Recommendation of COVID-19 articles using Deep Knowledge-Aware Network

Ivania Donoso-Guzmán
indonoso@uc.cl
Pontificia Universidad Católica de Chile
Santiago, Chile

ABSTRACT

In the last few years the number of papers that are published every day has increased enormously. Keeping up with the current state of the art has become a very difficult task for researchers. A recent work presented the use of Deep Knowledge-Aware network for paper recommendation. In this work, we expand its results by using SciBERT, BioBERT and Word2Vec as word embedding models and MeSH, MAG, and L·OVE as knowledge graphs to create the entity embeddings. Our results show that the knowledge graph does not influence the results, but the word embedding model can affect the model outcome.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

scientific paper recommendation, knowledge graph representation, deep neural networks

ACM Reference Format:
Ivania Donoso-Guzmán. 2018. Recommendation of COVID-19 articles using Deep Knowledge-Aware Network. In *RecSys '20: DCC UC IIC3633, December 14, 2020, Santiago, Chile.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Paper recommendation systems aim to help researchers find relevant work in their field. Every year the number of papers published increases: in 2016 it was estimated that it had increased 8-9% each year [5]. In the biomedical field alone it is estimated more than 1 million papers are published every year [14]. This year, with the COVID-19 pandemic the numbers of papers published in this area has seen an increase. The COVID-19 Open Research Dataset (CORD-19) [19] is a database of scientific papers on COVID-19 that has been put together by the Allen Institute for Artifical Intelligence since march 2020. It started with 29,500 on march 13th and up to the 9th of December it has gathered 377,308 papers [1]. This amount of information makes it difficult for researchers to keep up with the latest articles in their fields.

[1]More information in https://github.com/allenai/cord19

There have been efforts to create recommendation systems that could help researchers find relevant works more efficiently. These systems have mostly relied on interactions with applications (likes, bookmarks, saves) [10] and normally use a content based approach or a knowledge base graph that relies on author-paper relations. Most recently the tutorial [11] presented the use of Deep Knowledge-Aware Network (DKN) [16] for paper recommendation.

In this work we aim to expand the tutorial [11] by using different knowledge graphs, to create the entity embeddings, and also use different models to create the word embeddings. We want to understand what characteristic should have the embeddings for the tasks of user-item and item-item recommendation. The research questions can be summarized as:

- **RQ1** Are some knowledge graphs more appropriate for certain tasks than others?
- **RQ2** Do different word embeddings yield different results?

2 RELATED WORK

2.1 Scientific paper recommendation

There have been several recommendation systems created in the last few years to overcome this problem. The content based approach has been used, but most of them use bag of words to represent the content of the article. The knowledge graph based approaches use the citing net to capture the relations among researchers and papers. A thorough review of past models can be found in [1].

2.2 Knowledge Based Recommenders

Knowledge graphs, are graphs where the nodes represent entities and their characteristics, and the edges represent relations between them. In Recommender systems they have been use in content based approaches to capture how close are the entities of interest (`item-item`, `user-item`, `user-user`) [2, 15]. Recently the graph based methods have been used to compute embeddings for the relevant entities. These entities are used to feed a neural network that intelligently uses the information given by the embeddings [16–18].

3 METHODS

3.1 Recommendation algorithm

The recommendation algorithm DKN [16] creates a representation of the items taking as input words, context and entity embeddings. These embeddings are given to a convolutional neural network, called KCNN, that generates the item representation. The user

Table 1: Knowledge graphs descriptions. Mean represents the average number of entities per paper and Std is their standard deviation.

Name	Edges	Nodes	Edges types	Mean	Std
MeSH	15,978,100	5,649,273	51	18.69	7.43
MAG	745,117	54,060	13	9.52	1.43
L·OVE	50,861	38,916	4	16.91	12.45

is represented using the items the user has found relevant. An attention network computes a weight for each `(candidate, past item)`, that then is used to compute a weighted average of the past items. The user and the item representation are then passed to a neural network that predicts whether the item is relevant or not to the user. This algorithm is the same that was used in [11]. We used a modified version of the original implementation of the authors [2].

