A Document Retrieval System for Math Queries

Abhinav Thanda, Ankit Agarwal, Kushal Singla, Aditya Prakash, Abhishek Gupta
Samsung R&D Institute India, Bangalore
{abhinav.t89, ankit.ag, kushal.s, abhishek.gu}@samsung.com

ABSTRACT

We present and analyze the results of our Math search system in the MathIR tasks in the NTCIR-12 Information Retrieval challenge.

The Math search engine in the paper utilizes the co-occurrence finding technique of LDA and doc2vec to bring more contextual search. Additionally, it uses common patterns to improve the search output. To combine various scoring algorithms, it uses hybrid ranking mechanism to re-rank the documents retrieved from Elastic Search. In this paper, we evaluate the results from these algorithms and present possible future work for further improvements.

Team Name

SMSG5

Subtasks

MathIR arXiv Main Task (English), optional MathIR Wikipedia Task (English)

Keywords

NTCIR, text search, math formula search

1. INTRODUCTION

In contrast to normal keyword-based search engines, Math search engine poses unique set of challenges. This is due to the fact that a query can contain formula in i) its exact form ii) exact form but in different variable representation iii) different ordering of operator and/or variables and iv) partial form. As a result of these, additional complexity is required in the system to handle such cases.

NTCIR [6] gives its participants a unique opportunity to solve such challenges through Math Information Retrieval task. In particular, NTCIR-12 provided two different types of datasets: ArXiv dataset consisting of more than 100,000 files from arxiv website (with 60M formulae) and Wikipedia dataset with more than 300, 000 articles from Wikipedia website (with more than 500, 000 formulae). The initial processing of documents was done by NTCIR organizers and dataset was provided in the form of xhtml files having math equations in three Math formats. In all, 50 evaluation queries for ArXiv dataset and 30 queries for Wikipedia dataset were published which contained queries from varied domain of Mathematics and contained both keyword and formula queries with formula being in different forms, as discussed earlier. For each query, it is expected to retrieve 1000 responses from the system.

SMSG5 group participated for both the datasets but due to time constraint (our late entry into NTCIR-12 competition) and unavailability of appropriate infrastructure, we could not complete the ArXiv dataset according to the originally planned four runs and could submit only one run results within the stipulated deadline. For Wikipedia dataset, all the four planned runs were submitted, the details of each of them will be discussed in subsequent sections.

2. THE MATH SEARCH SYSTEM

We created a Math Search Engine with the capability of both formula as well as keyword search. The novelty of our search engine lies in:

- First use of Doc2Vec for Math formulae
- Using contextual information (formula+keyword) by exploiting Latent Dirichlet Allocation's (LDA) co-occurrence finding algorithm
- Use of pattern-based approach for common patterns
- Borda count based hybrid ranking system

We first process the data to extract useful information & carry out appropriate formula representation. We then use Elastic Search to index documents which act as a base ranking mechanism. Re-ranking of documents is achieved by using Borda Count-based hybrid ranking over a number of scoring mechanisms: Doc2Vec-based scoring, LDA-based scoring, and Pattern-based scoring along with ES scoring. Both the datasets (ArXiv and Wikipedia) were handled in the similar fashion and hence, a common description is given. In subsequent sections, we describe the overall processing and ranking mechanism.

2.1 Data Processing

Most of the initial processing of datasets is done by NTCIR-12 and represent them in xhtml files with formula being represented in three different formats: Presentation MathML, Content MathML and LaTeX . For our search engine, we used Presentation MathML format because of its simplistic representation compared to other two formats. In the approach we adopted, we removed unnecessary information in the presentation MathML like "$<$ $mathdisplay$ = "$inline$" id = "$3D\_projection$: 1" $>$" and filtered all the formulae with

only relevant MathML form like "$< math >$". IN NTCIR-12 dataset, it is observed that some of the formulae are represented in nested MathML form. This affects identification and matching with query formula. Hence, we identified such cases and re-processed them to represent them in useful form, as shown in fig. 1.

