A Formula Embedding Approach to Math Information Retrieval

Amarnath Pathak[1], Partha Pakray[1] and Alexander Gelbukh[2]

Dept. of Computer Science & Engineering
National Institute of Technology, Mizoram[1]
Aizawl, India
Instituto Politécnico Nacional[2]
Mexico City, Mexico
{amar4gate,parthapakray}@gmail.com, gelbukh@gelbukh.com

Abstract. Intricate math formulae, which majorly constitute the content of scientific documents, add to the complexity of scientific document retrieval. Although modifications in conventional indexing and search mechanisms have eased the complexity and exhibited notable performance, the formula embedding approach to scientific document retrieval sounds equally appealing and promising. Formula Embedding Module of our proposed system uses a Bit Position Information Table to transform math formulae, contained inside scientific documents, into binary formulae vectors. Each set bit of a formula vector designates presence of a specific mathematical entity. Mathematical user query is transformed into query vector, in similar fashion, and the corresponding relevant documents are retrieved. Relevance of a search result is characterized by extent of similarity between indexed formula vector and the query vector. Promising performance, under moderately constrained situation, substantiates competence of our proposed approach.

Keywords: Math Information Retrieval, Formula Embedding, Math Formula Search, Scientific Document retrieval, Precision

1 Introduction

Information Retrieval, an application area of Natural Language Processing, intends to content information needs of end users. Information needs vary from individual to individual which eventually poses challenge to design of search engines. Though the conventional search engines come equipped with text and multimedia content searching abilities, they often lack interface and requisite abilities to search math formulae present in scientific documents.

Effective retrieval of scientific documents is challenged by abundance and complexity of math formulae contained therein. Presence of these formulae significantly demarcates scientific documents from normal text documents and mandates modification in conventional retrieval mechanisms to ease their retrieval. Intelligence of scientific documents search engine (henceforth called as math search engine) is manifested in its ability to comprehend crux of math formula query prior to searching. Relevant search results should be substantially influenced and guided by semantics of query formula rather than its syntactic exactness. For example, given query formula to be $\frac{1}{1+e^x}$, usually, the end user will be equally content with search results containing $\frac{1}{1+e^p}$, $\frac{1}{1+a^x}$, $\frac{1}{1+a^p}$, $1+e^x$, $\frac{1}{1+e^{-x}}$ e^x, a^p and the list goes on. Thus, the preciseness of numbers, variables and syntactic constructs of query formula are often compromised for the sake of semantic equivalence or sub-formulae results. In a nutshell, in context of math formula search, relevance of search results encompasses a wide range of meanings which contrasts with the usual text query search wherein the relevance designates exactness and optimal match.

Distinctive stipulations, laid out by math formula search, necessitate revamping conventional techniques to successfully meet the expectations of end users of math search engine. To this end, document indexing and formulae search techniques have been subjected to numerous modifications which exhibit promising performance under normal as well as eccentric situations. In particular, indexing of canonicalized, tokenized and structurally unified variants of raw formulae have been found to engender relevant search results [15, 12, 13]. Canonicalization tackles representational non-uniformity of math formulae whereby equivalent formulae, with minor representational difference, are homogenized. In addition, tokenization and structural unification help retrieve sub-formulae and similar formulae, respectively. Weights assigned to tokenized and structurally unified formulae characterize extent of their similarity with original formulae which eventually help in ranking retrieved documents. Although the approach splendidly caters to the need of math information seekers, it can be fascinating to prospect other competent techniques which may depict similar traits with relatively lesser effort. Our proposed approach gets driven by this little motivation.

In this paper, we describe our formulae embedding approach to math information retrieval and a comprehensive analysis of obtained system results to infer strengths and weaknesses of proposed approach. Formulae embedding envisions math formulae of documents and the mathematical user query as binary vectors wherein each bit position designates absence or presence of a specific mathematical entity, such as single character variable, superscript, subscript, fraction, special symbol, and so on. After having preprocessed the scientific documents and the math formulae contained therein, each formula is transformed to a fairly large-sized vector which encompasses nearly all the information content of corresponding formula. Eventually, the multitude of formula vector to corresponding document mapping are indexed to assist the searcher module in its hunt for relevant documents. Count of matching set bits (set bits at same positions) of indexed formula vector and query vector constitutes the crucial criterion for assessing relevance of retrieved documents. Although there exist some pitfalls associated with proposed approach, it predominantly lives up to expectations of end users as a baseline approach.

Efficacy of proposed approach has been tested using a corpus comprising of 212 documents of NII Testbeds and Community for Information Access Research (NTCIR)-12 MathIR task[1] and a queryset comprising of 19 Wikipedia Main Task queries and 4 Wikipedia Browsing Task queries. Well known trec_-eval[2] tool has been employed to evaluate system's effectiveness on the grounds of Precision at 5 (*P_5*), Precision at 10 (*P_10*), mean average precision (*map*) and binary preference-based measure (*bpref*) measures. The tool compares system results against a Gold Dataset embodying judged entries for each query of the queryset. Considerably high values of evaluation measures serve as testament to the distinguished abilities of the proposed approach. Furthermore, the system results have been also compared with the results furnished by a conventional text search engine, using the same experimental framework. Vast gaps in the corresponding evaluation measures of the two systems are indicative of the conceptual difference between conventional text search and the math formulae search.

Rest of the paper is organized as follows: Section 2 describes some closely related works; Section 3 details system architecture and its working description; Section 4 discusses experimental framework using which system's effectiveness has been tested; Section 5 details system results and their comprehensive analysis; Section 6 points direction for future research; Section 7 concludes the paper.

2 Related Works

Information rich contents of scientific documents serve as crucial input to many scientific and technical research. However, scientific documents being primarily rich in math formulae, conventional text oriented indexing and search techniques often fail to retrieve information from such documents.

Although a number of past works have addressed the issue of information retrieval from scientific documents, the formulae embedding approach, in particular, is still in its infancy. A relevant work in this regard is the document retrieval system [16] of NTCIR-12 MathIR task which uses Document To Vector (Doc2Vec) and Latent Dirichlet Allocation (LDA) for retrieving similar formulae and sub-formulae. In particular, the Distributed Bag of Words (PV-DBOW) model, a special variant of Doc2Vec algorithm, has been exploited and extended to represent math expressions in terms of real valued vectors. Furthermore, the preliminary exploration of formulae embedding has shown promising results in context of math information retrieval [4]. Initially a symbol2vec method transforms formulae symbols into vector representation and the cosine distance of symbol vectors of closely related symbols turn out to be minimum. Symbol vectors and Distributed Memory Model of Paragraph Vectors (PV-DM) are then used to embed formulae. Well known cosine similarity measure computes similarity of formulae vectors and assigns suitable score. However, the insufficient system description and analysis of results barely convey strengths and weaknesses of the approach.