3.2 Data

We collected two kinds of data: knowledge graphs and papers metadata. We used five sources of metadata CORD-19 [19], Microsoft Academic Graph [12], MAG/CORD-19 [3], PubMed [4] and Epistemonikos [5]. These sources were combined into one dataset that has for each paper its entities in all the knowledge graphs that are used [6]. Table 2 contains the number of samples of each dataset and the percentage of completeness for each identifier.

We used three knowledge graphs: the MeSH ontology [7], the L·OVE taxonomy [8] and the extended metadata, fields of study, of Microsoft Academic Graph [9]. Table 1 describes these sources.

3.3 Models for embeddings

One of the characteristics of DKN is that the word and entity embeddings do not need to be of the same size. This makes it easy to combine different models for each part.

For the word embeddings we used three models: SciBERT [3] and BioBERT [6]. These model are BERT [4] fine-tuned with different datasets. We picked Word2Vec trained on PubMed [8] with 200 dimensions as baseline.

For the entities in the knowledge graph we use used PyTorch-BigGraph [7] to create the embeddings. This algorithm can create embeddings for very large datasets even if they do not fit in memory, like in our case. For each knowledge graph we obtained embeddings using the ComplEx [13] and the RESCAL [9] algorithms for $(32, 64, 129, 256)$ dimensions. In total we created 24 entity embedding models. The configuration files for each knowledge graph are available in https://github.com/PUC-RecSys-Class/proyecto-kgrecommender.

[2]https://github.com/indonoso/DKN

[3]https://github.com/microsoft/mag-covid19-research-examples/blob/master/src/data/releases.md

[4]https://pubmed.ncbi.nlm.nih.gov/

[5]https://es.epistemonikos.cl/

[6]Details on this procedure can be found in https://github.com/PUC-RecSys-Class/proyecto-kgrecommender

[7]https://www.nlm.nih.gov/mesh/meshhome.html

[8]https://es.epistemonikos.cl/project/love/

[9]https://www.microsoft.com/en-us/research/project/academic/

4 EXPERIMENTAL PROCEDURE

4.1 Data splits

To evaluate performance we define two tasks: *User-Item* recommendation and *Itemt-Item* recommendation. The *User-Item* dataset was created using the author and its papers references: for each paper were the user was the author we collected its references, and each one of them was assigned with a relevant label. We removed authors that had less than 10 papers and more than 100. We then splitted the data into 3 sets: training, test and validation with 70%, 20% and 10% of the data respectively. The splits were made like a time series; training has the oldest papers, validation has the papers in the middle and test has the most recent papers. On each set we sample negative items from the same time windows as follow: for each positive item we selected 4 negative samples and the most popular samples were more likely to be selected. This resulted in a training set of 7,049,091 autor-item tuples with 19.42 % of positive labels. Since the datasets for training, validation and testing where too big to fit in memory (Table 3), we decided to pick 10 random samples of 120,000 tuples for training, one sample of 120,000 for validation and one sample of 120,000 for testing.

For the Item-Item task we follow a similar procedure. The dataset was created using the references of a paper as positive items. We removed papers with less than 5 and more than 100 positive items. The training, test and validation samples were stratified on the paper id that receives the recommendations. The negative sampling procedure was the same that was used for the User-Item. A summary of the dataset is displayed in table 3.

For both tasks, once the hyperparamethers were found, we computed the threshold with the same validation set and then evaluated the results using the test set. The threshold was set to maximize f1 score of the positive class.

4.2 Use of document entity embeddings

In the article of DKN [16] the authors make emphasis on the fact the the KCNN allows to find better patterns because the words embeddings are augmented by a entity embedding if it corresponds. In this work we do not have entity embeddings associated with each word, instead we have entities associated with the document. These entities were assigned by a human and have been curated by others.