NESTED MATH FORMULA

```
<math xmlns="http://www.w3.org/1998/Math/MathML" xref="S3.p4.1.m1.1.cmml"
alttext="U" class="ltx_Math" id="S3.p4.1.m1.1" display="inline">
<semantics xref="S3.p4.1.m1.1.cmml" id="S3.p4.1.m1.1a">
<mi xref="S3.p4.1.m1.1.1.cmml" id="S3.p4.1.m1.1.1">U</mi>
<annotation-xml xref="S3.p4.1.m1.1" id="S3.p4.1.m1.1.cmml" encoding="MathML-
Content">
<ci xref="S3.p4.1.m1.1.1" id="S3.p4.1.m1.1.1.cmml">U</ci>
</annotation-xml>
<annotation xref="S3.p4.1.m1.1.cmml" id="S3.p4.1.m1.1b"
encoding="application/x-tex">U</annotation>
</semantics>
        <math xmlns="http://www.w3.org/1998/Math/MathML"
        xref="S3.p4.1.m2.1.cmml" alttext="X" class="ltx_Math" id="S3.p4.1.m2.1"
        display="inline">
        <semantics xref="S3.p4.1.m2.1.cmml" id="S3.p4.1.m2.1a">
        <mi xref="S3.p4.1.m2.1.1.cmml" id="S3.p4.1.m2.1.1">X</mi>
        <annotation-xml xref="S3.p4.1.m2.1" id="S3.p4.1.m2.1.cmml"
        encoding="MathML-Content">
        <ci xref="S3.p4.1.m2.1.1" id="S3.p4.1.m2.1.1.cmml">X</ci>
        </annotation-xml>
        <annotation xref="S3.p4.1.m2.1.cmml" id="S3.p4.1.m2.1b"
        encoding="application/x-tex">U</annotation>
        </semantics>
        </math>

</math>
```

PROCESSED MATH FORMULA

```
<math>
 <mi>U</mi>
</math>
<math>
 <mi>X</mi>
</math>
```

Figure 1: Nested MathML issue in NTCIR-12 dataset

Once a formula is appropriately modified, we represent each of the formula in two forms: specialized form with formula in exact version (ex- $< math >< mi > U < /mi >< /math >$) and generalized form with its generalized version (ex- $< math >< mi > * < /mi >< /math >$). For generalized form, all the formulae having any identifier in the $< mi >$ tag is replaced by "*". This feature assures that the matching happens for those formulae which are same in meaning but different in syntax. As an example, $a^2 = b^2 + c^2$ & $x^2 = y^2 + z^2$ are same because they represent the same equation using different syntax. Hence, we convert the two equations in single format: $*^2 = *^2 + *^2$. Additionally, due to complexity of formulae, the two forms are stored in encoded form: $P + S_i$, where P is a fixed prefix for all formulae (choice of these prefixes is arbitrary but it ensures that the combination $P + S_i$ is unique throughout the corpus; Ex- $htam$) and S_i is the i^{th} formula in the corpus, being encoded in base 26 (Ex- for $i = 28$, $S_i = AB$ and hence, $P + S_i = htamAB$). The corresponding files containing the formula are also modified.

2.2 Elastic Search (ES): Indexing and Scoring

2.2.1 Indexing

The modified files (with encoded formulae) are indexed with keyword and formulae. We create two levels of indexes: Level 1 and Level 2. Level 1 index contains $< P + S; formula >$ where key is $P + S$ and value is $formula$. It is created separately both for specialized [ES_INDX1] and generalized forms [ES_INDX2] of the formula. This would be used later for searching the query formula for top best matches and identifying its encoded form. For LDA purpose, it would be used to fetch best match formula and for ES to get top N matching results. Level 2 index contains: $< P + S; filename, formula\_id >$. It is also created separately both for specialized [ES_INDX3] and generalized [ES_INDX4] forms of the formula. This would be useful to retrieve the file names and formula id for the top N formulae retrieved (their encoded forms) from Level 1 for the query.