The necessity of math formulae search led to the introduction of a new math pilot task in NTCIR-10 conference [1, 6] and the task has continued to be part of

[1] http://ntcir-math.nii.ac.jp/
[2] http://trec.nist.gov/trec_eval/

subsequent NTCIR conferences, namely NTCIR-11 [2, 5] and NTCIR-12 [17, 7]. A Math Indexer and Searcher System (MIaS) of NTCIR-10 conference employs preprocessing of scientific documents followed by canonicalization and linearization of mathematical expressions to ease their retrieval [8]. An enhanced version of MIaS [12] uses better preprocessing, canonicalization, math representation and query expansion strategies which eventually helped it emerge as the winner system of NTCIR-11. Furthermore, combining text keywords with math query using different querying strategies has furthered the effectiveness of retrieval [9]. MIaS at NTCIR-12 conference was characterized by an additional structural unification component which helped retrieve semantically similar formulae [13]. Tangent-3 math retrieval system [3], at NTCIR-12, uses inverted indexing to store mathematical entities extracted from Symbol Layout Tree (SLT). This is followed by a 2 stage retrieval process. While the first stage primarily concerns retrieval of relevant expressions using iterator trees and trivial ranking, the second stage concerns strict re-ranking of top-k best candidates.

Modifications in conventional indexing techniques have also contributed notably to the domain of math formulae search. In particular, substitution tree based indexing techniques [14, 11] index math formulae along nodes and leaves of a substitution tree and minimize the memory requirement. Nodes of the tree correspond to substitutions whereas the leaves correspond to mathematical entities. A depth first traversal of the tree yields an indexed formula.

An architecture for scientific document retrieval, containing 3 different Text-Text, Text-Math and Math-Math entailment modules, has been proposed in [10]. Architecture supports search for user query embodying text as well as mathematical contents. Text Entailment (TE) module matches text part of the query to the indexed text contents, Math Entailment (ME) module matches mathematical part of query to the indexed math formulae and Text Math Entailment (TME) module matches text part of the query to the standard names of indexed math formulae.

3 System Description

This section provides detailed description of corpus used in our experimentation and the crucial components of system architecture which sequentially function to output top ranked indexed documents, corresponding to each user query. Figure 1 shows the system architecture and workflow of proposed system.

3.1 Corpus Description

Proposed system has been experimented with a corpus comprising of 212 documents chosen from vast arXiv and Wikipedia corpora of NTCIR-12 MathIR task. Total size of the corpus is 22.6 MB with majority of documents being formulae rich and considerably large in size. The documents are in XHTML format with formulae written using MathML.

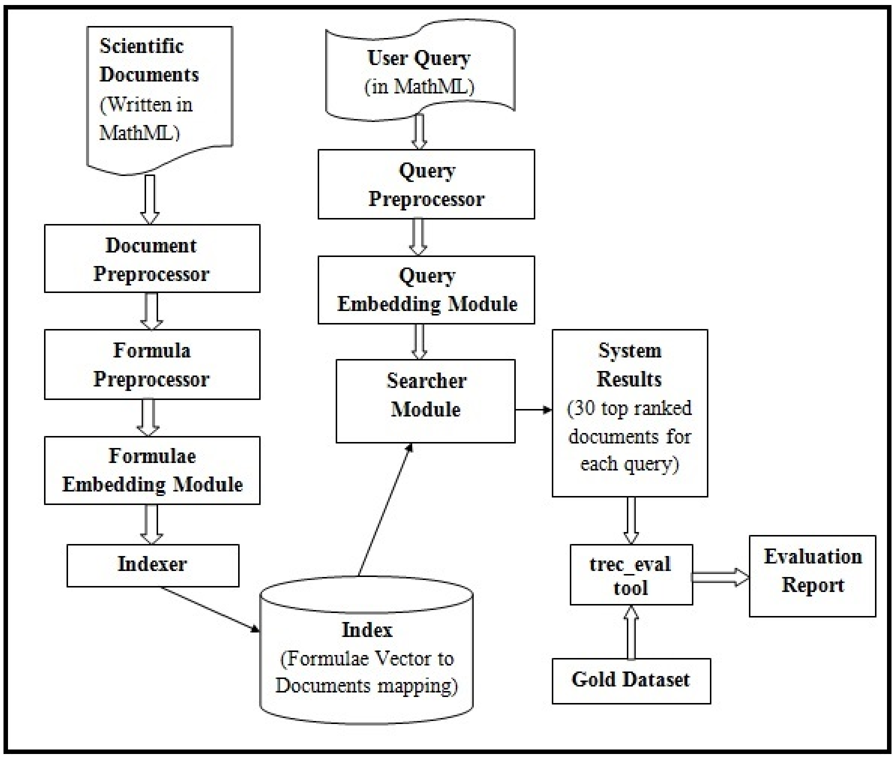


Fig. 1. System Architecture and Working Description

A) **arXiv corpus**
arXiv corpus of NTCIR-12 MathIR task comprises of 105,120 scientific documents chosen from the following arXiv categories: math, cs, physics:math-ph, stat, physics:hep-th and physics:nlin [17]. Documents contain roughly 60 million math formulae written using MathML. arXiv corpus is intended for technical users.

B) **Wikipedia corpus**
Wikipedia corpus of NTCIR-12 MathIR task comprises of 319,689 articles in XHTML format which contain 590,000 formulae encoded using LaTeX, Presentation MathML and Content MathML. Unlike arXiv corpus, Wikipedia corpus is intended for non-technical users.

While selecting the documents for experimentation, adequate care has been taken to ensure that:

1. a reasonable proportion of documents are common to different queries of the query set.
2. some documents contain exact formula query whereas some contain similar formulae, sub-formulae and parent formulae.

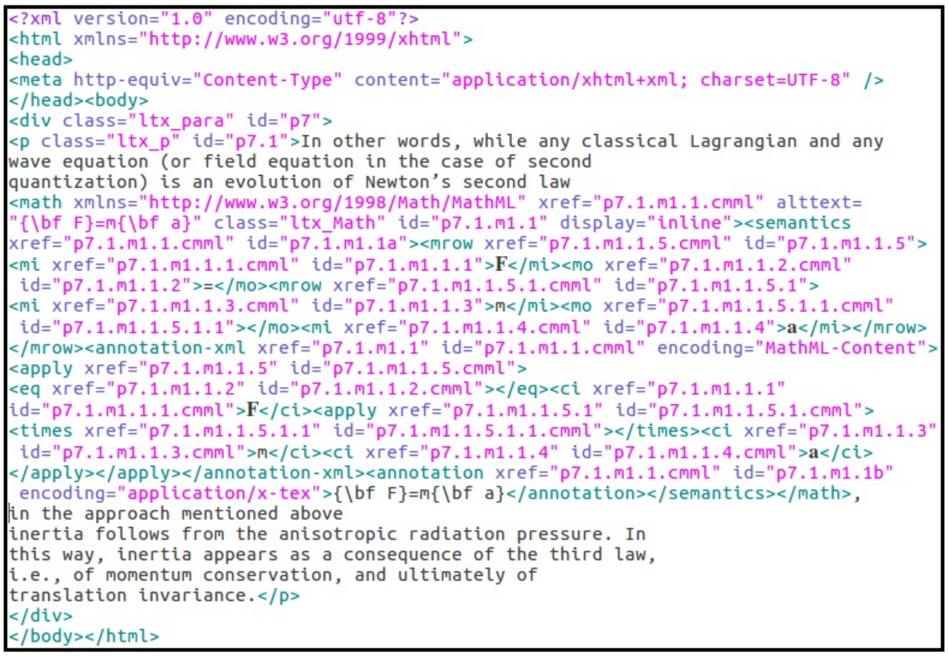
Table 1 summarizes our corpus information.