To test whether we could use the document entity embeddings as replacement of the word entity embeddings, we tested whether the predictions of the models changed when using different orders for the entities. There were three scenarios:

- Global order: the entities had one global order maintained in test and training sets. The entities of every article were sorted following this order. For example:

$$globalorder : (e_1, e_2, e_3, e_4, e_5)$$
$$a_1 : (e_1, e_2, e_3, e_4)$$
$$a_2 : (e_2, e_3, e_5)$$

- Article order: the entities had a fixed order for each article. This order was not shared across articles. In this example

Table 2: Percentage of presence of identifiers in the metadata sources. The Combined dataset columns has the statistics for the experiments dataset. A dash in a cell means that the source does not provide the identifier.

	CORD-19	MAG/CORD-19	MAG	Epistemonikos	PubMed	Combined dataset
N	288,800	301,928	154,317	120,138	103,462	38,571
DOI	0.6285	0.6131	0.9826	0.8273	0.9572	0.9656
Title	0.9997	0.9997	1.0	0.9999	0.9994	1.0
MAG id	-	0.8824	1.0	-	-	1.0
PMC id	0.3962	0.3431	0.6943	-	0.6372	0.726
PubMed id	0.5475	0.5057	0.9988	0.4016	1.0	1.0
Episte id	-	-	-	1.0	-	1.0
CORD-19 id	1.0	1.0	-	-	-	0.992
Arxiv id	0.007832	0.007035	-	-	-	0.0001556

Table 3: Datasets splits global descriptions

Task	User-Item	Item-Item
Training	7,049,091	747,540
Validation	1,606,049	215,602
Test	680,675	110,247

the e_2 appears after e_3 in a_1, but before e_3 in a_2.

$$a_1 : (e_4, e_1, e_3, e_2)$$
$$a_2 : (e_2, e_5, e_3)$$

- Random order: the entities were randomized each time they appear in the dataset. This means that with different authors a article can have different entity order.

$$[u_1, a_1 : (e_4, e_1, e_2, e_3)]$$
$$[u_2, a_1 : (e_1, e_3, e_4, e_2)]$$
$$[u_1, a_2 : (e_2, e_3, e_5)]$$
$$[u_3, a_2 : (e_3, e_5, e_2)]$$

Each scenario had 10 different trials. The Article and Random order were randomized and the Global order had 8 samples that were random and 2 where the order was according to entity frequency. The DKN model gives probability for the positive class, so to compute the threshold for each model we obtained the threshold that maximized the F1-Score.

To measure the stability of predictions we computed the entropy of the predictions for each sample in each scenario. In each scenario we had 10 predictions, if the stability was good, i.e. all the models predicted the same class, the entropy would zero but if the predictions were not stable, i.e. half of the models predicted relevant and then half predicted non-relevant, the entropy would be one. We tested only with the User-Item dataset. The hyperparameters were: BioBERT for word embeddings, 32 dimensions, complex diagonal operator, four items in history and 15 words per title.

4.3 Finding Hyperparamethers

The numbers of hyperparameters is too large to make a grid search to find the best combination. We decided to optimize the fowolling parameters, leaving the rest at fixed value: the dimension of the entity embedding (32, 64, 128, 256), the type of entity embedding model (RESCALL, ComplEx), the maximum number of words (10, 20) and the history size (5, 10, 15). We selected a random sample for the training data for each task and tested with the test dataset.

For both tasks, Item-Item and User-Item, we found consistent results:

(1) **Entity embedding size**. It did not have a significant effect. In some combinations 64 was better than 32. Since the entity size was associated with more training time, we decided to keep 64 for all the experiments
(2) **Entity embedding type**. The model used to create the entity embeddings had no significant effect. We decided to use ComplEx because it is state of the art.
(3) **Maximum number of words**. There no big differences between 20 and 10 words. Since using 10 words took less time, we decided to keep that number.
(4) **Maximum number of history**. Figure 1 displays the AUC for the different values of this hyperparameter for the Item-Item task. There were no differences for SciBERT and Word2Vec. When using BioBert, a longer history gave better results. The differences between 25 and 15 where not big, but training time was significantly higher (11.17 minutes vs ~17.03 minutes on average). For this reason we kept the value 15 for this hyperparameter.