For the formula field to store mathematical formula, we use custom analyzer for elastic search, as shown in fig. 2.

New Analyzer

```
"MathEx":{
            "type":"pattern",
            "pattern" : "<.*?>|[^<^>]+",
            "flag": "NONE",
            "group":"0"
          }
```

Figure 2: Custom ES analyzer

For keyword index [ES_INDX5], we use standard stopword list to remove high frequency words. In addition to stopwords, we also removed encoded formula: $P+S$. After cleaning the textual part, we then index into elastic search for keywords using normal ES mechanism: $< filename; file\_content >$.

2.2.2 Elastic Search-based Scoring

The NTCIR-12 queries are pre-processed in a similar fashion as that for documents: Presentation MathML, stopwords etc.. After query pre-processing, we generate internal queries for the five ES indexes (ES_INDX1 $\cdots$ ES_INDX5) using multi-search query mechanism of ES, corresponding to terms in the query for each index. One of the reasons of using elastic search is that it supports partial matches and also assigns score to each result depending upon the extent of match. As an example, for user query: $x + y$ Mean Arithmetic, we generated the following set of ES queries: [ES_INDX1] $x+y$ (in MathML), [ES_INDX2] $*+*$ (in MathML), [ES_INDX3] $<spec\_encoded\_formula>$, [ES_INDX4] $<gen\_encoded\_formula>$, [ES_INDX5]$mean\ arithmetic$.

ES search is carried out in phases depending on its requirement for LDA or for ranking of documents/formulae. For LDA, we are interested in the best match formula because LDA does not support partial matching and it models only over the existing formulae in the corpus. Hence, the need for obtaining top best match formula corresponding to the input query formula. In the other phase, we use ES query for ranking of documents and formulae separately. Both of the query phases are similar, the only difference being the number of results retrieved: in first phase, we retrieve topmost result and in other, top N results and hence, we discuss the two phases commonly.

We consider following points while generating ES query:

1. We use a *Boolean query*. That is, a query encompassing multiple sub-queries within.

2. Amongst the 3 sub-queries embodied inside the ES query, one of them is *Match Phrase query*. This query matches the exact mathematical string, if present. The

boost value associated with this query is kept the highest (10.0).

3. Next is a *Match query* with "AND" type. This gives a match, only if all the tokens (given in the query) are present in an index of elastic search. Its boost is kept the second highest (5.0).

4. Lastly, we use a *Match query* with "OR" type. It produces results ranked (boost value = 1.0) as per the number of tokens matched in any of the elastic search index.

5. Also, we use a notable parameter: "SLOP". If in a query, two subsequent tokens occur with S tokens in between, then it will be considered as a match only if SLOP value is greater than S. In other words, the value of slop decides upto what gap two subsequent token in a query can have in the match. In our case, we use $S = 30.0$.

We use default scoring mechanism of ES to score formulae and documents retrieved from ES. For formulae, we use exact match and partial exact match boosting to give higher score to specialized match formulae.

2.3 Doc2Vec-based Scoring

Paragraph vectors or doc2vec models are a class of algorithms which use neural networks to construct distributed representations of arbitrarily long sentences. [4] discusses the application of doc2vec to retrieval of text documents. In this section we explore the application of doc2vec algorithm to MIR. A specific variant of doc2vec algorithm called the distributed bag of words (PV-DBOW) model is used. We extend the DBOW model to represent 2-d expression trees in the form of real valued dense vectors such that structurally similar formulae appear closer in the vector space.

2.3.1 Paragraph Vector

One of the approaches for paragraph vector is distributed bag of words model (PV-DBOW) algorithm where each paragraph is represented by a unique vector which needs to be learned. The vectors are represented as column matrix D. In each iteration of the stochastic gradient descent learning algorithm, a window of text is sampled. From this window a word is sampled randomly. The network is forced to predict the sampled word with the paragraph vector as input.

