.



Adult MeSH Descriptor Data 2022

MeSH Heading	Adult
Tree Number(s)	M01.060.116
Unique ID	D000328
RDF Unique Identifier	http://id.nlm.nih.gov/mesh/D000328
Annotation	almost always check tag: NIM no qualifiers; see Manual Chapter 9
Scope Note	A person having attained full growth or maturity. Adults are of 19 through 44 years of age. available.
Entry Term(s)	Adults
Public MeSH Note	66
History Note	66
Date Established	1966/01/01
Date of Entry	1999/01/01
Revision Date	2015/06/23

Figure 1: Metadata for heading *Adult*

3.2 Link Prediction Module

Each article is annotated with MeSH headings by the human annotators. The task is to build a model that can predict MeSH headings for new unannotated articles. We formulate the problem as a link prediction between the article and heading nodes in a graph. A GNN model will be used in an inductive setting Veličković et al. (2018) to predict existence/non-existence of links between articles and MeSH headings.

The proposed GNN model consists of an encoder and a decoder. We use SAGEConv layer of the GraphSAGE Hamilton et al. (2017) to create our model. The encoder takes a graph which has two types of nodes, namely, article nodes and the heading nodes initialized by the corresponding embedding obtained from *Document Embedding Module* as described in the previous section. In inductive learning, we need to have three distinct graphs for training, validation, and test sets as described in the section 3.3. The edges in the graph are split into *message-passing* and *supervision* edges. Message-passing edges are used to update the node's embedding using neighborhood aggregation, whereas the existence/non-existence of link should be predicted for supervision edges. The output of the encoder is a graph with new low-dimensional embeddings obtained by using neighborhood aggregation based on message-passing edges. The updated node embedding x'_i for a node i is obtained using neighborhood aggregation as follows

$$x'_i = W_1 x_i + W_2 \cdot \mathrm{mean}_{j \in \mathcal{N}(i)} \, x_j \tag{1}$$

Where W_1 and W_2 are trainable parameters, x_i the current node embedding for node i and $\mathcal{N}(i)$ are neighbors of node i.

In order to determine if there is a link between two nodes x_i and x_j as specified by the supervision edges, the decoder uses the inner product between the node's output embeddings followed by a sigmoid activation function.

$$\sigma(x_i \cdot x_j) = \frac{1}{1 + e^{-x_i \cdot x_j}} \tag{2}$$

The result of sigmoid indicates a presence or absence of a link between two nodes of the supervision edges.

3.3 Graphs Construction

We have described the training, validation and test graphs in Table 1. All graphs have the same number of headings nodes, but they differ in the number of article nodes. An edge between the article and the heading node can be made if the article is annotated with a particular MeSH heading. The edges which are present in the graphs are referred to as positive

Graph	Edge Type	Description
Train	Message-passing	60% of positive train edges
	Supervision	40% of positive train edges + negative edges
Validation	Message-passing	“Borrowed graph” edges
	Supervision	All possible validation edges
Test	Message-passing	“Borrowed graph” edges
	Supervision	All possible test edges

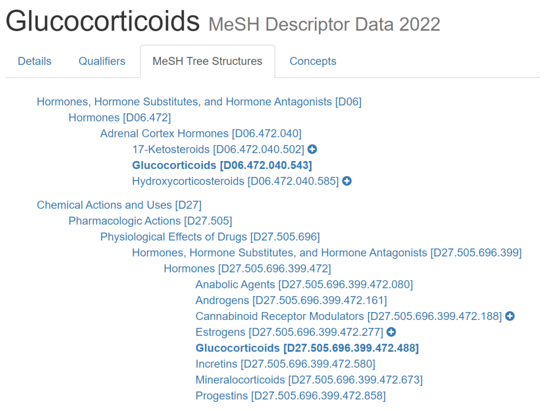
Table 1: Description of edge types for data splits.

