Using Surrounding Text of Formula towards More Accurate Mathematical Information Retrieval

Cheng Chen[1,2], Yifan Dai[2], Yuqi Shen[2], Jinfang Cai[3], and Liangyu Chen[1,2*]
[1]Engineering Research Center of Software/Hardware Co-design Technology and Application,
[2]Shanghai Key Laboratory of Trustworthy Computing,
[3]Institute of Vocational & Adult Education,
East China Normal University, Shanghai, China

***Abstract*—Formula retrieval is an important research topic in Mathematical Information Retrieval (MIR). Most studies have focused on comparing formulae to determine the similarity between mathematical documents. However, two similar formulae may appear in completely different knowledge domains and have different meanings. Based on N-ary Tree-based Formula Embedding Model (NTFEM), we introduce a new hybrid retrieval model combining formula with its surrounding text for more accurate retrieval. Using keywords extraction technology, we extract keywords from text around the formula which can supplement the semantic information of formula. Then we get the representation vectors of keywords by FastText N-gram embedding model, and the representation vectors of formulae by NTFEM. Finally, documents are first sorted according to the similarity of keywords, and then the ranking results are optimized by formula similarity. Experimental results show that the accuracy of top-10 results is at least 20% higher than that of NTFEM and can be 50% in some specific topics.**

***Index Terms*—Mathematical Information Retrieval, Formula Similarity, Formula Embedding, Word Embedding, Keywords Extraction.**

I. Introduction

Nowadays, the retrieval methods for linear sequence text are widely developed and used, such as $Google$, $Baidu$, $Bing$ and other search engines [1]. However, these methods do not work well for complex formulae that frequently appear in mathematical documents. In this case, formula-based embedding models are proposed for solving the retrieval problem of formulae with two-dimensional structures. The models can capture the structural features of mathematical formulae, but may lack semantic interpretation. Namely, formulae with similar structure may appear in completely different knowledge domains and have different meanings. Therefore, the retrieval results may be unsatisfactory. It is a difficult problem that need to be tracked.

Generally, the text around the formula is typically a very good indication of what domain application the formula is used for. In text-based retrieval methods, documents are represented by a group of keywords, as keywords can summarize the information of text [2]. Then the similarity between two documents is calculated by keyword matching algorithms. More specifically, using keywords instead of long text as the retrieval units, can not only reduce the storage, but also improve the retrieval accuracy. Therefore, the additional semantic information of the formula can be supplemented through the keywords of the text around the formula.

In this paper, we present a hybrid retrieval model combining the formula with its surrounding text for more accurate Mathematical Information Retrieval (MIR). We first use Rapid Automatic Keyword Extraction (RAKE) algorithm [3] to extract the keywords from text around the formula, and the representation vectors of the keywords are then obtained by FastText N-gram embedding model [4]. Meanwhile, we get the representation vectors of formulae by NTFEM [5]. Finally, the mathematical documents are first sorted based on the similarity of the keyword vectors, then the retrieval results are reordered according to the formulae similarity. Experimental results on the dataset provided by TopicEq [2] show our model has achieved more accurate retrieval results than other retrieval models and can effectively capture semantic features of formulae.

The remainder of this paper is organized as follows. We first introduce the preliminaries and challenges of formula retrieval in Section II. In Section III, we then present the hybrid retrieval model combining the formula with its surrounding text. In Section IV, we evaluate our model on the dataset provided by TopicEq and compare with other retrieval models. Section V reviews related work on formula retrieval methods and keyword extraction methods. Finally, we conclude this paper in Section VI.