Table 1. Corpus Description

Size of Corpus	Number of Documents	Source of Documents	Format of Documents	Encoding of Documents
22.6 MB	212	arXiv & Wikipedia corpora of NTCIR-12 MathIR task	XHTML	Presentation & Content MathML

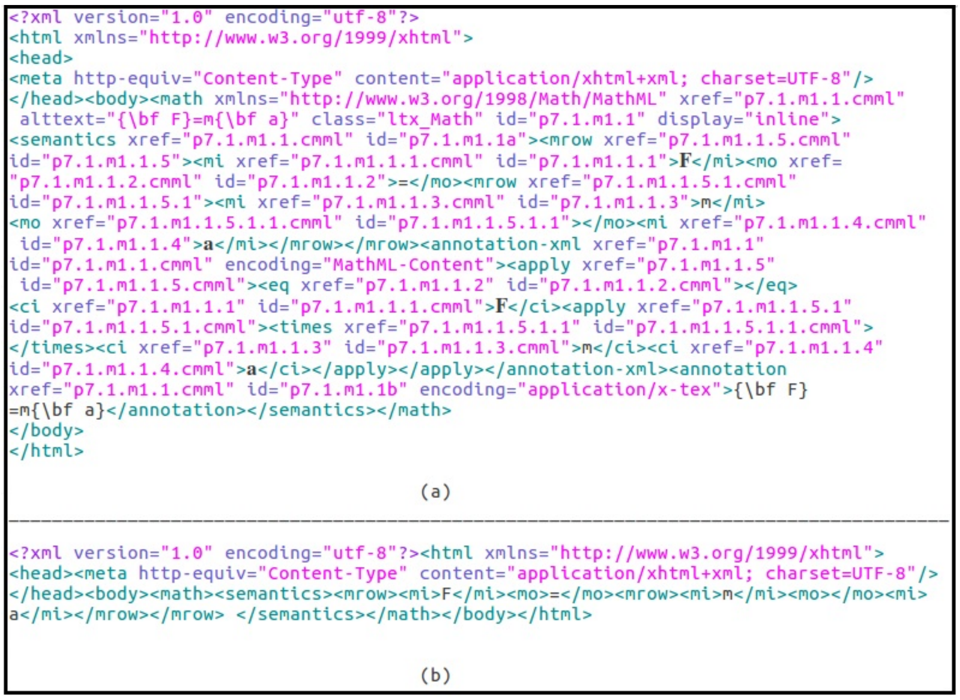
3.2 Document Preprocessor

Core focus of our proposed approach being formulae retrieval and formulae matching, document preprocessor extracts only MathML formulae from the documents containing text as well as math contents. Figure 2 shows a sample XHTML document containing text as well as math contents. Figure 3(a) shows output of document preprocessor wherein text contents have been removed.



```
<?xml version="1.0" encoding="utf-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="application/xhtml+xml; charset=UTF-8" />
</head><body>
<div class="ltx_para" id="p7">
<p class="ltx_p" id="p7.1">In other words, while any classical Lagrangian and any
wave equation (or field equation in the case of second
quantization) is an evolution of Newton’s second law
<math xmlns="http://www.w3.org/1998/Math/MathML" xref="p7.1.m1.1.cmml" alttext=
"{\bf F}=m{\bf a}" class="ltx_Math" id="p7.1.m1.1" display="inline"><semantics
xref="p7.1.m1.1.cmml" id="p7.1.m1.1a"><mrow xref="p7.1.m1.1.5.cmml" id="p7.1.m1.1.5">
<mi xref="p7.1.m1.1.1.cmml" id="p7.1.m1.1.1">F</mi><mo xref="p7.1.m1.1.2.cmml"
 id="p7.1.m1.1.2">=</mo><mrow xref="p7.1.m1.1.5.1.cmml" id="p7.1.m1.1.5.1">
<mi xref="p7.1.m1.1.3.cmml" id="p7.1.m1.1.3">m</mi><mo xref="p7.1.m1.1.5.1.1.cmml"
 id="p7.1.m1.1.5.1.1"></mo><mi xref="p7.1.m1.1.4.cmml" id="p7.1.m1.1.4">a</mi></mrow>
</mrow><annotation-xml xref="p7.1.m1.1" id="p7.1.m1.1.cmml" encoding="MathML-Content">
<apply xref="p7.1.m1.1.5" id="p7.1.m1.1.5.cmml">
<eq xref="p7.1.m1.1.2" id="p7.1.m1.1.2.cmml"></eq><ci xref="p7.1.m1.1.1"
id="p7.1.m1.1.1.cmml">F</ci><apply xref="p7.1.m1.1.5.1" id="p7.1.m1.1.5.1.cmml">
<times xref="p7.1.m1.1.5.1.1" id="p7.1.m1.1.5.1.1.cmml"></times><ci xref="p7.1.m1.1.3"
 id="p7.1.m1.1.3.cmml">m</ci><ci xref="p7.1.m1.1.4" id="p7.1.m1.1.4.cmml">a</ci>
</apply></apply></annotation-xml><annotation xref="p7.1.m1.1.cmml" id="p7.1.m1.1b"
 encoding="application/x-tex">{\bf F}=m{\bf a}</annotation></semantics></math>,
in the approach mentioned above
inertia follows from the anisotropic radiation pressure. In
this way, inertia appears as a consequence of the third law,
i.e., of momentum conservation, and ultimately of
translation invariance.</p>
</div>
</body></html>
```

Fig. 2. A sample XHTML document containing text as well as math formulae



```
<?xml version="1.0" encoding="utf-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="application/xhtml+xml; charset=UTF-8"/>
</head><body><math xmlns="http://www.w3.org/1998/Math/MathML" xref="p7.1.m1.1.cmml"
 alttext="{\bf F}=m{\bf a}" class="ltx_Math" id="p7.1.m1.1" display="inline">
<semantics xref="p7.1.m1.1.cmml" id="p7.1.m1.1a"><mrow xref="p7.1.m1.1.5.cmml"
id="p7.1.m1.1.5"><mi xref="p7.1.m1.1.1.cmml" id="p7.1.m1.1.1">F</mi><mo xref=
"p7.1.m1.1.2.cmml" id="p7.1.m1.1.2">=</mo><mrow xref="p7.1.m1.1.5.1.cmml"
id="p7.1.m1.1.5.1"><mi xref="p7.1.m1.1.3.cmml" id="p7.1.m1.1.3">m</mi>
<mo xref="p7.1.m1.1.5.1.1.cmml" id="p7.1.m1.1.5.1.1"></mo><mi xref="p7.1.m1.1.4.cmml"
 id="p7.1.m1.1.4">a</mi></mrow></mrow><annotation-xml xref="p7.1.m1.1"
id="p7.1.m1.1.cmml" encoding="MathML-Content"><apply xref="p7.1.m1.1.5"
 id="p7.1.m1.1.5.cmml"><eq xref="p7.1.m1.1.2" id="p7.1.m1.1.2.cmml"></eq>
<ci xref="p7.1.m1.1.1" id="p7.1.m1.1.1.cmml">F</ci><apply xref="p7.1.m1.1.5.1"
id="p7.1.m1.1.5.1.cmml"><times xref="p7.1.m1.1.5.1.1" id="p7.1.m1.1.5.1.1.cmml">
</times><ci xref="p7.1.m1.1.3" id="p7.1.m1.1.3.cmml">m</ci><ci xref="p7.1.m1.1.4"
id="p7.1.m1.1.4.cmml">a</ci></apply></apply></annotation-xml><annotation
xref="p7.1.m1.1.cmml" id="p7.1.m1.1b" encoding="application/x-tex">{\bf F}
=m{\bf a}</annotation></semantics></math>
</body>
</html>
```