edges. We can split the positive edges into message-passing and supervision edges according to some ratio, e.g. 60/40. We split the positive edges for train graph. However, the test set will not have any edges which could be split, as there are no links between articles and MeSH headings. The lack of message-passing edges is problematic because the GNN model needs them for neighborhood aggregation. This could be handled by making edges between all articles and MeSH headings and using them as message-passing and supervision edges. However, the fact that the message-passing edges should be the correct edges and not randomly created will result in a nonrobust model. Therefore, we extract a sub-graph from the train graph named “borrowed graph” by randomly selecting some articles nodes in the train graph. The number of articles nodes to be extracted is treated like a hyper-parameter (40k in our case). As the heading nodes in all the graphs are similar, and we know the correct edges between article and heading nodes in “borrowed graph”, we can add it to the test graph so that we have correct edges from “borrowed graph” which could be used as message-passing edges.

In the test graph, the supervision edges are all possible edges between the article nodes of test set and heading nodes. In addition to positive edges in the train graph, supervision edges also contain randomly sampled negative edges, i.e. edges which are not present in the graph. They are included in order to improve the ability of the model in terms of preventing false positive predictions.

4 Dataset

The dataset provided by the organizers of the 2022 version of the BioASQ challenge for task 10a is composed of articles obtained from PubMed. The training dataset consists of 16,218,838 articles and 29,681 distinct MeSH headings. MeSH headings are the concepts that are part of the MeSH ontology, which makes it easy to index and search medical and health-related information. Each article is assigned 12.68 MeSH headings on average. Each human-annotated MeSH heading has a unique ID assigned to it, which needs to be predicted for each article. An example of the MeSH heading named *Adult* is shown in Figure 1 where the heading is described by “Scope Note”. The test set provided by the challenge organizers for the first week containing 9659 samples is used for testing.



Glucocorticoids MeSH Descriptor Data 2022

Hormones, Hormone Substitutes, and Hormone Antagonists [D06]
    Hormones [D06.472]
        Adrenal Cortex Hormones [D06.472.040]
            17-Ketosteroids [D06.472.040.502]
            Glucocorticoids [D06.472.040.543]
            Hydroxycorticosteroids [D06.472.040.585]

Chemical Actions and Uses [D27]
    Pharmacologic Actions [D27.505]
        Physiological Effects of Drugs [D27.505.696]
            Hormones, Hormone Substitutes, and Hormone Antagonists [D27.505.696.399]
                Hormones [D27.505.696.399.472]
                    Anabolic Agents [D27.505.696.399.472.080]
                    Androgens [D27.505.696.399.472.161]
                    Cannabinoid Receptor Modulators [D27.505.696.399.472.188]
                    Estrogens [D27.505.696.399.472.277]
                    Glucocorticoids [D27.505.696.399.472.488]
                    Incretins [D27.505.696.399.472.580]
                    Mineralocorticoids [D27.505.696.399.472.673]
                    Progestins [D27.505.696.399.472.858]

Figure 2: MeSH hierarchy for *Glucocorticoids*

MeSH headings are categorized into 16 categories, which are further divided into subcategories. Each subcategory has a hierarchical depth of up to 13 where headings are organized from general to specific[5]. One important property of the MeSH hierarchy is that it is a graph instead of a tree. In a tree, each node can have only one parent, which does not hold true in the case of the MeSH hierarchy. Figure 2 shows that the MeSH heading named *Glucocorticoids* has 2 parent nodes, namely *Adrenal Cortex Hormones* and *Hormones*.

5 Experimental Setup

Taking into consideration the large size of the dataset, 70k articles are randomly sampled from the original training set to be used as the training set. Additionally, the validation set of 10k samples is sampled from the original training set. The random sampling of a small subset of articles could lead to a training dataset that has a considerably different distribution than the original dataset, resulting in non-generalizable results. We tried to mitigate that by sampling the training articles using the MeSH ontology, which is described further in Appendix A. However, there was no improvement observed over the random sampling. Therefore, we report results on randomly sampled training data to keep the method intelligible.