5 RESULTS

5.1 Use of document entity embeddings

Table 4 presents the entropy for each scenario and KG. We found that the Random scenario had more stability. Its entropy was very low for MeSH and MAG, but it was significantly higher for L·OVE. The Global scenario was the one with the highest entropy on average. The Article scenario had a low entropy for MeSH but high for L·OVE and MAG. In the case of L·OVE, this could be explained by the low number of entities in the graph and the high variance on the number of entities per article. In the case of MAG we found that each article had on average 8.80 entities, even when the number of nodes was 30% higher than L·OVE. One important finding is that even though the entropy results were different among KG, all the scenarios reported similar AUC values (table 5). This could mean

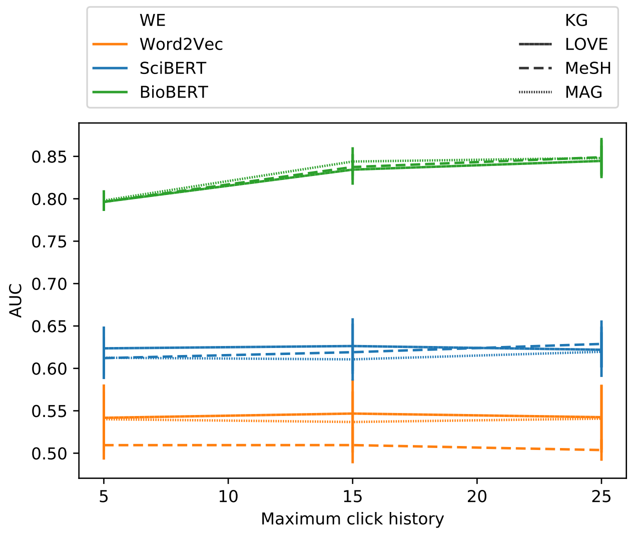

Figure 1: Mean AUC for different clicks' history sizes using the Item-Item dataset.

Table 4: Entropy obtained in each scenario for each knowledge graph

	L·OVE	MeSH	MAG
Article	0.37	0.04	0.39
Global	0.37	0.40	0.40
Random	0.39	0.04	0.04

Table 5: Average AUC obtained in each scenario for each knowledge graph

	L·OVE	MeSH	MAG
Article	0.91	0.91	0.90
Global	0.91	0.91	0.91
Random	0.91	0.91	0.91

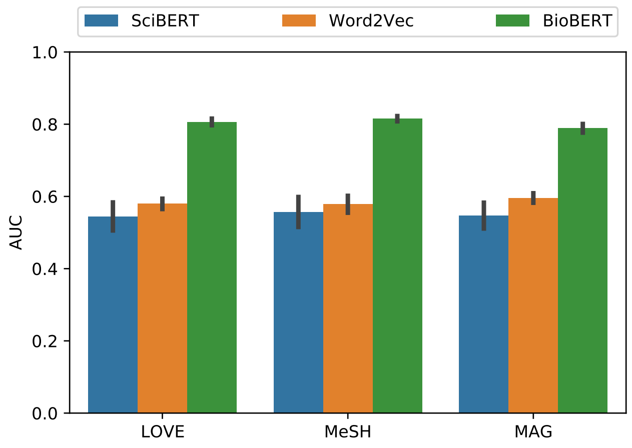

Figure 2: Item-Item AUC. The line represents one standard deviation.