II. Formula Retrieval

A. Definition of Formula Retrieval

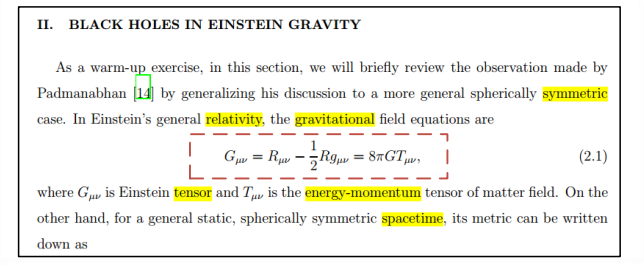
Most present information retrieval systems usually do not consider mathematical notations and formulae in documents because they cannot build effective indexes for them. Search engines like *Google* mostly treat user inputs of symbols, equations and formulae as normal text without understanding their mathematical semantics [6]. They may be able to find similar text, but are very often fail to find the exact match. Given a query formula, the system should be able to parse its structure and semantics and then find the matching documents with similar formula. For instance, given an incomplete equation $e^{n\pi} + 1$, the formula retrieval system

*Corresponding author: Liangyu Chen (email:lychen@sei.ecnu.edu.cn).
DOI reference number: 10.18293/SEKE2021-143.

should match to the Euler's Identity formula: $e^{i\pi} + 1 = 0$. The key issue is how to measure the similarity between two formulae which is different from the text similarity. We need extract features of math formulae so that we can distinguish from different formulae, then get similar formulae.

B. Challenges of Formula Retrieval

Formulae are generally displayed in two-dimensions. However, the current representation ways, such as LaTeX and MathML, cannot reflect the structural characteristics of formulae. Besides, formulae are highly abstract. Two similar formulae may appear in completely different knowledge domains and have different meanings, and this lead to unexpected match [2]. Therefore, both structural and semantic features of formulae should be considered in retrieval process.



II. BLACK HOLES IN EINSTEIN GRAVITY

As a warm-up exercise, in this section, we will briefly review the observation made by Padmanabhan [14] by generalizing his discussion to a more general spherically symmetric case. In Einstein's general relativity, the gravitational field equations are

$$G_{\mu\nu} = R_{\mu\nu} - \frac{1}{2}Rg_{\mu\nu} = 8\pi G T_{\mu\nu}, \quad (2.1)$$

where $G_{\mu\nu}$ is Einstein tensor and $T_{\mu\nu}$ is the energy-momentum tensor of matter field. On the other hand, for a general static, spherically symmetric spacetime, its metric can be written down as

Fig. 1. Formulae and text snippets from Physics.

An interesting observation is, with text around the formula, the meaning of the formula is determined more clearly. As shown in Fig. 1, the highlighted words are keywords extracted by keywords extraction algorithms, and the word *relativity* and term *gravitational* clearly show the formula is intended for Physics. This example shows that keywords can greatly help formula retrieval, and the combination of formula retrieval and context analysis can better capture the semantics of documents and lead to more accurate match.

III. Method

The process of our hybrid retrieval model is shown in Fig. 2. And the detail of process is explained as follows:

1) Processing of mathematical information: Formulae are parsed into n-ary trees, and keywords are extracted from text around the formula.

2) The representation vectors of formulae and keywords: The representation vectors of keywords are obtained by an n-gram word embedding model, while the representation vectors of the formulae by NTFEM.

3) Processing of queries: In the same way, the representation vectors of keywords and formulae from input queries are calculated respectively by above models.

4) Optimization for retrieval results: The mathematical documents in the database are sorted based on the similarity of the keywords vectors first, after that, the retrieval results are sorted again on the basis of formulae similarity.

