Preliminary Experiments with Transformer based Approaches To Automatically Inferring Domain Models from Textbooks

Rabin Banjade
University of Memphis
rbnjade1@memphis.edu

Priti Oli
University of Memphis
poli@memphis.edu

Lasang Jimba Tamang
University of Memphis
ljtamang@memphis.edu

Vasile Rus
University of Memphis
vrus@memphis.edu

ABSTRACT

Domain modeling is a central component in education technologies as it represents the target domain students are supposed to train on and eventually master. Automatically generating domain models can lead to substantial cost and scalability benefits. Automatically extracting key concepts or knowledge components from, for instance, textbooks can enable the development of automatic or semi-automatic processes for creating domain models. We explore in this work the use of transformer based pre-trained models for the task of keyphrase extraction. Specifically, we investigate and evaluate four different variants of BERT, a pre-trained transformer based architecture, that vary in terms of training data, training objective, or training strategy to extract knowledge components from textbooks for the domain of intro-to-programming. We report results obtained using the following BERT-based models: **BERT**, **CodeBERT**, **SciBERT** and **RoBERTa**.

Keywords

Domain modeling, knowledge component extraction, key phrase extraction, transformer based models, Intelligent Tutoring System

1. INTRODUCTION

Computer-based adaptive instructional technologies, our focus, must have a representation of the target domain, i.e., a domain model. Hence, there is a need for domain modeling, which specifies the key knowledge components or units of knowledge that students have to master in a target domain, such as physics, biology, or computer programming. Furthermore, a domain model should include a structure that specifies the relationship among the knowledge components, typically in the form of a prerequisite knowledge structure suggesting which basic concepts must be mastered before more complex concepts that rely on the basic concepts, e.g., addition should be mastered before multiplication [14, 17, 8]. The prerequisite knowledge structure of a domain model implies a specific trajectory or trajectories towards mastery that students must follow. According to some, a domain model should also include links to related learning objects, i.e., instructional tasks, which help students practice to master the targeted concepts. More recently, A broader view of the domain model has been argued, which should include all the key concepts, skills, ideas, principles, and the values, identity, and epistemology of the community of experts or professionals active in the target domain [1].

Domain models can be developed from different information sources: experts, textbooks (written by domain and pedagogical experts), and learner performance data. Expert-driven approaches to domain modeling are expensive, time-consuming, and not very scalable within and across domains. To overcome these challenges, automated or semi-automated approaches are much needed. This work explores such novel automated methods for domain model discovery from textbooks, specifically for the target domain of intro to computer programming. In particular, we explore to what extent the process can be automated. While the proposed methods are fully automated, their output is not perfect, which means a human expert must be involved to curate the output before being used in an existing adaptive instructional system. Nevertheless, this semi-automated process is much more cost-effective and scalable than the manual approach to building domain models. It should be noted that another source of information for domain modeling for programming that has been explored in the past is code itself, e.g., using a Java parser [18, 38, 15].

There are two advantages of using textbooks to extract domain models for programming compared to using code only. First, textbooks contain both code examples and textual explanations of the concepts, which is advantageous for higher-level concepts such as sorting, which is challenging to infer from a block of code that implements it. Second, concepts extracted directly from code are difficult to interpret and are often programming language-specific constructs. A textual description can accompany the grammar of a programming language in the form of comments or descriptions, but this requires substantial additional expert involvement. Build-

ing more interpretable automated ways to extract domain models from source code is an interesting research topic beyond this paper's scope. In this work, we experiment with a pretrained transformer-based model to extract key concepts or knowledge components from computer science (intro to programming) textbooks. We use the code examples in textbooks, but we do not parse them to extract candidate key concepts but to rank the concepts extracted from the textual explanations.