(a)

```
<?xml version="1.0" encoding="utf-8"?><html xmlns="http://www.w3.org/1999/xhtml">
<head><meta http-equiv="Content-Type" content="application/xhtml+xml; charset=UTF-8"/>
</head><body><math><semantics><mrow><mi>F</mi><mo>=</mo><mrow><mi>m</mi><mo></mo><mi>
a</mi></mrow></mrow> </semantics></math></body></html>
```

(b)

Fig. 3. (a) Output of Document Preprocessor (b) Output of Formula Preprocessor

3.3 Formula Preprocessor

Primary job of formula preprocessor is to filter out unnecessary MathML elements and attributes which barely contribute to information content of the formula. Table 2 shows some examples of preprocessing done by formula preprocessor. Figure 3(b) shows a complete output of formula preprocessor wherein only relevant and requisite information of the raw formula have been preserved.

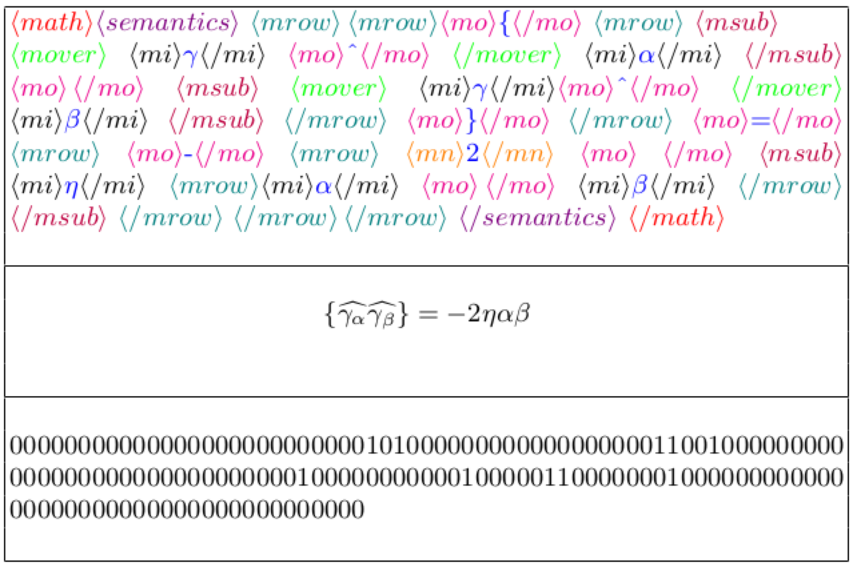
3.4 Formula Embedding Module

Formula embedding module takes processed formula as input and outputs a corresponding binary vector. Careful observation of Presentation MathML formulae guides us to the fact that $\langle mi \rangle$ and $\langle mo \rangle$ elements form the key constituents of nearly all formulae. Thus, the module parses processed MathML formula to fetch contents from all occurrences of $\langle mi \rangle$ and $\langle mo \rangle$ elements. Besides, the module also accounts for all other MathML elements, such as $\langle msqrt \rangle$, $\langle msub \rangle$, $\langle msup \rangle$, and so on, present in the formula. Eventually, the module uses information of fetched contents and the bit position information table to set respective bits of corresponding formula vector which has a fixed length of 150 bits. A bit position information table, see Table 3, depicts bit positions assigned to different mathematical entities. A vector of length 150 suffices to encode mathematical

Table 2. Preprocessing done by Formula Preprocessor

Original	Processed
⟨*mi xref*=“*p7.1.m1.1.1.cmml*” *id*=“*p7.1.m1.1.1*”⟩	⟨*mi*⟩
⟨*mo xref*=“*p7.1.m1.1.2.cmml*” *id*=“*p7.1.m1.1.2*”⟩	⟨*mo*⟩
⟨*math xmlns*=“*http://www.w3.org/1998/Math/MathML*”⟩	⟨*math*⟩
⟨*semantics xref*=“*p7.1.m1.1.1.cmml*” *id*=“*p7.1.m1.1.1*”⟩	⟨*semantics*⟩
⟨*annotation-xml*⟩...⟨/*annotation-xml*⟩	Contents between the two elements are removed
⟨*annotation*⟩...⟨/*annotation*⟩	Contents between the two elements are removed

contents of all the documents in our corpus. However, module offers flexibility to vary (increase or decrease) length of formula vector, if required. Figure 4 shows typeset representation and formula vector of a processed MathML formula.


⟨*math*⟩⟨*semantics*⟩ ⟨*mrow*⟩ ⟨*mrow*⟩⟨*mo*⟩{⟨/*mo*⟩ ⟨*mrow*⟩ ⟨*msub*⟩ ⟨*mover*⟩ ⟨*mi*⟩γ⟨/*mi*⟩ ⟨*mo*⟩ˆ⟨/*mo*⟩ ⟨/*mover*⟩ ⟨*mi*⟩α⟨/*mi*⟩ ⟨/*msub*⟩ ⟨*mo*⟩ ⟨/*mo*⟩ ⟨*msub*⟩ ⟨*mover*⟩ ⟨*mi*⟩γ⟨/*mi*⟩⟨*mo*⟩ˆ⟨/*mo*⟩ ⟨/*mover*⟩ ⟨*mi*⟩β⟨/*mi*⟩ ⟨/*msub*⟩ ⟨/*mrow*⟩ ⟨*mo*⟩}⟨/*mo*⟩ ⟨/*mrow*⟩ ⟨*mo*⟩=⟨/*mo*⟩ ⟨*mrow*⟩ ⟨*mo*⟩-⟨/*mo*⟩ ⟨*mrow*⟩ ⟨*mn*⟩2⟨/*mn*⟩ ⟨*mo*⟩ ⟨/*mo*⟩ ⟨*msub*⟩ ⟨*mi*⟩η⟨/*mi*⟩ ⟨*mrow*⟩⟨*mi*⟩α⟨/*mi*⟩ ⟨*mo*⟩ ⟨/*mo*⟩ ⟨*mi*⟩β⟨/*mi*⟩ ⟨/*mrow*⟩ ⟨/*msub*⟩ ⟨/*mrow*⟩ ⟨/*mrow*⟩ ⟨/*mrow*⟩ ⟨/*semantics*⟩ ⟨/*math*⟩

$$\{\widehat{\gamma_\alpha}\widehat{\gamma_\beta}\} = -2\eta\alpha\beta$$

```
00000000000000000000000000101000000000000000000011001000000000
00000000000000000000010000000000010000011000000010000000000000
00000000000000000000000000
```

Fig. 4. Typeset Representation and Formula Vector of a processed MathML formula

The following points are worth noticing about bit position information table and formula vector:

1. Bit positions 0–25, 57–65 and 71–100 correspond to content of ⟨*mi*⟩ tag, bit positions 26–45, 66–70 and 101–149 refer to contents of ⟨*mo*⟩ tag and bit

positions 46–56 refer to essential MathML tags which contribute to semantics of formula.