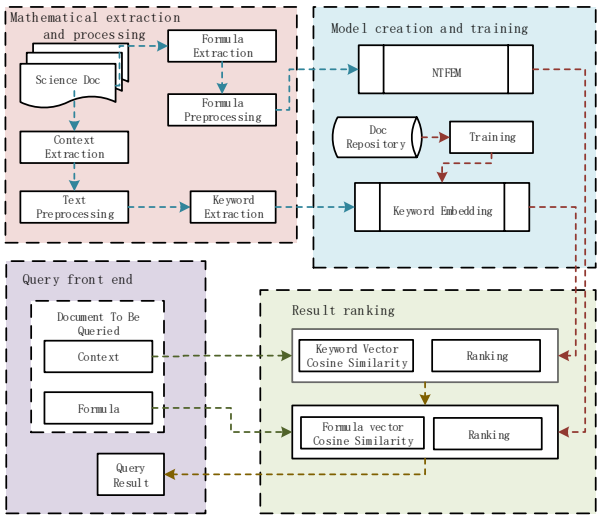


Fig. 2. Our hybrid retrieval model

A. Formula Embedding Model

NTFEM first transforms the mathematical formula from a two-dimensional structure to a one-dimensional linear sequence. The transformation steps are listed as follows:

- Convert a formula (MathML) into an n-ary tree.
- Generate the tuple sequences and tokenize the tuples.

Fig. 3 shows the process of the formula $a+b\times c+2\times b\times c$ being converted from MathML notation to tree structures, in which (b) is the binary tree, and (c) is the n-ary tree.

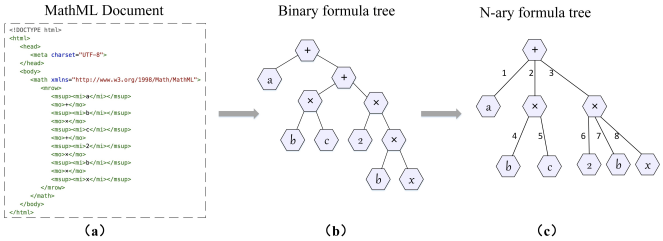


Fig. 3. Example of formula conversion

Then NTFEM sets labels for symbols in the formulae, where these labels fall into the following categories:

- Numbers "N".
- Identifiers such as variable symbols "V".
- Commutative operators "U".
- Non-commutative operators "O".

As shown in the Table I, NTFEM uses a pair-based method [7] to define "words" and generate "sentences" by breadth-first traversal which is the input of the embedding model.

In order to better represent the features of formulae, NTFEM uses the weighting strategy based on FastText N-gram embedding model [4].

TABLE I
TUPLES GENERATED FROM THE FORMULA REPRESENTED IN FIG. 3(C).

ID	symbol pairs	tree tuples
1	$(+ , a)$	(U:ADD,V:a)
2	$(+ , \times)$	(U:ADD,U:times)
3	$(+ , \times)$	(U:ADD,U:times)
4	$(\times , b)$	(U:times,V:b)
5	$(\times , c)$	(U:times,V:c)
6	$(\times , 2)$	(U:times,N:2)
7	$(\times , b)$	(U:times,V:b)
8	$(\times , x)$	(U:times,V:x)

1) Level Weight: Taking depth and complexity into consideration, the weight of tuple t can be expressed in the formula sequence f with

$$W_{f[t]} = \frac{(depth - D(t))}{depth} \times \alpha + \frac{L(D(t))}{\sum_i^{depth} L(i)} \times \beta, \quad (1)$$

where $depth$ is the depth of the n-ary tree of f, $D(t)$ is the level of tuple t in the n-ary formula tree, and $L(i)$ is the number of tuples at the i-th level, and α, β are two tuning parameters. Here we set $\alpha + \beta = 1$, and adjust these two parameters to get better results.

2) Frequency Weight: With consideration of tuple frequency in the corpus, NTFEM combines SIF [8] with level weight to get the final formula embedding v_f:

$$v_f = \frac{1}{|f|} \sum_{t \in f} \frac{\gamma}{\gamma + p_t} v_t w_t, \quad (2)$$

where γ is hyperparameters, p_t is the frequency of tuple t in corpus, and v_t is the formula tuple embedding.