Transformer-based pretrained models, which are trained on massive unlabelled text collections, have been successful in various NLP tasks such as keyphrase extraction [3, 33] which is relevant to this work. However, with the rise of many pretrained models for various specific tasks, there is a need to explore which of these pretrained models are helpful for what tasks. In this work on domain model discovery for computer programming, we explore an overgeneration-and-ranking approach for keyphrase extraction using four pretrained transformer models: BERT[11], CodeBERT[13], RoBERTa [25], and SciBERT [2]. These pretrained models vary in different aspects, such as training data and training mechanisms. For instance, BERT is trained on general domain corpora such as news articles and Wikipedia, RoBERTa is trained on a larger dataset of English language corpora consisting of books, news, web text; SciBERT is trained on papers on computer science and biomedical domain whereas CodeBERT is pretrained in NL-PL (Natural language-Programming language) pairs for multiple programming languages. In this study, we experiment with embeddings obtained from each of these methods to evaluate and compare domain-specific models such as CodeBERT and SciBERT versus models such as Roberta and BERT trained on general corpora for the tasks of keyphrase extraction and subsequently domain modeling.

The paper is organized as follows. Section 2 discusses relevant works to domain modeling. In section 3, we detail the methodology followed to generate domain models from textbooks. Section 4 presents the evaluation dataset and the metrics used to evaluate the performance of the pretrained models. A conclusion section follows the results.

2. RELATED WORK

Our work focuses on extracting key knowledge components from textbooks using embeddings obtained from transformer based pretrained models in an unsupervised manner. This section presents most relevant prior works to concept extraction, unsupervised keyphrase extraction, and pretrained models in NLP.

Concept extraction or identifying important concepts that a learner should master has been studied for various applications in the educational domain, such as concept-based textbook indexing (adaptive hyperbook for constructive teaching, Elm-art [5]), concept prediction [19], and concept hierarchy creation [36]. Concept extraction is related to keyphrase extraction, which is extracting the most important concepts in a given document. Keyphrase extraction has been explored using different approaches: rule-based, supervised, and unsupervised including deep-learning [30]. Typically, keyphrase extraction consists of two steps: candidate generation and ranking. The first step extracts key concepts based on heuristics, such as all noun phrases, while the second step ranks the extracted candidate phrases based on scoring rubric that indicates the importance of the candidate phrase for the document and/or goal. In our work, we use unsupervised embedding-based ranking methods for candidate concepts generated from sections or subtopics in chapters in intro-to-programming textbooks. We consider a subtopic as a reference document when assessing the importance of each candidate key concept.

Existing unsupervised keyphrase extraction methods can be broadly categorized as statistics based such as TF-IDF, e.g., KP-Miner [12], graph-based such as TextRank [27], SingleRank [35] and topic-based methods such as TopicRank [4] even though many of the works use a combination rather than a single approach. Recent advances in representational methods of words, phrases and documents like Word2Vec [28], Doc2vec [20], and Sent2vec [29] led to novel ranking methods for keyphrase extraction [3, 23, 33]. For instance, EmbedRank [3] uses sentence embeddings based on Doc2vec or Sent2vec to represent candidate phrases and the document in the same high dimensional vector space based on which a ranking of the candidate key concepts is obtained using the cosine similarity score between the embedding vectors of the candidate phrases and the documents. Our work is similar in the sense that we use pretrained embeddings for keyphrase extraction in the context of domain modeling.

Embeddings obtained using transformer [34] based pretrained models [31, 32, 37] have shown dramatic improvements in various tasks in different domains such as software engineering, computer vision, and education. The reason behind this improvement is the high-quality semantic representations. Models such as BERT have been trained on different domain-specific data, some examples being BioBERT [21] in biology, legalBERT [7] in legal documents, SciBERT in scientific articles, and CodeBERT on code and natural language text pairs. To the best of our knowledge, experiments on how these embeddings affect the downstream task of knowledge component extraction and consequently on domain modeling have not been done before. In this work, we experimented with embeddings obtained from four BERT models, BERT, CodeBERT, SciBERT, and RoBERTa, for knowledge component extraction and the larger task of domain modelling.