The model is trained by maximizing the average log probability of a word given the paragraph/document vector as input. The objective function can be written as

$$J(\theta) = \frac{1}{|D|} \sum_{d \in D} \sum_{w \in d} log(P(w|d)) \quad (1)$$

where D is the set of documents, w is a word in document d. In order to speed up the training, hierarchical soft-max algorithm [4][5] is used.

Inferencing

During test time, a new document or paragraph is presented. The paragraph vector for the new document is obtained by performing an inference step using the trained model. The inference step involves adding an additional column to the paragraph matrix D and running the gradient-descent algorithm on the new document. During test time only the paragraph matrix is updated while the rest of the parameters are left unchanged.

Application of PV-DBOW to Math formulae

Mathematical formulae have 2-dimensional structures and can be represented in the form of expression trees (as shown in fig. 3) with operators occupying non-leaf nodes and operands occupying leaf nodes. We define a token as a set of nodes containing a single operator and its immediate child nodes.

In the PV-DBOW model for formulae, each formula is represented by a unique vector. A formula is composed of tokens where each token is mapped to a unique id (uid). The tokens for formula in fig. 3 are listed in Table 1. In each iteration of the learning algorithm for the formula model, we sample a window of tokens and train the model similar to the procedure discussed earlier, with formula vector as input instead of paragraph vector.

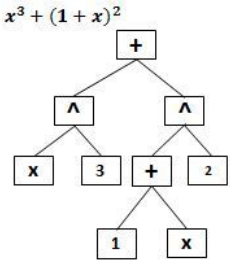

Figure 3: A binary expression tree representation

Table 1: Tokens of formula in fig. 3

Token	Unique Id
{x, ∧, 3}	uid1
{1, +, x}	uid2
{uid2, ∧, 2}	uid3
{uid1, +, uid3}	uid4

2.3.2 Detailed Description

Offline System

Offline system in fig. 4 performs formula and text data preparation, tokenization and generalization of math formulae, training PV-DBOW models for formulae and documents and indexing of formula and document vectors. The training dataset for document vectors consists of text of the document with all its formulae replaced by their unique ids. All the models are trained using Gensim's doc2vec module.

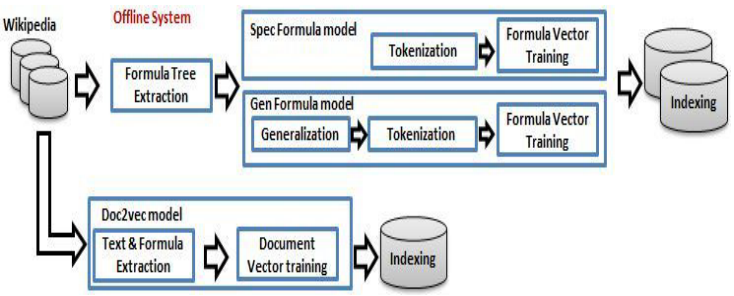

Figure 4: Offline system for Math IR using doc2vec

Generalization
Generalization is the process of modification of a math formula so that all the variables in the formula are replaced by generic variables. This enables the system to match those formulae which are same in all aspects except for the variables. Note that generalization here is different from generalization discussed in earlier sections: here each unique variable is represented by an "id" which is incremented whenever new variable is encountered, with constants being unchanged. Equation 2 gives an example of the generalization form.

$$\begin{aligned} x^2 + y^2 &= z^2 \\ &\implies id_0^2 + id_1^2 = id_2^2 \\ a^2 + b^2 &= c^2 \end{aligned} \tag{2}$$

Tokenization
Tokenization is performed by traversing the formula tree recursively (in-order), with the concept of token being same as discussed in previous section. Each new token is indexed along with a unique token id. Tokenization is performed over both generalized as well as the original specialized formula tree. The two sets of tokens are indexed separately.

Formula model training
A map of formula ids and tokens of the formula is created during tokenization process. The Two models are trained separately for generalized and specialized formulae. Each formula is represented by a 100 dimensional real vector. The window size being 3 and the initial learning rate is set to 0.025. The PV-DBOW model training is run for 50 iterations.