B. Extract and Train the Keywords

In this paper, we use RAKE algorithm [3] to extract the keywords from text around the formula. Contrary to other methods that rely on Natural Language Processing (NLP) technologies, RAKE can automatically extract keywords from text with only one traversal. Moreover, it can extract key phrases from mathematical text, especially longer technical terms, which is in line with the task scenario of this paper. The algorithm first uses punctuation marks to break a document into clauses, and then, for each clause, uses stop words as delimiters to divide the clauses into phrases that serve as candidates for the final extracted keywords. Next, each phrase can be divided into several words by space. And each phrase can be scored by the sum of its word scores. The score of word w is calculated as follows:

$$WordScore(w) = \frac{Degree(w)}{Frequency(w)}, \quad (3)$$

where $Degree(w)$ is the degree of the word (a concept in the network) and $Frequency(w)$ is the frequency of the word. Finally, the top third of candidate phrases are identified as keywords, after the phrases are sorted by their scores in descending order.

Like processing in NTFEM, FastText is also used for keywords training in this paper. In order to better capture the semantic features of keywords, we make the following improvements to the training process:

1) Stop words adjustment: Stop words such as by, $allows$, $almost$ and $everywhere$ appear frequently in documents and have little effect on reflecting useful information. In most models, these words would be removed in order to improve training efficiency. But for mathematical text corpus, stop words may appear in the definition of mathematical formula, the description of the mathematical theory, and other text which is important for reflecting mathematical semantics. Therefore, we have adjusted the stop words and preserved some of them that may affect the mathematical semantics.

2) Negative Sampling: In the CBOW model, a word w is predicted by its context. Namely, for a given $Context(w)$, the word w is a positive sample and the others are negative samples. Generally, 5 negative sample words will be selected for each $Context(w)$. The probability that the sample w_i is selected is:

$$P(w_i) = \frac{f(w_i)}{\sum_{j=0}^{n} (f(w_j))}. \quad (4)$$

Then positive and negative samples are represented with 1 and 0 respectively. In this case, the results of the output layer can be normalized between $[0, 1]$. Compared with a group of negative samples obtained by random sampling, the objective function of the model is listed as follows:

$$F = \sum_{n=1}^{N} \left[\log\left(1 + e^{-\gamma(s_n, c_j)}\right) + \sum_{m \in M_{c_j}} \log\left(1 + e^{\gamma(m, C_j)}\right) \right], \quad (5)$$

where M_c represents a group of negative samples obtained by negative sampling. And $\gamma(S, C)$ is the evaluation function related to the word S and its context C, it can be calculated as follows:

$$\gamma(S, C) = \frac{1}{|C|} \sum_{s' \in C} \mathbf{u}_{s'} \mathbf{v}_s. \quad (6)$$

Here $\mathbf{v}_s$ represents the word vector of the word S, and $u_{s'}$ is the word vector of the word s' in the $Context(S)$.

C. Similarity of Mathematical Documents

After obtaining the representation vectors of the formulae and the surrounding text, we use *cosine* similarity of vectors as the basic ranking. The documents is first sorted according to the similarity between the keywords in the database and the user input. For the query document q and a document p in the database, with their text vector V_q and V_p respectively, the similarity is measured as follows:

$$\text{Sim(p,q)} = \frac{\sum_{i=1}^{n} (V_{q_i} \times V_{p_i})}{\sqrt{\sum_{i=1}^{n} (V_{q_i})^2} \times \sqrt{\sum_{i=1}^{n} (V_{p_i})^2}}. \quad (7)$$

Then, top-k results are reordered according to the *cosine* similarity of the formulae. The final results are the similar candidate documents of the user input.