[5]MeSH Tree Structures

		P_{micro}	R_{micro}	$F1_{micro}$
BERTMeSH	Val	0.584	0.397	0.473
	Test	0.628	0.399	0.488

Table 2: Results obtained for BERTMeSH

Loss Function	Train	Valid	Test
Binary Cross Entropy	$\begin{bmatrix} 1018551(TN) & 31449(FP) \\ 45900(FN) & 244452(TP) \end{bmatrix}$	$\begin{bmatrix} 2035800 & 74610 \\ 204 & 1086 \end{bmatrix}$	$\begin{bmatrix} 2033029 & 77574 \\ 155 & 942 \end{bmatrix}$
Focal Loss	$\begin{bmatrix} 1041248 & 8752 \\ 116308 & 174044 \end{bmatrix}$	$\begin{bmatrix} 2089116 & 21294 \\ 551 & 739 \end{bmatrix}$	$\begin{bmatrix} 2081154 & 29449 \\ 392 & 705 \end{bmatrix}$

Table 3: Results reported as confusion matrix for GNN

The best-performing model of the 2021 version of BioASQ challenge is used as a baseline model. The model is trained for 5 epochs with an initial learning rate of 1e-5 which is altered using the learning rate scheduling function *get_cosine_schedule_with_warmup* from transformers library (Wolf et al., 2020).

Unlike BERTMeSH, the results on validation and test datasets for GNN are based on only the first 100 articles of both datasets, for computational resource reasons (test graphs for the remaining articles can be made for evaluation as explained in section 3.3). The architecture of GNN is composed of 2 SAGEConv layers where the input, hidden and output dimensions are 768, 256, 128 respectively. GNN is trained with a learning rate of 0.005 and Adam optimizer Kingma and Ba (2015) with the default hyperparameters. Two models are trained using different loss functions, namely, Binary Cross Entropy and Focal Loss. The hyperparameters used for Focal Loss are $\alpha = 0.2$ and $\gamma = 0.2$.

6 Results

Table 2 shows the results obtained for BERTMEsH on micro-averaged precision, recall, and f1 score. The model was able to score 0.488 f1 score. The results for GNN are reported as a confusion matrix in Table 3 because the f1 score is very low and is, therefore, not helpful in understanding the results. When Binary Cross Entropy is used as a loss criterion, the number of FN predictions (155) is considerably low as desired. However, the number of FP predictions is large. In the case of Focal loss, the loss criterion helps to reduce the number of FP predictions from 77574 to 29449 for the test dataset. However, the number of FN predictions increased from 155 to 392 accordingly.

7 Error Analysis

The results obtained using the focal loss indicate that the number of False Positive predictions can be improved using methods that give more importance to hard negatives. The negative edges which are difficult to discern from the positive edges are called hard negatives. Therefore, we assumed that the creation of hard negative samples improves the FP results and used *Dynamic Random Sampling* Zhang et al. (2013) and *mixup* Zhang et al. (2018) to add hard negatives during the training process instead of randomly sampling negative edges.