2. The list of symbols is by no means exhaustive but it accounts for nearly all the symbols contained in our corpus. In future version of proposed system the length of formula vector will be extended to accommodate new math symbols.
3. In order to retrieve specific results ahead of non-specific ones, we have maintained distinction between single alphabet variables by assigning them different bit positions, ranging from 0–25. However, we make no distinction between cases of variables and the different cases of same variable are assigned same bit position. For example, 'a' and 'A' are assigned same bit position equal to 0. Furthermore, since the entities "exp" and 'e' are interchangeably used, they have been assigned same bit position equal to 4. For the same reason, "log" and "ln" have been assigned same bit position.
4. Proposed system seeks generalized results for query term involving trigonometric ratios, such as sin, cos, tan, cot, and so on. Thus, all such ratios have been assigned same bit position equal to 90.
5. An entity which can be part of $\langle mi \rangle$ as well as $\langle mo \rangle$ tags, i.e. one which can act as variable as well as operator, has been assigned two distinct bit positions. One such entity is Σ which has been assigned bit positions 76 and 112 designating variable and operator, respectively.
6. Bit position 65 designates a multi character variable whose name is not a standard variable name such as, lim, log, gcd, and so on.
7. Formula vector does not account for multiple occurrences of the entities. Even though an entity occurs more than once in the formula, only one corresponding bit of formula vector is set. This limitation, however, causes inability to retrieve relevant and precise results if the user query contains repetition of an entity.

However, none of the above constraints incurs loss of generality and the approach is scalable. Scalability can be ensured through improvement in search and indexing technique

3.5 Indexer

Indexer stores formula vector to corresponding document mapping in an index. System indexes a total of 5,969 processed formulae, derived from 212 documents of the corpus. Moreover, the index size is 1 MB which is 4.42% of the corpus size. Corpus size and the size of formula vector are the primary factors which affect size of the index. Indexer indexes formulae in document wise fashion, which means that only after having indexed all the formulae of a document, it goes for indexing next document of the corpus. However, it is observed that '0' bits of a formula vector, which designate absence of mathematical entities, unnecessarily increase the size of index. Our primary concern being presence of an entity, future modification will refrain from storing '0' bit information in the index.

Table 3. Bit Position Information Table

Entity	Position	Entity	Position	Entity	Position	Entity	Position
a/A to z/Z	0-25	⟨msqrt⟩	56	log, ln	88	$\pm$	120
exp	4	$\mathbb{Z}$	57	!	89	$\int$	121
$=$	26	$\mathbb{N}$	58	Trigo. Ratios	90	$\circ$	122
Product	27	$\mathbb{Q}$	59	$\mathbb{R}$	91	$'$	123
-	28	$\urcorner$	60	θ	92	$\wedge$	124
,	29	α	61	gcd	93	$\exists$	125
$+$	30	γ	62	xor	94	$\neg$	126
∇	31	ω	63	τ	95	lim	127
∂	32	ϑ	64	η	96	$<,\langle$	128
$\rightarrow$	33	Var. Name	65	σ	97	$>,\rangle$	129
.	34	Null	66	$\mathit{\Omega}$	98	$\otimes$	130
(	35	$\dagger$	67	#	99	$\vdash$	131
)	36	:	68	$\ulcorner$	100	$\sqcap$	132
$\equiv$	37	dist	69	$\neq$	101	$\sqcup$	133
$\gg$	38	$\mp$	70	{	102	$\Vert$	134
$\propto$	39	ϕ, φ, $\mathit{\Phi}$	71	}	103	$\cup$	135
$\approx$	40	$\hbar$	72	$\odot$	104	$\cap$	136
/	41	π	73	$\leq$	105	$\mapsto$	137
$\subseteq$	42	$\triangle$	74	$\in$	106	$\subset$	138
$\oplus$	43	μ	75	[	107	det	139
$\sim$	44	$\mathit{\Sigma}$	76	]	108	$\prod$	140
$\vert$	45	$\in$	77	$*$, x	109	mod	141
⟨mfrac⟩	46	...	78	$\notin$	110	sup	142
⟨mn⟩	47	δ	79	$\hat{}$	111	$\geq,\geqslant,\gtrsim$	143
⟨msub⟩	48	ψ, $\mathit{\Psi}$	80	$\mathit{\Sigma}$	112	dim	144
⟨msup⟩	49	$\mathit{\Gamma}$	81	;	113	$:=$	145
⟨msubsup⟩	50	∞	82	$\bar{}$	114	$\cong$	146
⟨mover⟩	51	ρ	83	$\Longleftrightarrow$, $\Leftrightarrow$	115	max	147
⟨munderover⟩	52	β	84	$\Rightarrow$	116	inf	148
⟨munder⟩	53	λ	85	$\lceil$	117	min	149
⟨mtable⟩	54	ξ	86	$\rceil$	118		
⟨mmultiscripts⟩	55	$\square$	87	$\forall$	119		

3.6 Query Preprocessor

The task of query preprocessor resembles that of document and formula preprocessors. It extracts math formula from the user query (text+math) and processes it to remove unnecessary MathML elements and attributes.

3.7 Query Embedding Module

The module transforms processed user query into a binary query vector in a way similar to formula embedding module. An important observation is that some NTCIR queries comprise of query variable (qvar) which may correspond to a variable name, numeric value or complex expression. Without loss of generality, the module assumes qvar to be numeric value and sets the bit position 47, corresponding to element $\langle mn \rangle$, equal to 1 if *qvar* is encountered.

3.8 Searcher Module

Given a user query in vector form, primary objective of searcher module is to retrieve relevant search results. Module uses number of matching set bits of query vector and formula vector as the criterion to judge relevance of formula vector and hence the search results. Although the criterion is not foolproof, it works effectively well for majority of user queries. Probable errors and problems of selected criterion have been comprehensively analyzed, with suitable examples, in subsection 5.4. The module retrieves 30 most similar top ranked documents corresponding to each user query. It should be noted that a query vector may match different formulae of same document with different scores and if such a document succeeds to find place in the result list, the module reports highest of all the scores. In any case, the module refrains from reporting redundant results which may hamper the evaluation process.

Eventually, the system results are fed to trec_eval evaluation tool which compares them with Gold dataset entries and outputs evaluation report which summarizes effectiveness of system in terms of a number of parameters.

4 Experimental Design

This section details Queryset, Gold Dataset and Evaluation Parameters used to inspect system's robustness and efficiency.

4.1 Queryset Description

Our queryset comprises of 23 MathML queries derived from NTCIR-12 MathIR Wikipedia Main Task and Wikipedia Formula Browsing Task querysets. The queryset is a mix of simple and complex queries wherein each query is characterized by a QueryID. While selecting documents for our corpus, it has been ensured that the documents embody either exact form or similar form or sub-formula form of query formulae in the queryset. Each of the queries in queryset is transformed into a query vector by the query embedding module.