IV. Experiments and Results

A. Dataset

For evaluation of our hybrid model, we use the dataset provided by TopicEq [2], which includes nearly $100,000$ scientific and technological articles published in arXiv.org in the past five years and generate $400,000$ pieces of data. Each data consists of a formula in LaTeX notation and ten sentences around the formula, where five before the formula and the rest after it. The dataset covers 10 topics and labels them with $T_1, T_2, \ldots, T_{10}$, including astrophysics (T_1), relativity (T_2), graph theory (T_3), linear algebra (T_4), machine learning (T_5), quantum physics (T_6), particle physics (T_7), number theory (T_8), optimization (T_9) and probability (T_{10}). We randomly select $93,051$ pieces of data from 5 categories and used 90% of the data as the training set and the rest as the test set. The distribution of the data is listed as follows:

TABLE II
The distribution of data

Topic	Training Set	Test Set	Total
astrophysics	18,401	1,816	20,217
relativity	18,826	1,823	20,649
graph theory	17,469	1,678	19,147
linear algebra	16,522	1,652	18,174
machine learning	13,048	1,240	14,828

Note that we have made a tool to convert the formula with notation from LaTeX to MathML. In this case, we can obtain the representation vectors of formulae through NTFEM.

B. Retrieval Results and Evaluation Standards

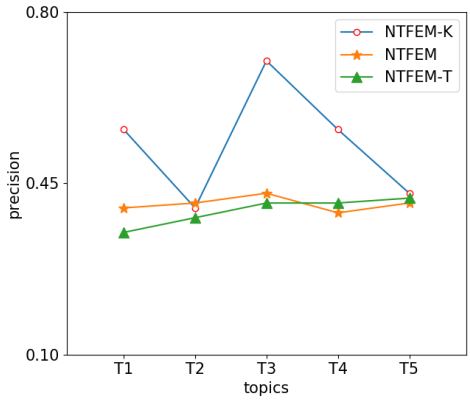
In this paper, we used $P@k$ to calculate the accuracy of retrieval results. For a query, $P@k$ represents the proportion of results related to user input among the top-k retrieval results, and the calculation formula is:

$$\text{P@}k = \frac{\text{true positives@}k}{\text{true positives@}k + \text{false positives@}k}. \quad (8)$$

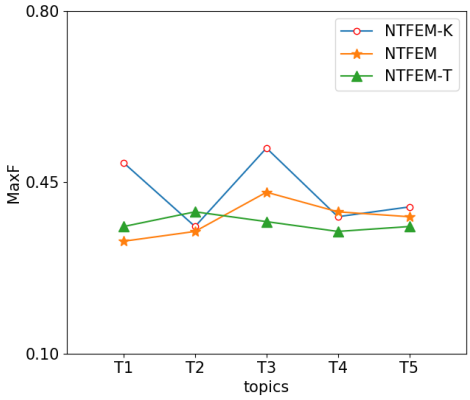
Experimental results shows that the retrieval accuracies after enhanced through the incorporation of surrounding text are greatly improved, compared with pure formula retrieval methods. As shown in Table III, for the topic of *astrophysics*, the accuracy of top-10 retrieval results of NTFEM-K is improved by 50% compared with NTFEM. Furthermore, retrieval accuray in the topic of *machine learning* is much lower than others of NTFEM retrieval model.This may be because the field of machine learning intersects with other fields, and NTFEM has difficulty in distinguishing formulae with similar structures, which can be solved through the incorporation of the surrounding text.

TABLE IV
Retrieval results of NTFEM-T

Topic	P@10	P@50	P@100	P@200	P@500
astrophysics	**0.400**	**0.320**	**0.300**	0.315	**0.324**
relativity	0.300	**0.300**	**0.260**	**0.285**	**0.286**
graph theory	0.300	**0.300**	**0.310**	**0.260**	**0.262**
linear algebra	**0.300**	0.220	**0.250**	**0.180**	**0.188**
machine learning	**0.700**	**0.500**	**0.430**	**0.310**	**0.250**