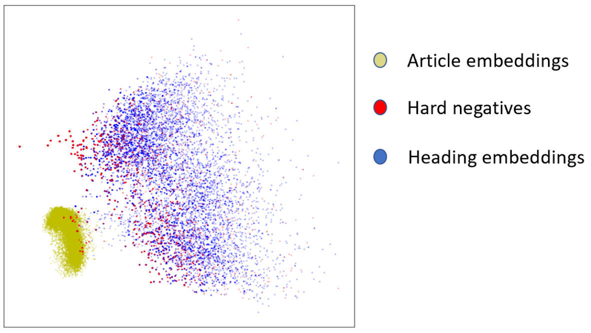


Figure 3: Hard negatives created using Dynamic negative sampling

For Dynamic Random Sampling, we start adding the hard negatives after second epoch. For each article, 5 random negatives and up to 10 hard negatives are added. Negative edges for which the dot product is too high (FP) are ignored in order to avoid the hardest negatives. To this end, negative edges which have dot product between 0.6 and 0.95 are considered hard negatives. The 2-dimensional representation of the embeddings obtained at the output layer of GNN model is shown in Figure 3. It can be observed that the hard negatives are closer to article embeddings in vector space as compared to the embeddings of remaining headings. To empirically verify our observation, we calculated the cosine similarity between the mean of article embeddings and the mean of hard negatives, which turns out to be -0.14. Similarly, a cosine similarity of -0.75 was obtained between the mean of article embeddings and heading embeddings. Although the model correctly selects the hard negatives as indicated by cosine similarity, the results obtained on the evaluation metrics do not surpass the results obtained using Focal Loss only.

The second approach *mixup* uses a linear interpolation of the positive and negative samples to create hard negatives. The following equation is used to linearly interpolate article embedding e_p and heading embedding e_n to create hard negative e_h.

$$e_h = \lambda e_p + (1 - \lambda) e_n \tag{3}$$

We set λ equal to 0.9 for the experimentation. This approach also yields no improvements over the results obtained using Focal Loss only.

8 GNN Limitations

Although GNN has improved performance on many tasks which benefit from graph-structured data, the architecture of GNN has some inherent limitations. One of the problems that Neural Networks has is that the performance is decreased as the number of layers is increased. The vanishing gradient coerces us to limit the number of layers, resulting in a shallow network that is not able to generalize. In addition to the vanishing gradient problem, the GNN model is limited to a small number of layers due to over-smoothing. Li et al. (2018) have shown that the convolution operation of GNN is the source of its predictive power, but is also the cause of its limitation. They proved that the convolution operation of GNN is a kind of Laplacian smoothing, which helps to learn new embedding from the neighboring nodes. However, the repeated application of Laplacian smoothing results in the features of all nodes being identical, which deteriorates the predictive power of the model. As the number of layers increased, the nodes in the graph increasingly have similar neighbors to update their embeddings, resulting in identical nodes.

The architecture of GNN has another limitation, named over-squashing. GNN is less effective on tasks that benefit from long-distance interactions. Equation 1 shows a node update using neighborhood aggregation for a particular layer. It can be seen that as the number of layers increases, the receptive field also grows exponentially. Therefore, the need for the model to encode information from long-distance neighbors creates a bottleneck because the model tries to cram too much information into a single vector. Alon and Yahav (2021) has shown that the information from exponentially growing k-hop neighbors for a k-layer GNN can not be crammed into a single vector representation, which results in low performance for tasks that require long-distance information. Figure 4 illustrates the bottleneck while updating a node's feature representation based on its 3-hop neighbors.

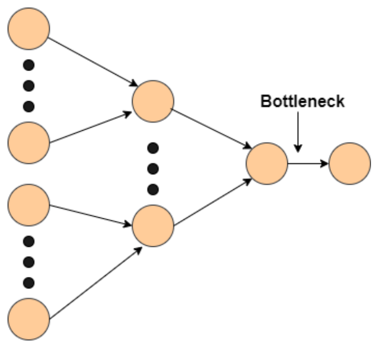


Figure 4: GNN Bottleneck (adapted from Alon and Yahav (2021)). Dots represent arbitrary number of nodes.

.

Additionally, the degree distribution of our bipartite graph follows a power law and is potentially scale-free graph (Broido and Clauset, 2019). This also forces us to cram a lot of information into high-degree nodes.

Over-smoothing and negative sampling does not seem to be the main cause of low performance in our case. The potential reason for the superior performance of transformer-based models than GNN is the mitigation of the over-squashing problem. BERTMeSH avoids over-squashing by making a unique representation for each label using Multi-label attention instead of making a single vector representation as described in the paper. This allows the model to avoid over-squashing, which leads to improved performance.