3. METHODOLOGY

Our method to extract knowledge components from intro-to-programming textbooks is based on pretrained embeddings inferred from various BERT-based models. To evaluate the automated methods, we annotated a dataset by selecting key concepts for each section in five randomly selected chapters in two textbooks. As noted, we rely on an overgeneration-and-ranking approach for key concept extraction. The embeddings are mainly used to rank the key candidate phrases. Since textbooks for intro-to-programming contain both code examples and related explanatory text, we can use bimodal pretrained models such as CodeBERT trained on both text and code. Even though we provide code to the CodeBERT model as input, the model is used only for ranking candidate keyphrases, i.e., our approach and evaluation is based on candidate concepts generated from the textual parts of the textbooks. We could consider statements, code blocks,

and code comments as sources of candidate concepts. However, we limit the scope of this work to candidate concepts from the textual part of intro to computer programming textbooks.

In this section, we explain the preprocessing steps, such as candidate concept extraction, phrase and document embedding generation, and candidate concept ranking. We also discuss the performance metrics of specificity and relevancy. Before applying methods related to candidate concept extraction, we resolve pronouns in the text to boost our knowledge component extraction methodology by resolving pronouns such as 'it',' this', and 'their' using a pretrained deep learning model [22] based on span ranking architecture.

3.1 Candidate Concept extraction

The step of candidate concept extraction consists of noun phrase extraction and filtering. We used Stanford CoreNLP Tools [1] for tokenizing, part-of-speech tagging, and noun phrase chunking. We only considered noun phrases which are unigrams (one token), bigrams (two consecutive tokens), trigrams (three consecutive tokens), and quadigrams (four consecutive tokens), for candidate generation.

3.2 Phrase and Document Embedding

From the prior step of generating 3.1 candidate phrases we obtain a tokenized form of a document D represented as $D = t_1, t_2, t_3....,t_N$ where t_n represent tokens. We also obtain a list of candidate phrases $C_0, C_1, ..., C_N$. Based on this, we then obtain contextualized embeddings for each of the tokens as shown by the Equation 1.

$$E_1, E_2, E_3....E_N = Model(t_1, t_2, t_3...t_N) \tag{1}$$

where E_n represents embeddings of each token, and the model refers to any of the four pretrained models we chose: BERT, CodeBERT, SciBERT or RoBERTa.

Each document embedding is obtained using average pooling across all the tokens of the document. Similarly, to obtain embedding for each candidate phrase from a document, we average across embeddings of tokens. Although the best pooling strategy is still an area of active research, we opted for the average pooling strategy as it has shown better performance than using the output of the first token [CLS] for different tasks[9].

3.3 Ranking

We have experimented with a number of ranking and performance metrics as presented in this section.

3.3.1 Cosine Similarity

Once the embeddings for each document and phrase are obtained, we compute the cosine similarity (normalized dot product) between the phrase and document embedding vectors. The cosine similarity scores for candidate phrases capture the semantic 'relatedness' or 'closeness' of a phrase to the underlying document. Those scores are used to rank candidate phrases.

[1]https://nlp.stanford.edu/software/tagger.shtml

3.3.2 Adjusted Maximal Marginal Relevance To Balance General and Specific Concept Ranking

Our goal is to extract all important knowledge components of a target domain. A challenge to this coverage problem is that key concepts can have different granularity levels, making it more challenging to design a single metric that ranks more general and more specific concepts highly.

For instance, in the context of a chapter focusing on the concept of 'loop', some of the important key concepts are 'for loop' or 'while loop', which are broad, high-level types of loops. Nonetheless, a more specific concept such as a nested loop or loop continuation condition is important too.