(a) Precission

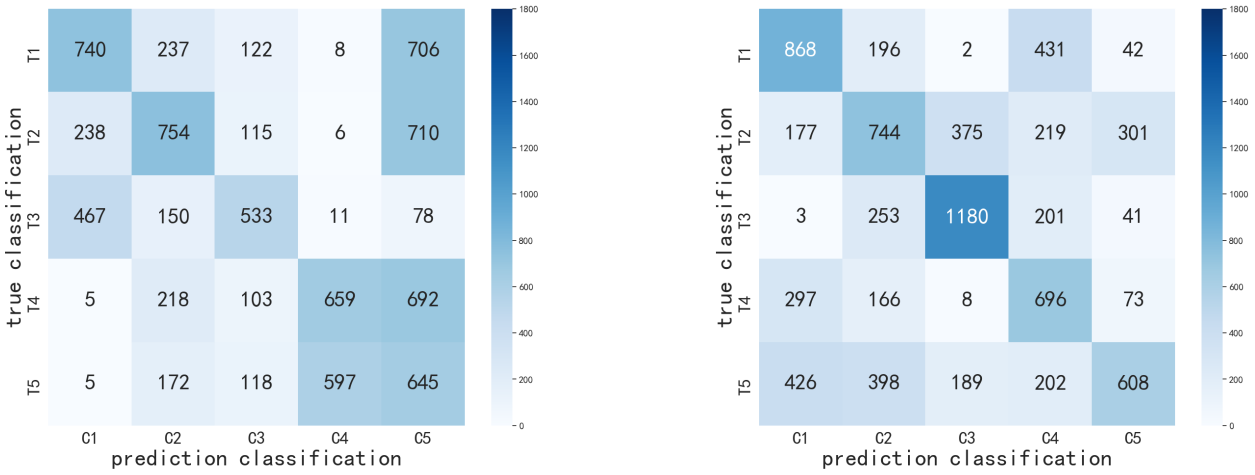


(b) MaxF

Fig. 4. Precison and MaxF indicators of the above three models

TABLE III
Retrieval results of NTFEM and NTFEM-K

Topic	P@10		P@50		P@100		P@200		P@500	
	NTFEM	NTFEM-K	NTFEM	NTFEM-K	NTFEM	NTFEM-K	NTFEM	NTFEM-K	NTFEM	NTFEM-K
astrophysics	0.300	**0.800**	0.260	**0.780**	0.280	**0.770**	0.325	**0.630**	0.310	**0.454**
relativity	0.300	**0.500**	0.280	**0.400**	0.230	**0.430**	0.255	**0.415**	0.278	**0.354**
graph theory	0.300	**0.500**	0.240	**0.480**	0.250	**0.400**	0.225	**0.345**	0.228	**0.316**
linear algebra	0.200	**0.900**	0.220	**0.740**	0.170	**0.680**	0.165	**0.675**	0.158	**0.598**
machine learning	0.100	**0.900**	0.180	**0.560**	0.150	**0.510**	0.215	**0.420**	0.202	**0.294**



(a) Classification results of NTFEM

(b) Classification results of NTFEM-K

Fig. 5. Confusion matrix heat map

In this paper, we also study the effect of embedding all surrounding text without extracting keywords, which is called NTFEM-T model. We find that the overall performance of NTFEM-T is also better than NTFEM on retrieval accuracy. As shown in Table IV, the retrieval accuracies of NTFEM-K and NTFEM-T are much higher than that of NTFEM for the $machine\ learning$ topic. This means that the text around the formula can supplement the semantics of the formula. In addition, for retrieval tasks on topics $relativity$ and $graph\ theory$, the retrieval accuracies of NTFEM-T are about 20% lower than those of NTFEM-K, which shows that the keywords extraction method can efficiently capture the semantic features of the formula.

We use two indicators Precision and MaxF to compare the performance of the above models. As shown in Fig. 4, the precision of NTFEM-K is much higher than other models, especially in retrieval tasks of T_1, T_3 and T_4 topics. MaxF is the harmonic value of precision and recall, which can reflect the overall performance of the model. It is easily observed that the NTFEM-K model has best performance.

Finally, we evaluate the above models on the task of document clustering. A heat map of confusion matrix (Fig. 5) is used to visualize the clustering results. The diagonal elements of the heat map indicate the number of documents which are correctly labelled. According to the experiment results, NTFEM-K has also achieved better performance on the classification task compared to the NTFEM model.