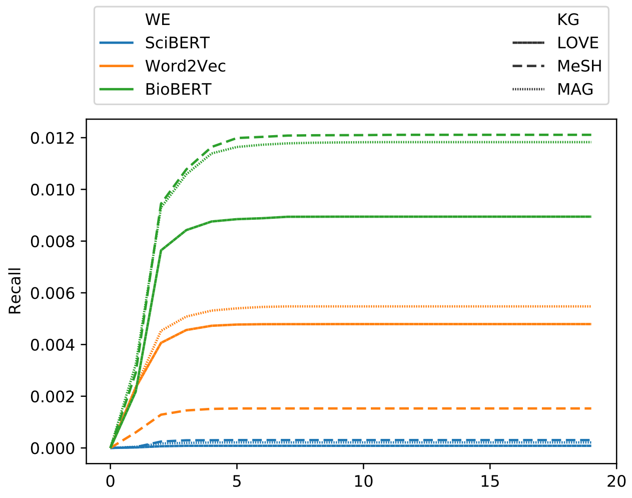

Figure 3: Item-Item Recall@n

that even though the network learned different things each time, it learned what was relevant.

5.2 Item-Item recommendation

Figures 2, 3 and 4 display the results for this task. BioBERT was the word embedding model that yielded the best AUC (figure 2), the best accumulated recall and the best recall for the relevant articles (figure 4).

The worst results were obtained by SciBERT: it has the lowest AUC for all the KG and also the lowest recall for the relevant articles (figure 4). This model has the highest recall for the non relevant articles. This suggests that when using this word embedding, the model learned more represented class.

The knowledge graph that was used did not change the results by a significant amount in any of the scenarios. The word embedding model was influencing the results the most.

5.3 User-Item recommendation

For the User-Item recommendation we found that the knowledge graph L·OVE yielded the worst results. In all its combinations with word embeddings its accumulated recall was zero.

The Word2Vec word embedding model obtained the best accumulated recall, followed by SciBERT. The best model recall was obtained by SciBERT, followed by Word2Vec. This suggests that Word2Vec was able to learn a ranking for more users, even if it did not find all the relevant articles for them.

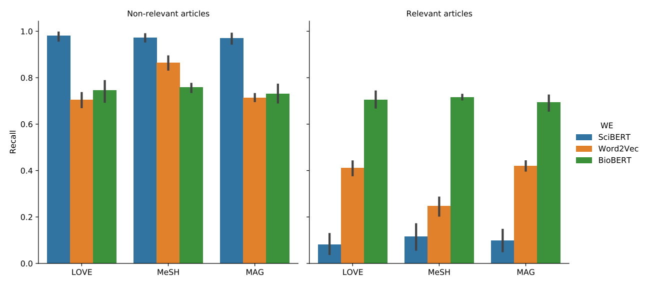

Figure 4: Model recall for relevant and non relevant classes in the Item-Item task. The line represents one standard deviation.

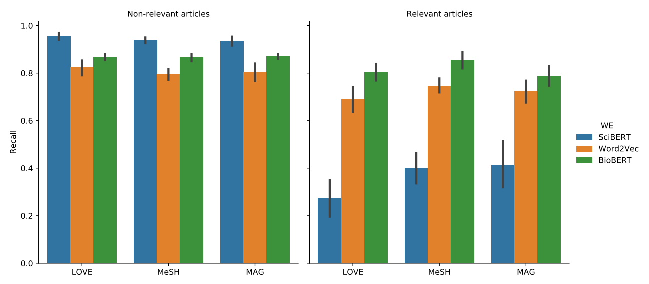

Figure 7: Model recall for relevant and non relevant classes in the User-Item task. The line represents one standard deviation.