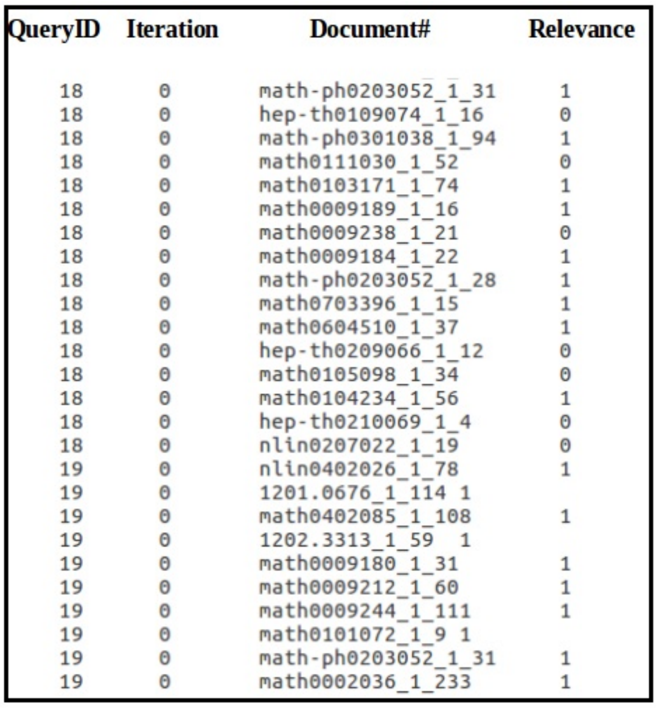
4.2 Gold Dataset

Gold Dataset (also known as qrel file) strictly adheres to the Text REtrieval Conference (TREC) qrel format[3] and comprises of a set of human assessed documents for each query in the queryset. Gold dataset has the format:

QueryID	*Iteration*	*Document#*	*Relevance*

[3] http://trec.nist.gov/data/qrels_eng/

where, QueryID designates specific query, Iteration is an irrelevant field usually set to 0 and ignored by trec_eval tool, Document# specifies document number of the retrieved document and Relevance designates binary judgment (1 for relevant and 0 for irrelevant). A document is judged relevant even if it contains small content of the query formula. Our Gold Dataset comprises of 690 entries wherein 77 entries have been judged irrelevant. Figure 5 shows snapshot of Gold Dataset.



QueryID	Iteration	Document#	Relevance
18	0	math-ph0203052_1_31	1
18	0	hep-th0109074_1_16	0
18	0	math-ph0301038_1_94	1
18	0	math0111030_1_52	0
18	0	math0103171_1_74	1
18	0	math0009189_1_16	1
18	0	math0009238_1_21	0
18	0	math0009184_1_22	1
18	0	math-ph0203052_1_28	1
18	0	math0703396_1_15	1
18	0	math0604510_1_37	1
18	0	hep-th0209066_1_12	0
18	0	math0105098_1_34	0
18	0	math0104234_1_56	1
18	0	hep-th0210069_1_4	0
18	0	nlin0207022_1_19	0
19	0	nlin0402026_1_78	1
19	0	1201.0676_1_114 1	
19	0	math0402085_1_108	1
19	0	1202.3313_1_59 1	
19	0	math0009180_1_31	1
19	0	math0009212_1_60	1
19	0	math0009244_1_111	1
19	0	math0101072_1_9 1	
19	0	math-ph0203052_1_31	1
19	0	math0002036_1_233	1

Fig. 5. Snapshot of Gold Dataset

4.3 Evaluation Parameters

System results have been evaluated on grounds of *P_5*, *P_10*, *map* and *bpref*. All these measures are computed for each query and individual measures are then averaged over all the queries in the queryset. *P_k*, with *k* being equal to 5 and 10, refers to the count of relevant documents out of first *k* retrieved documents. In order to compute *map*, precision score is computed whenever a relevant document is retrieved. The precision scores are then averaged over all the relevant documents, corresponding to a query, to get average precision. Average precision scores are averaged over all the queries to get *map*. Furthermore, *bpref* measures ability of system to retrieve relevant documents ahead of irrelevant ones. Besides, we have also computed fraction of relevant documents which are retrieved (denoted as *frac_ret*). Number of relevant documents (*num_rel*) divided by number of retrieved relevant documents (*num_rel_ret*) gives the measure of *frac_ret*. Values of *num_rel* and *num_rel_ret*, used to compute *frac_ret*, can be obtained from evaluation report. All the five parameters range from 0 to 1 with larger values signifying better.

5 Results and Analysis

5.1 Format of Result Set

The Result Set, embodying system results, strictly adheres to result file format of TREC. Each tuple of result set attains the form:

QueryID	*Iteration*	*Document#*	*Rank*	*Similarity*	*Run_ID*

Although *Iteration (iter)* and *Rank* fields are mandatory, the two are ignored by evaluation tool. *Similarity (sim)* is usually float in nature and attains larger value for documents which are retrieved first. In our case, count of matching set bits refers to *Similarity* score. *Run_ID*, a trivial field, is a string which characterizes system run and gets printed on the evaluation report. Trivial fields, namely *Iteration*, *Rank* and *Run_ID*, have been set to dummy values "Q0", "20" and "demo", respectively. Figure 6 shows snapshot of Result Set.

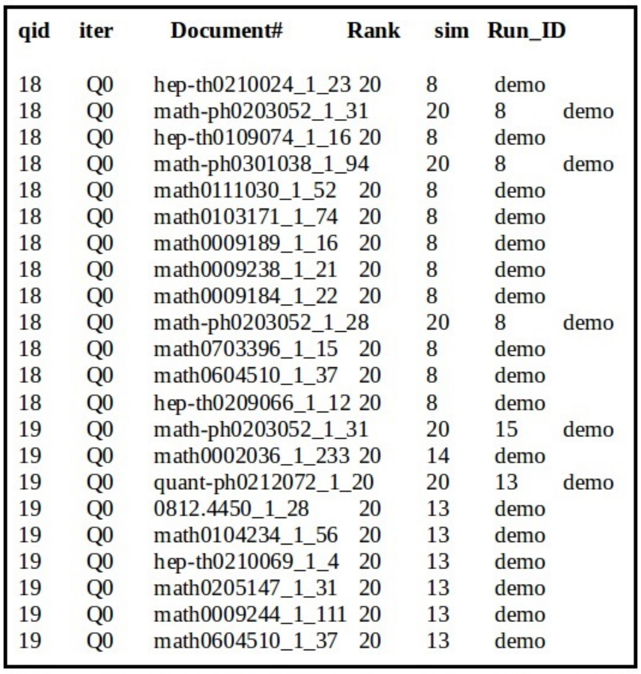
5.2 Evaluation Report

Evaluation report, generated by trec_eval, contains values of a number of parameters which summarize effectiveness of proposed system. However, only 5 parameters, discussed in subsection 4.3, have been used to infer system's effectiveness. Labeled bar chart, shown in Figure 7, depicts values of evaluation parameters for the proposed system. Fairly high values of the parameters substantiate effectiveness of proposed approach. Out of a total of 613 relevant documents, system succeeds in retrieving 535 documents which equates to 87.27% (frac_ret=0.8727) of the total relevant documents.

5.3 Comparison with Conventional Text Search Engine

The same experimental framework (Corpus, Gold Dataset and Query Set) has been used to retrieve results from Apache Nutch[4] based conventional text search engine. Labeled bar chart, shown in Figure 8, depicts comparison of evaluation parameters for the two systems. Significant differences in corresponding evaluation measures are attributed to the fact that math search is way different than conventional text search in terms of retrieval needs of end users and the way query term is matched to indexed terms. In particular, incompetence of conventional search engine is reflected in its ability to retrieve only 54 out of 613 relevant documents which equates to 8.81% (frac_ret=0.0881) of total relevant documents.