V. Related Work

The research on Math Information Retrieval has been a hot topic in the fields of information retrieval and knowledge engineering. In the early stage, formula-based methods used various transformation algorithms to convert the formulae into special notations, which are suitable for indexing and retrieved by matching algorithms. Later, the text around the formula were incorporated to supplement the information of the formula.

A. Formula-based Retrieval

For pure formula retrieval, most previous studies can be roughly categorized into *text-based* and *tree-based* models [9], [10]. In *text-based* methods, formulae are converted to ordered strings, which can be considered as inputs with the same processing in the traditional text retrieval models. Math Indexer and Searcher (MIAS) [11] implemented formulae matching in three steps: ordering commutative operations, variables unification and constants unification. DLMF project [12] implemented the textualization of math formulae through flattening and normalization process, in which each formula generates a unique form for all possible orders of operator symbols. Kumar et al. [13] applied the largest common sub-string algorithm to calculate the similarity between documents. Unfortunately, most text-based methods are inefficient, and the retrieval accuracy is low due to the inability to extract the two-dimensional structural features of the formula.

In *tree-based* methods, the formulae are transformed into trees and the partial matching methods are usually used for retrieval tasks. For example, MathWebSearch (MWS) [14], a model based on the substitution tree index, used term indexing to minimize access time and storage. Moreover, different representation trees of formulae also affect retrieval performance. Zanibbi et al. compared the performance based on two hierarchical representations, Symbol Layout Trees (SLTs) and Operator Trees (OPTs), and designed a series of mathematical retrieval systems [15].

In recent years, some embedded models have been applied to the field of mathematical information retrieval. Thanda et al. [16] first used the *Doc2Vec* model on mathematical formulae by representing the formula tree as a 100-dimensional

vector. Gao et al. [1] proposed *symbol2vec* and *formula2vec*, where *symbol2vec* simply learns the symbols of formulae in LaTeX based on CBOW by using negative sampling, and *formula2vec* treats the formulae as sentences and uses the *PV-DM* to learn the characteristics of formulae.

B. Combination of Formula and Its Surrounding Text

Compared to the pure formula retrieval, we believe that the retrieval model combining formula with text can learn more useful features by extracting the mathematical semantics of the text around the formula. Krstovski and Blei [6] proposed a word embedding model for mathematical expressions. They treated the entire equation as a word through the distributed representation of the formula, and embeded it in conjunction with the surrounding text. However, they ignored the internal structure and symbolic semantics of formula. Yasunaga and Laferty [2] designed a topic model for scientific documents containing formulae. In this model, they used long and short-term memory models (LSTM) [17] to learn the characteristics of the formula sequence. Through a series of verifications, the effect of the topic model that combines formula and the surrounding text is better than that of a simple text topic model. Most previous hybrid models handle formulae in a rough way, so that they cannot capture the characteristics of formulae well. Moreover, the research on the combination of formula and the surrounding text is still in its early stage and cannot be effectively applied to specific retrieval tasks.

C. Keywords Extraction

Keywords use a set of words to define the content of the text, which can reduce storage space and streamline the calculation of document similarity. Generally speaking, there are two ways to generate keywords:

- Keyword allocation technique selects multiple words in a given thesaurus as keywords to label a document.
- Keyword extraction technique extracts some words from a document as keywords to label it.

At present, the more prevalent method is keyword extraction technique, which can be mainly divided into two categories:

1) Supervised learning algorithm: A supervised learning algorithm transforms the process of keyword selection into a binary classification problem. The specific method is to set a boolean label on the extracted candidate words to indicate whether the candidate word is a keyword, and then train the keyword classifier with a certain amount of datasets with labels. For a document, all candidate words can be extracted first, and then the trained keyword classifier is used to label and classify, and finally all the keywords are obtained.