Document model training
A map of file names and their corresponding text is created during data preparation. The PV-DBOW model is trained for fifty iterations. Each document is represented by a real valued vector of 300 dimensions. The window size being 8 and the initial learning rate is set to 0.025.

Online System
The online system is depicted in fig. 5. Generalization and tokenization steps are same as that in the offline case. The formula models are used to generate a real valued vector for the formulae in the query and is referred as *inferencing*, as described earlier. The inferred vectors are then used to fetch the most similar formulae from the indexed formulae using cosine similarity score. The most similar formulae are mapped to their corresponding documents. The formula in the query is replaced by id of the highest scoring formula obtained from the formula model. The complete query including the text and formula is fed to the document model and most similar documents are retrieved. Finally, the documents retrieved from the formula and document models are ranked using Borda ranking algorithm to get final scores of retrieved documents.

Ranking
Borda count method is used to rank the results obtained from formula and document models. The Borda method [2] is based on the Borda count voting method where each

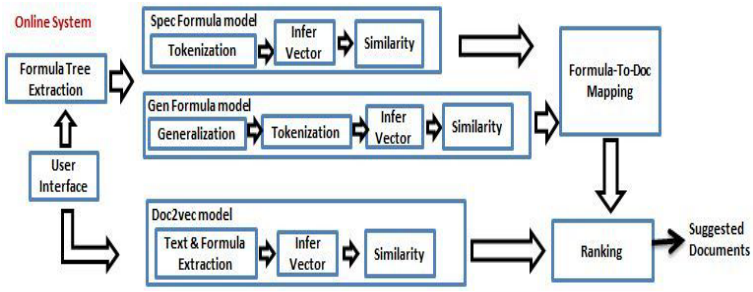

Figure 5: Online system for Math IR using doc2vec

search engine is seen as a voter. Each voter ranks a fixed number c of candidates, where the first candidate is given a score of c, the second $c-1$, and so on. If there are any candidates left unranked by the voter, the remaining points are evenly distributed among them:

$$s_\tau(x) = \begin{cases} 1 - \frac{\tau(x)-1}{|\Omega_\tau|}, & \text{if } x \in \Omega_\tau \\ \frac{|\Omega| - |\Omega_\tau| + 1}{2 * |\Omega|}, & \text{otherwise} \end{cases} \tag{3}$$

where
Ω: the universe of information objects to be ranked.
P: the set of rank sources to be combined.
τ: a rank source $\tau \in$.
Ω_τ: the set of items $\Omega_\tau \subset \Omega$ returned by τ.
$\tau(x)$: given $x \in \Omega_\tau$, $\tau(x)$ is the position of x in the ranking returned by τ.
$s_\tau(x)$: the normalized score for x corresponding to τ.
$sp(x)$: the final combined score for x.

2.3.3 Discussion

Tables 6(a) and 6(b) show the formulae retrieved from the specialized formula model for two different queries. We observe that our technique performs a basic form of "normalization", especially in case of simpler formulae (row 1 of table 6(a)). Also partial matches corresponding to the query formulae are retrieved (row 3). However, this normalization is not consistent across all formulae. Furthermore, formulae with same tree structure but containing different variables (leaf nodes) are not retrieved. This necessitated the use of separate models for generalized and specialized formulae. The results along with generalized formulae show an improvement, as shown in last 3 rows of table 6(a) and table 6(b). However, we observed that sometimes the results retrieved are completely unrelated to the query (row 3 of table 6(b)). Hence, in order to make the results more reliable, we use doc2vec along with elastic search rather than as a standalone system.

The document model trained on text and formulae combined performs as expected. Figure 7 shows a plot of a subset of documents chosen randomly from seven topics where the topics & corresponding documents were obtained by parsing wikipedia category pages (For example- `https://en.wikipedia.org/wiki/Category:Algebra`). The dimensions are reduced from 300 to 2 using t-SNE algorithm [3]. Documents related to similar topics appear together.