In order to extract both the general and specific concepts, candidate key phrases must be similar to the current document and less relevant to other documents. Furthermore, in order to rank higher topic-specific concepts so that top k candidates consist of more topic-specific concepts, we modified the Maximal Marginal Relevance(MMR) [6] metric, which was used initially in information retrieval and text summarization to control relevancy and diversity of retrieved documents. The embedding-based keyphrase extraction method proposed by Bennani-Smires et al.[3] used a modified version of MMR to tackle redundant key phrases from a document. In our case, we modify MMR to balance through the control parameter λ general and specific key candidate phrases.

As indicated in the equation 2, we balance the similarity of a candidate phrase C_i to the current document *doc*, which is captured by the first term on the right-hand side of the equation, with its similarity with other documents which is indicated by the second term that represents the similarity between the candidate phrase C_i from all the other N documents in the corpus.

If a candidate phrase is highly relevant to a document other than the current document, the similarity measure with the current document is penalized, indicating the term is not specific enough to the current document.

$$\begin{aligned} \mathrm{MMR}_{\text{adjusted}} := \arg\max_{C_i \in K} \Big[& \lambda \cdot \mathrm{sim}_1(C_i, \mathrm{doc}) \\ & - (1-\lambda) \max_{doc_j \in N \setminus doc} \mathrm{sim}_2(C_i, doc_j) \Big] \end{aligned} \tag{2}$$

4. EXPERIMENT AND RESULTS

As already noted, we experimented with four pretrained models and evaluated their performance on a dataset that we built.

4.1 Dataset

The evaluation dataset was built based on sections from two intro-to-programming textbooks, "Introduction to JAVA programming" [24] and "JAVA How to program" [10]. From each of the two textbooks, we randomly selected 20 sections that focus on some specific concepts such as 'while-loop,' 'sorting arrays,' and 'exception handling overview.'

For each of the 20 sections, two computer science graduate students manually extracted the key concepts using a

two-phase annotation scheme. In the first phase, each annotator selected key concepts. The inter-annotator agreement in this phase was 0.7 as measured by Cohen's Kappa [26]. Then, annotators discussed iteratively and extensively until a final agreement was reached. The resulting key concepts form our gold standard for evaluating the proposed methods. We have also asked each annotator to rank the gold standard concepts. This ranking is used for another performance metric that compares the automated ranking to the human ranking.

4.2 Evaluation Metrics

We evaluated the four pretrained models using two approaches. First, we evaluated the key concept extraction using precision, recall, and the F-score at rank k for $k = 5, 10, 15$. This evaluation approach is widely used in key-phrase extraction systems[30]. The other evaluation approach is based on the Normalized Discounted Cumulative Gain at p ($NDCG_p$) [16]. $NDCG_p$ compares the target ranking to the positions that key concepts occupy in the gold standard ranking and penalizes any mismatches. We opt for this evaluation to evaluate the extracted concepts based on $MMR_{adjusted}$ ranking method described earlier.

4.3 Results

First, we report results based on precision, recall, and F-score for top k ranks where $k = 5, 10, 15$ using cosine similarity-based ranking. We also report results for $NDCG_p$ where p= 10 using adjusted MMR metric with $\lambda = 0.5$ and $\lambda = 1$. The $NDCG_p$ ranking results obtained using $\lambda = 1$ for adjusted MMR metric is same as ranking using cosine similarity only. Since the average number of key concepts per section in the gold standard is 11, we chose p = 10 for normalized discounted cumulative gain (NDCG) reporting.

Table 1: Precision, recall and Fscore at 5,10,15 for knowledge component extraction

K	Model	P	R	F
5	BERT	**0.66**	**0.423**	**0.499**
	CodeBERT	0.54	0.342	0.405
	RoBERTa	0.5	0.288	0.354
	SciBERT	0.58	0.41	0.480
10	BERT	**0.578**	**0.659**	**0.597**
	CodeBERT	0.456	0.521	0.473
	RoBERTa	0.522	0.589	0.538
	SciBERT	0.533	0.608	0.55
15	BERT	**0.516**	**0.801**	**0.615**
	CodeBERT	0.367	0.578	0.442
	RoBERTa	0.434	0.673	0.516
	SciBERT	0.483	0.745	0.576