2) Unsupervised learning algorithm: An supervised learning algorithm first extracts candidate words from the document, and scores the candidate words according to certain rules. Different scoring strategies focus on the features of different dimensions of a text, and then output top-k candidate words with the highest scores as the final keywords. The common algorithms are TF-IDF, RAKE, LDA, and etc.

VI. Conclusion

In this paper, we present a hybrid model NTFEM-K in corporation of surrounding text for mathematical information retrieval. We use NTFEM to produce formula embeddings and obtain the representation vectors of keywords by FastText that captures mathematical semantics. Then we use quadratic sorting method to obtain the retrieval results. Experiments show that our model achieves higher retrieval accuracy than previous models of formula retrieval.

Acknowledgement

This work was partially supported by OneSmart Education Group and the Open Research Fund of Engineering Research Center of Software/Hardware Codesign Technology and Application, Ministry of Education (East China Normal University).

References

[1] L. Gao, Z. Jiang, Y. Yin, K. Yuan, Z. Yan, and Z. Tang, "Preliminary exploration of formula embedding for mathematical information retrieval: can mathematical formulae be embedded like a natural language?" *arXiv preprint arXiv:1707.05154*, 2017.

[2] M. Yasunaga and J. D. Lafferty, "Topiceq: A joint topic and mathematical equation model for scientific texts," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 7394–7401.

[3] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Automatic keyword extraction from individual documents," *Text mining: applications and theory*, vol. 1, pp. 1–20, 2010.

[4] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[5] Y. Dai, L. Chen, and Z. Zhang, "An n-ary tree-based model for similarity evaluation on mathematical formulae," in *SMC*. IEEE, 2020, pp. 2578–2584.

[6] K. Krstovski and D. M. Blei, "Equation embeddings," *arXiv preprint arXiv:1803.09123*, 2018.

[7] K. Davila, R. Zanibbi, A. Kane, and F. W. Tompa, "Tangent-3 at the ntcir-12 mathir task." in *NTCIR*, 2016.

[8] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," in *5th International Conference on Learning Representations*, 2017.

[9] D. Stalnaker and R. Zanibbi, "Math expression retrieval using an inverted index over symbol pairs," in *Document Recognition and Retrieval XXII*, 2015, p. 940207.

[10] R. Miner and R. Munavalli, "An approach to mathematical search through query formulation and data normalization," in *Towards Mechanized Mathematical Assistants*. Springer, 2007, pp. 342–355.

[11] M. Ružicka, P. Sojka, and M. Líška, "Math indexer and searcher under the hood: History and development of a winning strategy," in *Proceedings of the 11th NTCIR Conference on Evaluation of Information Access Technologies*, 2014, pp. 127–134.

[12] B. R. Miller and A. Youssef, "Technical aspects of the digital library of mathematical functions," *Annals of Mathematics and Artificial Intelligence*, vol. 38, no. 1, pp. 121–136, 2003.

[13] P. P. Kumar, A. Agarwal, and C. Bhagvati, "A structure based approach for mathematical expression retrieval," in *International Workshop on Multi-disciplinary Trends in Artificial Intelligence*, 2012, pp. 23–34.

[14] M. Kohlhase, S. Anca, C. Jucovschi, A. G. Palomo, and I. A. Sucan, "Mathwebsearch 0.4, a semantic search engine for mathematics," *Manuscript at http://mathweb. org/projects/mws/pubs/mkm08. pdf*, 2008.

[15] R. Zanibbi, K. Davila, A. Kane, and F. W. Tompa, "Multi-stage math formula search: Using appearance-based similarity metrics at scale," in *SIGIR*, 2016, pp. 145–154.

[16] A. Thanda, A. Agarwal, K. Singla, A. Prakash, and A. Gupta, "A document retrieval system for math queries." in *NTCIR*, 2016.

[17] Z. Huang, X. Wei, and Y. Kai, "Bidirectional lstm-crf models for sequence tagging," *Computer Science*, 2015.