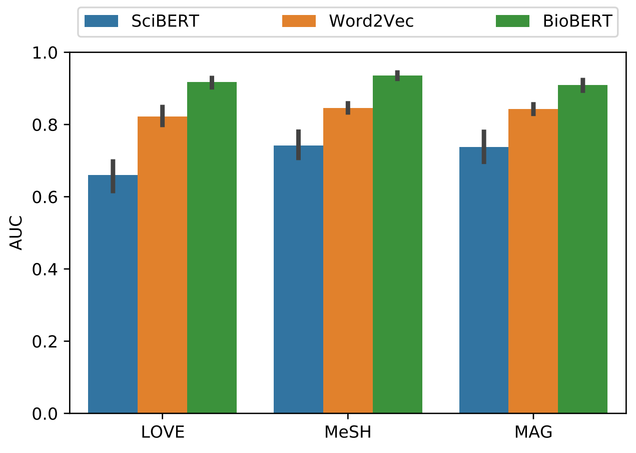

Figure 5: User-Item AUC. The line represents one standard deviation.

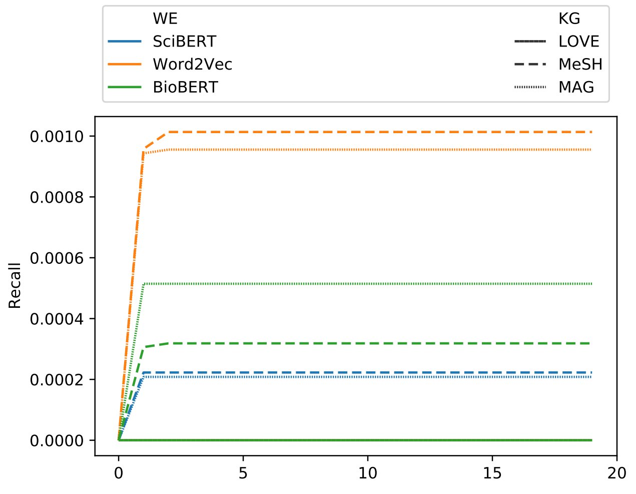

Figure 6: User-Item Recall@n

6 DISCUSSION

The results show that there are some strong differences between the word embedding models, but not between knowledge graphs.

For the Item-Item task we found that BioBERT yielded by far the best results. We did not expect that SciBERT would have such bad performance. We suspect this difference between BERT based models is due to the data they used to fine-tune. Our database had only papers that were in PubMed, so the Word2Vec and BioBERT had an advantage.

In the User-Item task, we found some mixed results. On one hand BioBERT had the best model recall, but Word2Vec had the best accumulated recall. The KG L·OVE had a model recall that was very similar to the other KG, but in accumulated it got zero as average. These differences suggest that the models that obtained best accumulated recall were probably able to predict relatively good rankings for more users than the other models.

With respect to the RQ1, we found that there is some evidence suggesting that L·OVE may not be the best for User-Item task. This evidence is not conclusive, more experiments need to be done to be able to draw definitive conclusions. Besides that, we did not find very strong differences between the different KG. We suspect this might be due to the fact that we had to reorder the entities for every article in every sample. This action might have made it difficult for the network to learn from this data.

For RQ2, we found that BioBERT was by far the best model to obtain word embeddings. Word2Vec trained with PubMed was the second best. This suggests that the embeddings created or fine-tuned with articles from the same source of the dataset had the best results. It is surprising that SciBERT got very bad results. It is possible that the percentage of papers that come from another field might have interfered too much.

The limitations of these results come from the evaluation we performed. Because of time and memory restrictions we could not train with the whole dataset in any of the tasks. The sample procedure (10 samples from training and the same test and validation sets) aimed at diminishing variance of the results, but we know that the results could change if we had trained with the whole dataset or used another procedure to select the samples. Another limitation comes from the fact that we used a recommendation algorithm that used word entity embeddings, not document entity embeddings.

We showed that the predictions were stable, but the results suggests that the network was not able to learn their differences.

7 CONCLUSION

In this work we have explored how different knowledge graphs and word embedding models can affect a recommendation task. Our results show that the knowledge graph did not have a important effect in the results and that the word embedding model was the one factor that affected the recommendation the most. BioBERT was found to be the best embedding model followed by Word2Vec.