2.4 LDA-based Scoring

As discussed earlier, corresponding to query formula, we retrieve topmost match (representative)formula from ES, en-

	Nearest Formula	Distance	Document
Specialized query results	$z^2 = x^2 + y^2$	0.6513	Pseudomanifold
	$x^2 + y^2 = z^2$	0.6472	Formulas for generating Pythagorean triples
	$x^2 + y^2 = c$	0.5925	Orthogonal trajectory
Generalized query results	$a^2 + b^2 = c^2$	0.7393	AMS·LaTeX
	$x^2 + y^2 = L^2$	0.7372	EcosimPro
	$r^2 + h^2 = d^2$	0.7355	Slant height

(a) Query: $x^2 + y^2 = z^2$

	Nearest Formula	Distance	Document
Specialized query results	${}_2F_1(a,b;c;z)$	0.7360	Generalized hypergeometric function
	${}_2F_1(a,b;c;z)$	0.7225	K-noid
	$y(t) = A_c \cos(2\pi \int_0^t f(\tau) d\tau$	0.5819	Frequency modulation
Generalized query results	${}_2F_1(a,b;c;z)$	0.8279	Generalized hypergeometric function
	${}_2F_1$	0.6013	Binomial Transform
	${}_2F_1(-n,\alpha\ \alpha+\beta;1-e^t)$	0.5968	Beta Binomial distribution

(b) Query: ${}_2F_1(a,b;c;z)$

Figure 6: Nearest formulae and their cosine distances

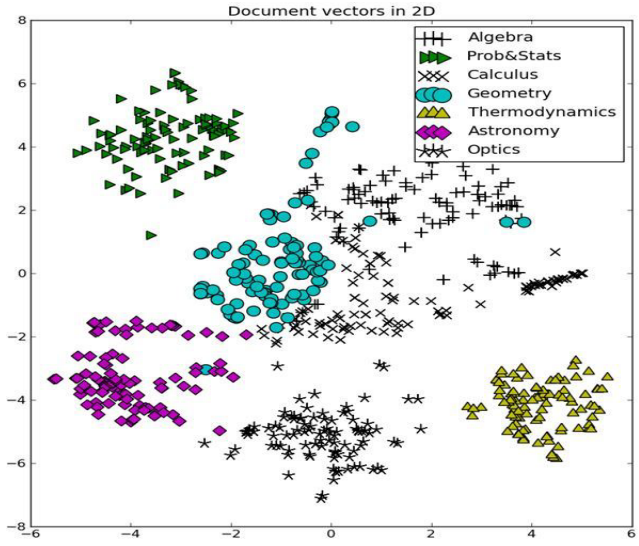

Figure 7: Subset of documents plotted in 2D. Documents of similar topics appear together

code the formula, combine it with query keywords and use the complete formed query text for LDA inferencing. In the offline part, we use the prepared document files (with encoded formula) and use them to generate two different LDA models: specialized LDA model and generalized LDA model. We use Gensim's online LDA to create these LDA models and also, for inferring topics of the query.

Apart from the two models, we also generate two files which list down the topics of each cleaned file after querying from respective model files and index these topics into ES in the format: $< Id; Topicprob, Topic >$ where Id is the file name, $Topicprob$ is the ordered(descending) list of all topics & their probabilities for that document and $Topic$ is the list of top 10 topic ids for that document.