Table 1 shows Precision, recall and F-scores for different pretrained embeddings for $k = 5, 10, 15$. We can see that BERT has the highest precision, recall, and F-score for all the values of k. Similarly, SciBERT yields better performance compared to RoBERTa and CodeBERT. We can observe that even though CodeBERT is trained in NL-PL pairs for different programming languages, it does not provide any advantage over BERT. This might be because the training data for CodeBERT, which is specific to the code in Github, relates to higher-level software engineering concepts than intro-to-programming concepts. Also, the natural language texts used for training CodeBERT is mainly code documentation which might not be directly relevant to basic programming concepts. RoBERTa, even though trained with a different strategy and more data, does not show any better performance than BERT and SciBERT.

Table 2: Normalized Discounted Cumulative Gain($NDCG_{10}$) using $MMR_{adjusted}$ for $\lambda = 1$ and $\lambda = 0.5$.

Model	NDCG@10	
	$\lambda = 1$	$\lambda = 0.5$
BERT	0.83	0.87
CodeBERT	0.75	0.73
SciBERT	0.81	0.80
RoBERTa	0.76	0.77

Also, it is evident from the results that SciBERT, trained in scholarly documents from computer science and biomedical domain, performs on par with BERT. The results show that even though BERT is not trained on any domain-specific data, it performs better than other models trained with more domain-specific or more data.

Table 2 shows the NDCG score using $MMR_{adjusted}$ in equation 2. The results are obtained using $\lambda = 1$ which is equivalent to using cosine similarity only and $\lambda = 0.5$ which gives equal importance to the general and specific components in the adjusted MMR metric. As shown in the results, BERT outperforms other embedding methods similar to the evaluation results shown in Table 1 based on precisioin, recall, and F-measure. The value of NDCG of all the models except for CodeBERT was higher when ranking was done using $\lambda = 0.5$ for $MMR_{adjusted}$ compared to value of λ set to 1.

5. DISCUSSION AND CONCLUSION

Our results indicate that we can use pretrained models for domain model extraction. We also noticed that even though models like CodeBERT and SciBERT are trained on domain-specific data, they did not provide any advantage mainly due to the nature of their training data. Our experiments also evaluated ranking based on Maximal Marginal Relevance to balance general and specific concepts. The main idea behind the evaluation was to check if adjusted MMR as we propose, ranks topic-specific concepts higher. Even though control parameter λ can be trained across different documents to get a more precise value based on cumulative gain we used 0.5 for key concept extraction using a relevance score based on Mean Marginal Relevance to provide equal importance to general and specific concepts. Evaluation based on Table 2 shows that $MMR_{adjusted}$ ranking obtained when $\lambda = 0.5$ is closer to gold-standard ranking or preferred ranking by human annotators.

Acknowledgement

This work has been supported by two NSF awards: Learner Data Institute (NSF award 1934745) and CSEdPad:Investigating and Scaffolding Students' Mental Models during Computer Programming Tasks to Improve Learning, Engagement, and Retention (NSF award 1822816). The opinions, findings, and results are solely the authors' and do not reflect those of NSF.

6. REFERENCES

[1] R. Banjade, P. Oli, L. J. Tamang, J. Chapagain, and V. Rus. Domain model discovery from textbooks for computer programming intelligent tutors. *The International FLAIRS Conference Proceedings*, 34, Apr. 2021.

[2] I. Beltagy, K. Lo, and A. Cohan. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

[3] K. Bennani-Smires, C. Musat, A. Hossmann, M. Baeriswyl, and M. Jaggi. Simple unsupervised keyphrase extraction using sentence embeddings. *arXiv preprint arXiv:1801.04470*, 2018.

[4] A. Bougouin, F. Boudin, and B. Daille. Topicrank: Graph-based topic ranking for keyphrase extraction. In *International joint conference on natural language processing (IJCNLP)*, pages 543–551, 2013.