There are several directions in which this work can be extended. The first is to replace the KCNN network by a component that can handle document entities. This component could still use CNN for the words but it should use another structure to learn the entity embeddings. Another extension is to generate a visualization to explain the results. The network has an attention layer that could be easily used to justify the results to users, so only the visualization should be created. Finally, in this work we only use titles, but all the articles have abstracts that explain more in detail the contents of the papers. Using the abstract could be another way to add more information to the network.

REFERENCES

[1] Xiaomei Bai, Mengyang Wang, Ivan Lee, Zhuo Yang, Xiangjie Kong, and Feng Xia. 2019. Scientific paper recommendation: A survey. *IEEE Access* 7 (2019), 9324–9339.

[2] J. Beel, A. Aizawa, C. Breitinger, and B. Gipp. 2017. Mr. DLib: Recommendations-as-a-Service (RaaS) for Academia. In *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. 1–2. https://doi.org/10.1109/JCDL.2017.7991606

[3] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: Pretrained Language Model for Scientific Text. In *EMNLP*. arXiv:arXiv:1903.10676

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[5] Esther Landhuis. 2016. Scientific literature: Information overload. *Nature* 535, 7612 (2016), 457–458.

[6] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* (09 2019). https://doi.org/10.1093/bioinformatics/btz682

[7] Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. 2019. PyTorch-BigGraph: A Large-scale Graph Embedding System. In *Proceedings of the 2nd SysML Conference*. Palo Alto, CA, USA.

[8] Ryan McDonald, Georgios-Ioannis Brokos, and Ion Androutsopoulos. 2018. Deep relevance ranking using enhanced document-query interactions.

[9] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data.. In *Icml*, Vol. 11. 809–816.

[10] Denis Parra and Peter Brusilovsky. 2009. Collaborative Filtering for Social Tagging Systems: An Experiment with CiteULike. In *Proceedings of the Third ACM Conference on Recommender Systems* (New York, New York, USA) *(RecSys '09)*. Association for Computing Machinery, New York, NY, USA, 237–240. https://doi.org/10.1145/1639714.1639757

[11] Iris Shen, Le Zhang, Jianxun Lian, Chieh-Han Wu, Miguel Gonzalez Fierro, Andreas Argyriou, and Tao Wu. 2020. In Search for a Cure: Recommendation With Knowledge Graph on CORD-19. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 3519–3520. https://doi.org/10.1145/3394486.3406711

[12] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In *Proceedings of the 24th International Conference on World Wide Web* (Florence, Italy) *(WWW '15 Companion)*. Association for Computing Machinery, New York, NY, USA, 243–246. https://doi.org/10.1145/2740908.2742839

[13] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML).

[14] Konstantinos Z. Vardakas, Grigorios Tsopanakis, Alexandra Poulopoulou, and Matthew E. Falagas. 2015. An analysis of factors contributing to PubMed's growth. *Journal of Informetrics* 9, 3 (2015), 592 – 617. https://doi.org/10.1016/j.joi.2015.06.001

[15] Gang Wang, XiRan He, and Carolyne Isigi Ishuga. 2018. HAR-SI: A novel hybrid article recommendation approach integrating with social information in scientific social network. *Knowledge-Based Systems* 148 (2018), 85–99.

[16] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep Knowledge-Aware Network for News Recommendation. In *Proceedings of the 2018 World Wide Web Conference* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1835–1844. https://doi.org/10.1145/3178876.3186175

[17] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-Aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 968–977. https://doi.org/10.1145/3292500.3330836

[18] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge Graph Convolutional Networks for Recommender Systems. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 3307–3313. https://doi.org/10.1145/3308558.3313417

[19] Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Michael Kinney, Yunyao Li, Ziyang Liu, William Merrill, Paul Mooney, Dewey A. Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex D. Wade, Kuansan Wang, Nancy Xin Ru Wang, Christopher Wilhelm, Boya Xie, Douglas M. Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. 2020. CORD-19: The COVID-19 Open Research Dataset. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*. Association for Computational Linguistics, Online. https://www.aclweb.org/anthology/2020.nlpcovid19-acl.1