9 Conclusion

Taking into consideration, the need for an efficient system to automatically classify MeSH headings, we implemented GNN in the link prediction setting. The use of advanced negative sampling strategies did not yield improved results. We highlighted the limitations of GNN and hypothesized that GNN is not able to generalize due to the over-squashing.

Acknowledgements

This work is part of the Master Thesis done at the Universität Stuttgart. We would like to thank the examiners Roman Klinger and Steffen Staab for the evaluation of the work. We would like to extend our thanks to Felix Weil and QUIBIQ GmbH for sponsoring and their constant support of the work. Finally, we thank Juan G Diaz Ochoa and the reviewers for their valuable feedback.

References

Uri Alon and Eran Yahav. 2021. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*.

Anna Broido and Aaron Clauset. 2019. Scale-free networks are rare. *Nature Communications*, 10.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Qiao Jin, Bhuwan Dhingra, William Cohen, and Xinghua Lu. 2018. AttentionMeSH: Simple, effective and interpretable automatic MeSH indexer. In *Proceedings of the 6th BioASQ Workshop A challenge on large-scale biomedical semantic indexing and question answering*, pages 47–56, Brussels, Belgium. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR, Conference Track Proceedings*.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Yuqing Mao and Zhiyong Lu. 2017. Mesh now: automatic mesh indexing at pubmed scale via learning to rank. *Journal of Biomedical Semantics*, 8.

Shengwen Peng, Ronghui You, Hongning Wang, ChengXiang Zhai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2016. Deepmesh: deep semantic representation for improving large-scale mesh indexing. *Bioinformatics*, 32:i70 – i79.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992. Association for Computational Linguistics.

Grigorios Tsoumakas, M. Laliotis, Nikos Markantonatos, and I. Vlahavas. 2013. Large-scale semantic indexing of biomedical publications at bioasq. *CEUR Workshop Proceedings*, 1094.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Guangxu Xun, Kishlay Jha, Ye Yuan, Yaqing Wang, and Aidong Zhang. 2019. MeSHProbeNet: a self-attentive probe net for MeSH indexing. *Bioinformatics*, 35(19):3794–3802.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 974–983. ACM.

Ronghui You, Yuxuan Liu, Hiroshi Mamitsuka, and Shanfeng Zhu. 2020. BERTMeSH: deep contextual representation learning for large-scale high-performance MeSH indexing with full text. *Bioinformatics*, 37(5):684–692.

Hongyi Zhang, Moustapha Cissé, Yann Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. *ArXiv*, abs/1710.09412.

Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. SIGIR '13, page 785–788. Association for Computing Machinery.

A Data Preprocessing

Taking into consideration the large size of the dataset, we randomly filtered articles to train the models efficiently. However, random sampling could result in a dataset subset that has a considerably different distribution than the original dataset. Therefore, we also used the hierarchical structure of MeSH ontology to reduce the number of training articles.

Groups of articles are made by putting the articles into 6749 groups, where 6749 is the number of MeSH headings at depth 3 of the MeSH ontology. Some of the groups along with the number of articles they contain are shown in the table 4.

Groups	No. of articles
Adult	161367
Adolescent_Adult	43519
Treatment Outcome	19234
Adolescent_Adult_Child	17284

Table 4: Groups based on MeSH ontology

As there are numerous shared MeSH headings between articles, the groups overlap with each other. The groups which are made by the combination of two or more MeSH headings have an underscore in their name, e.g. "Adolescent_Adult" is a group that contains articles that are labeled with MeSH labels "Adolescent" and "Adult". The number of articles in the groups follows the distribution of Zipf's law, where a lot of groups have less than 10 articles. Therefore, different percentages of articles are sampled from different groups that are based on the number of articles they contain. For the groups containing articles between 10 and 200. 0.1 percent of the articles are filtered from each group. If a group contains more than 200 articles, then 0.05 percent of the articles are filtered. Finally, 50k groups are randomly sampled from the groups that have less than 10 articles. The number of filtered articles obtained after applying the previously described filtering is 400k articles. Finally, we used the training set to train the model as explained in Section 5.