[4] http://nutch.apache.org/



qid	iter	Document#	Rank	sim	Run_ID
18	Q0	hep-th0210024_1_23	20	8	demo
18	Q0	math-ph0203052_1_31	20	8	demo
18	Q0	hep-th0109074_1_16	20	8	demo
18	Q0	math-ph0301038_1_94	20	8	demo
18	Q0	math0111030_1_52	20	8	demo
18	Q0	math0103171_1_74	20	8	demo
18	Q0	math0009189_1_16	20	8	demo
18	Q0	math0009238_1_21	20	8	demo
18	Q0	math0009184_1_22	20	8	demo
18	Q0	math-ph0203052_1_28	20	8	demo
18	Q0	math0703396_1_15	20	8	demo
18	Q0	math0604510_1_37	20	8	demo
18	Q0	hep-th0209066_1_12	20	8	demo
19	Q0	math-ph0203052_1_31	20	15	demo
19	Q0	math0002036_1_233	20	14	demo
19	Q0	quant-ph0212072_1_20	20	13	demo
19	Q0	0812.4450_1_28	20	13	demo
19	Q0	math0104234_1_56	20	13	demo
19	Q0	hep-th0210069_1_4	20	13	demo
19	Q0	math0205147_1_31	20	13	demo
19	Q0	math0009244_1_111	20	13	demo
19	Q0	math0604510_1_37	20	13	demo

Fig. 6. Snapshot of Result Set

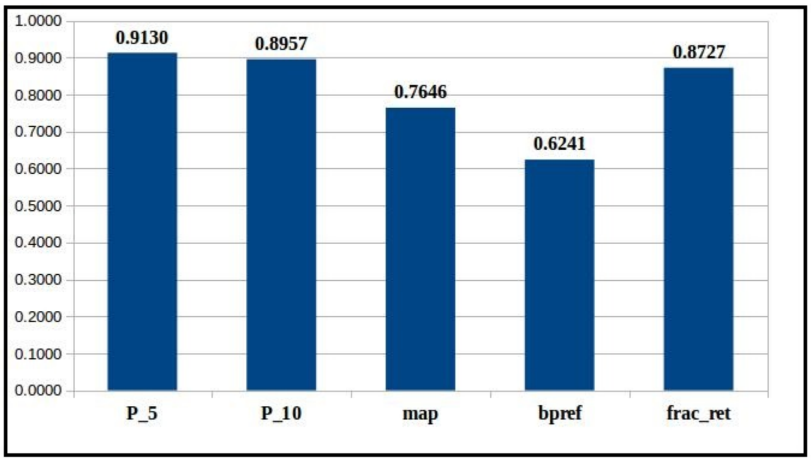


Fig. 7. Bar graph showing values of evaluation parameters for the proposed system

5.4 Results Analysis

In this subsection, we have comprehensively analyzed strengths and possible limitations of proposed method from different perspectives. Following observations and system outcomes are worth considering:

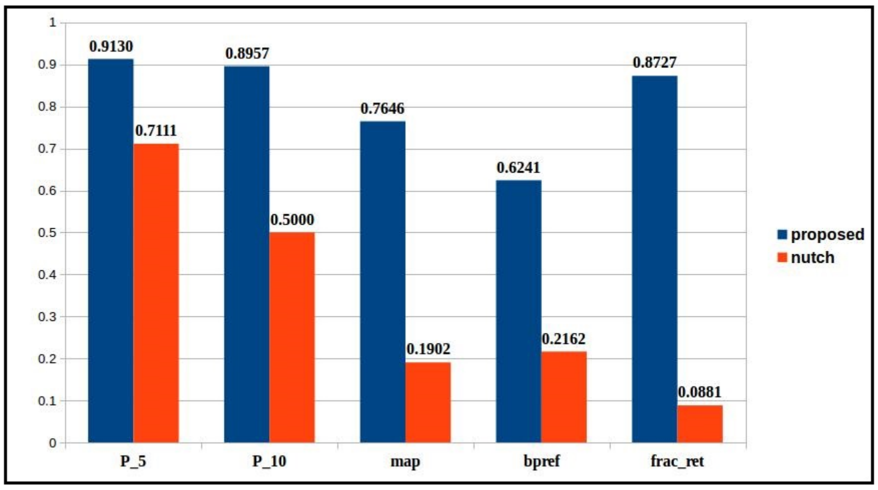


Fig. 8. Bar graph showing comparison of proposed system and conventional text search engine

1. First observation is that given a query formula, system flawlessly retrieves all search results wherein the query term is either present in its exact form or present as sub-formula of a larger parent formula. This actually happens because similarity score (number of matching set bits of query vector and the search result) for such results will be maximum. For example, Table 4 shows some of the search results for the user query $a \oplus b$. In first and second search results, the query term appears in its exact form whereas in third and fourth search results, it appears as sub-formula of its parent formula. Moreover, the maximum similarity score is 3 as the query formula comprises of three distinct entities, namely a, $\oplus$ and b.

Table 4. Search results for the user query $a \oplus b$

Search Results	Documents	Similarity Score
$a \oplus b$	hep-th0005087_1_93	3
$b \oplus a$	1310.4652_1_21	3
$a \oplus b = b$	1209.1718_1_16	3
$(a, b) \in A \oplus B$	math0011063_1_119	3
$M(B) = M(A) \oplus M(A^{-1}B)$	math0102078_1_5	3
${W_P}^k = {A_P}^k \oplus {B_P}^k$	math0206213_1_24	3

2. Second observation is that even the complex formulae, which share semantic relatedness with user query and contain all its entities, are successfully retrieved. For example, the fifth and sixth search results (see Table 4) neither contain exact query formula nor its parent form but they are retrieved by

our system as they semantically resemble the query formula and contain all its entities.

3. Third observation is that the proposed system successfully retrieves sub-formulae and similar formulae of a given user query. Consider, for example, the search results for user query $O(mn \log m)$, shown in Table 5:

Table 5. Search results for the user query $O(mn \log m)$

Search Results	Documents	Similarity Score
$O(\log n + \mathrm{k})$	0805.1348_1_4	5
$O(\sqrt{\log n)}$	quant-ph0205083_1_82	5
$O(n \overset{1}{\overline{\log_{p} m}})$	quant-ph0211179_1_57	5
$O(\log n)$	0805.1348_1_32	5
$O(nt \log n)$	1308.3898_1_18	5

Snippets, shown in first 4 search results, depict sub-formulae of the user query. Also, the fifth search result is semantically similar to the user query.

4. Fourth observation is that although the decision of correlating relevance with high similarity score predominantly works well and seems rational, it does incur failure in certain cases. Consider, for example, the search results for the user query ${}_2F_1(a, b; c; z)$, shown in Table 6:

Table 6. Search results for the user query ${}_2F_1(a, b; c; z)$

Search Results	Documents	Similarity Score
${}_2F_1(a+1, 1; 2; z)$	math-ph0203052_1_8	10
${}_2F_1(k+1, k+1; 1; \lambda)$	quant-ph0212072_1_20	7
${}_2F_1(-n, b; \gamma; y)$	math-ph0203052_1_21	7
$\widetilde{h} = \widetilde{W}^{-1} dx^2 + h_{AB}(x, x^C) dx^A dx^B$	hep-th0108170_1_12	7

The first three search results are exceedingly good but the last search result is absolutely irrelevant. Such an irrelevant result succeeds in finding place in the result list, essentially because it contains majority of query formula entities, namely ‘A’, ‘B’, ‘C’, ‘,’, ‘(’, ‘)’and ‘1’.