[5] P. Brusilovsky, E. Schwarz, and G. Weber. Elm-art: An intelligent tutoring system on world wide web. In *International conference on intelligent tutoring systems*, pages 261–269. Springer, 1996.

[6] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336, 1998.

[7] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos. Legal-bert: The muppets straight out of law school. *arXiv preprint arXiv:2010.02559*, 2020.

[8] H. Chau, I. Labutov, K. Thaker, D. He, and P. Brusilovsky. Automatic concept extraction for domain and student modeling in adaptive textbooks. *International Journal of Artificial Intelligence in Education*, 31(4):820–846, 2021.

[9] H. Choi, J. Kim, S. Joe, and Y. Gwon. Evaluation of bert and albert sentence embedding performance on downstream nlp tasks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 5482–5487. IEEE, 2021.

[10] P. J. Deitel and H. M. Deitel. *Java*. Pearson, 2015.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] S. R. El-Beltagy and A. Rafea. Kp-miner: Participation in semeval-2. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 190–193, 2010.

[13] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, et al. Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*, 2020.

[14] I. Goldin, P. I. Pavlik Jr, and S. Ritter. Discovering domain models in learning curve data. *Design Recommendations for Intelligent Tutoring Systems: Volume 4-Domain Modeling*, 4:115–126, 2016.

[15] R. Hosseini and P. Brusilovsky. Javaparser: A fine-grain concept indexing tool for java problems. In *CEUR Workshop Proceedings*, volume 1009, pages 60–63. University of Pittsburgh, 2013.

[16] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[17] K. R. Koedinger, A. T. Corbett, and C. Perfetti. The knowledge-learning-instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive science*, 36(5):757–798, 2012.

[18] A. N. Kumar. Using enhanced concept map for student modeling in programming tutors. In *FLAIRS Conference*, pages 527–532, 2006.

[19] I. Labutov, Y. Huang, P. Brusilovsky, and D. He. Semi-supervised techniques for mining learning outcomes and prerequisites. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 907–915, 2017.

[20] J. H. Lau and T. Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*, 2016.

[21] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[22] K. Lee, L. He, M. Lewis, and L. Zettlemoyer. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, 2017.

[23] X. Liang, S. Wu, M. Li, and Z. Li. Unsupervised keyphrase extraction by jointly modeling local and global context. *arXiv preprint arXiv:2109.07293*, 2021.

[24] Y. D. Liang. *Introduction to Java programming and data structures*. Pearson Education, 2018.

[25] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[26] M. L. McHugh. Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3):276–282, 2012.

[27] R. Mihalcea and P. Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411, 2004.

[28] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space, 2013.

[29] M. Pagliardini, P. Gupta, and M. Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*, 2017.

[30] E. Papagiannopoulou and G. Tsoumakas. A review of keyphrase extraction. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(2):e1339, 2020.

[31] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics:*

Human Language Technologies, Volume 1 (Long Papers), pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[32] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. 2018.

[33] Y. Sun, H. Qiu, Y. Zheng, Z. Wang, and C. Zhang. Sifrank: a new baseline for unsupervised keyphrase extraction based on pre-trained language model. *IEEE Access*, 8:10896–10906, 2020.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[35] X. Wan and J. Xiao. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*, volume 8, pages 855–860, 2008.

[36] S. Wang, C. Liang, Z. Wu, K. Williams, B. Pursel, B. Brautigam, S. Saul, H. Williams, K. Bowen, and C. L. Giles. Concept hierarchy extraction from textbooks. In *Proceedings of the 2015 ACM Symposium on Document Engineering*, pages 147–156, 2015.

[37] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[38] M. Yudelson, R. Hosseini, A. Vihavainen, and P. Brusilovsky. Investigating automated student modeling in a java mooc. *Educational Data Mining 2014*, pages 261–264, 2014.