5. Fifth observation is that for the user query $F = ma$, system retrieves an irrelevant search result, namely $\int_a^m F(x)dx$ embodying all the entities of query formula, among top-ranked search results. An even painful truth is that system fails to retrieve a commendable formula, namely $F = bc$, primarily because of its low similarity score equal to 2 and the constraint laid on the system to retrieve only top 30 documents. Although an increase in search result limit will undoubtedly retrieve such commendable formulae, intelligence

lies in system's ability to retrieve such formulae among top-ranked documents, under the imposed constraint. Moreover, the intelligence will also be manifested in ability of system to invalidate irrelevant search results which embody query formula entities but posses little or no semantic relatedness with the query.

Our fourth and fifth observations (pitfalls, so to speak) guide us to the fact that using number of matching set bits of formula vector and query vector as similarity measure criterion is not foolproof. A search result with low similarity score might prove to be more competent and relevant than the one with better score. Thus, even though the proposed system lives up to our expectations as a baseline system, a future extension necessitates revision in similarity measure criterion to overpower existing inabilities.

6 Future Directions

As the system is baseline, list of possible future directions is prolonged. Some salient future directions are undermentioned:

1. Proposed system currently refrains from indexing and searching text contents. Thus, augmenting baseline system with text indexing and searching functionality can add to the evaluation measures and possibly invalidate some of the irrelevant search results.
2. Formula and Query vectors need to be lengthened to accommodate new symbols and possibilities. For example, some bit positions may account for count of occurrences of an entity or co-occurrence of two or more entities which will eventually let the searcher module prefer relevant results over irrelevant ones.
3. Similarity measure criterion can be modified to assign weightage to certain bit positions. Our observation says that it is an inherent tendency of end users to often compromise the variable names but not the operators. For example, given user query to be $a \oplus b$, end user will be more content with $c \oplus d$ than $a + b$, even though the similarity score for latter will be more than the former. Thus, assigning more weightage to operator bit positions, while computing similarity score, can do wonders.
4. Another fascinating idea could be to prompt the end users for key entities of their queries. Search results, not adhering to such key entities, should not be retrieved. This can be ensured by assigning more weightage to bit positions of key entities, specified by end user, while computing the similarity score.

7 Conclusion

In this paper, we describe an unconventional formula embedding approach which can ease retrieval of math formulae contained inside scientific documents. Uncommonly used formula embedding approach opens door to new horizons and helps combat underlying challenges of scientific document retrieval. Having transformed the processed document formulae and the user query into vectors of fixed size, proposed system uses number of matching set bits of formula and query vectors as similarity measure to retrieve indexed formulae and to judge their relevance. Comprehensive appraisal of system results affirms underlying strengths and weaknesses of proposed system. Although the approach has associated limitations, it shows competence in retrieving exact query formula, parent formulae, sub-formulae and similar formulae. Modifying similarity measure criterion and incorporating support for indexing and searching of textual terms are some of the notable future modifications which can further the effectiveness of retrieval.

Acknowledgement

The work presented here falls under the Research Project Grant No. YSS/2015/000988 and supported by the Department of Science & Technology (DST) and Science and Engineering Research Board (SERB), Govt. of India. The authors would like to acknowledge the Department of Computer Science & Engineering, National Institute of Technology Mizoram, India for providing infrastructural facilities and support.

References

1. Aizawa, A., Kohlhase, M., Ounis, I.: NTCIR-10 Math Pilot Task Overview. In: Proceedings of the 10th NTCIR Conference. pp. 654–661. Tokyo, Japan (2013)
2. Aizawa, A., Kohlhase, M., Ounis, I., Schubotz, M.: NTCIR-11 Math-2 Task Overview. In: Proceedings of the 11th NTCIR Conference. pp. 88–98. Tokyo, Japan (2014)
3. Davila, K., Zanibbi, R., Kane, A., Tompa, F.W.: Tangent-3 at the NTCIR-12 MathIR Task. In: Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies. pp. 338–345. Tokyo, Japan (2016)
4. Gao, L., Jiang, Z., Yin, Y., Yuan, K., Yan, Z., Tang, Z.: Preliminary Exploration of Formula Embedding for Mathematical Information Retrieval: Can Mathematical Formulae Be Embedded Like A Natural Language? arXiv preprint arXiv:1707.05154 (2017)
5. Joho, H., Kishida, K.: Overview of NTCIR-11. In: Proceedings of the 11th NTCIR Conference. pp. 9–12. Tokyo, Japan (2014)
6. Joho, H., Sakai, T.: Overview of NTCIR-10. In: Proceedings of the 10th NTCIR Conference. pp. 1–7. Tokyo, Japan (2014)
7. Kishida, K., Kato, M.P.: Overview of NTCIR-12. In: Proceedings of the 12th NTCIR Conference. pp. 1–7. Tokyo, Japan (2016)
8. Líška, M., Sojka, P., Ružicka, M.: Similarity Search for Mathematics: Masaryk University team at the NTCIR-10 Math Task. In: Proceedings of the 10th NTCIR Conference on Evaluation of Information Access Technologies. pp. 686–691. Tokyo, Japan (2013)
9. Líška, M., Sojka, P., Ružicka, M.: Combining Text and Formula Queries in Math Information Retrieval: Evaluation of Query Results Merging Strategies. In: Proceedings of the First International Workshop on Novel Web Search Interfaces and Systems. pp. 7–9. ACM, Melbourne, Australia (2015)

10. Pakray, P., Sojka, P.: An Architecture for Scientific Document Retrieval Using Textual and Math Entailment Modules. In: Proceedings of Recent Advances in Slavonic Natural Language Processing. pp. 107–117. Karlova Studnka, Czech Republic (2014)
11. Pathak, A., Pakray, P., Sarkar, S., Das, D., Gelbukh, A.: MathIRs: Retrieval System for Scientific Documents. Computación y Sistemas 21(2), 253–265 (2017)
12. Ružicka, M., Sojka, P., Líška, M.: 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. pp. 127–134. Tokyo, Japan (2014)
13. Ruzicka, M., Sojka, P., Líska, M.: Math Indexer and Searcher under the Hood: Fine-tuning Query Expansion and Unification Strategies. In: Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies. pp. 331–337. Tokyo, Japan (2016)
14. Schellenberg, T., Yuan, B., Zanibbi, R.: Layout-Based Substitution Tree Indexing and Retrieval for Mathematical Expressions. In: Proceedings of the Document Recognition and Retrieval. pp. 1–8. California, USA (2012)
15. Sojka, P., Líška, M.: The Art of Mathematics Retrieval. In: Proceedings of the 11th ACM symposium on Document engineering. pp. 57–60. California, USA (2011)
16. Thanda, A., Agarwal, A., Singla, K., Prakash, A., Gupta, A.: A Document Retrieval System for Math Queries. In: Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies. pp. 346–353. Tokyo, Japan (2016)
17. Zanibbi, R., Aizawa, A., Kohlhase, M., Ounis, I., Topic, G., Davila, K.: NTCIR-12 MathIR Task Overview. In: Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies. pp. 299–308. Tokyo, Japan (2016)