LDA processing is based on the assumption that we could always retrieve a best match formula (corresponding to the query) from ES Index. This requirement comes from the fact that LDA's co-occurrence works when the query token already appears in the LDA model (the corresponding document(s) used for model generation). After receiving the query, we infer topics of the query for each form using the two LDA models generated. Once topics for the query are inferred, the probability distribution of query's topics is used to compare (using a similarity match) with that of probability distribution of documents in ES, to find the ranked list of documents. For this, we consider top 10 topics of query and these topics are queried into elastic search index to find matched docs. An example of the ES query used to retrieve all query topic matching documents is given in fig. 8.

```
{
  "size": 50,
  "query":
     {    "bool": {"should": [{"match": {
           "topic": {
           "query": "278 333 330 218 243 295 231 113 78 378","type": "phrase","boost":
100.0
         }
       }
     },
     {    "match": {"topic": {
           "query": "278 333 330 218 243 295 231 113 78 378","type":
"boolean","operator": "AND","boost": 10.0
         }
       }
     },
    {"match": {"topic": {
           "query": "278 333 330 218 243 295 231 113 78 378", "type": "boolean",
"operator": "OR", "boost": 1.0
         }
       }
     } ]
   }
  }
}
```

Figure 8: Sample ES query to retrieve documents matching query topics

The retrieved documents from ES are ranked by comparing the topic probability distribution of query and the retrieved documents. We use similarity metric based on *Jensen-Shannon divergence*. The resulting similarity score then becomes the LDA-based document score. The similarity metric is given as

$$Sim_{lda} = 1 - JSD(P(Q^T)||P(D_i^T)) \quad (4)$$

where $P(Q^T)$ is the probability distribution for query topics, $P(D_i^T)$ is the topic probability distribution for i^{th} retrieved document (from ES) and $JSD(P||Q)$ is the Jensen-Shannon divergence between P & Q and is given as

$$JSD(P||Q) = \frac{1}{2}D(P||R) + \frac{1}{2}D(Q||R),\ R = \frac{P+Q}{2} \quad (5)$$

where $D(X||Y)$ is KL-divergence between the two random variables X & Y and is given as $D(X||Y) = \sum_i X(i) log_a \frac{X(i)}{Y(i)}$. Note that in Eqn. 4, JSD is computed for log to the base 2 (i.e., $a = 2$) and hence, $JSD \in [0, 1]$.

NTCIR-12 Query	Elastic Search: Specialized (top retrieved result)	Top Topic's surrounding words for Specialized model	Elastic Search: Generalized (top retrieved result)	Top Topic's surrounding words for Generalized model
$*1*^2 + *2*^2 = *3*^2$	$A^2 + w^2 = b^2$	• $w^2 + B^2 = a^2$ • $(\text{Eq.2}) a^2 - b^2 = (B + A)(B - A)$ • $a^2 - B^2 = b^2 - A^2$ • $B^2 + w^2 = a^2$ • $w^2 + A^2 = b^2$ • $w = \sqrt{b^2 - A^2}$. • $(a^2 - b^2)^2 = (A + B)^2 (A + B)((A + B) - 4h)$ • $(a^2 - b^2)^2 = (B + A)^2 (B^2 - 2AB + A^2)$ • $(a^2 - b^2)^2 = (A + B)^2 ((A^2 + 2AB + B^2) - 4AB)$, • $b^2 - a^2$	$*^\star + *^\star = *^\star$	• triples • $*_\star = \frac{*'}{*''}$. • $*^\star = \frac{*}{***}$ • $[*, *, *]$ • $*^\star * + *^\star * = *$ • $* \times (*^\star + *^\star)$ • newton • capacitance • falls • rays
proof $x^2 + y^2 = z^2$	$x^2 + y^2 = z^2$	• $m^2 - n^2$ • $(m^2 + n^2)(p^2 + q^2) = (mp - nq)^2 + (np + mq)^2 = (mp + nq)^2 + (np - mq)^2$. • $a^2 + b^2 = (m^2 - n^2)^2 + (2mn)^2 = (m^2 + n^2)^2 = c^2$. • $a^2 + b^2 = c^2 + d^2$ • $(a^2 - b^2)^2 + (2ab)^2 = (a^2 + b^2)^2$ • $c^2 - a^2 = b^2$ • triples • $x^2 + (x + 1)^2 + \cdots + (x + q)^2 + p^2 = (p + 1)^2$, • $(2ad)^2 + (2bc)^2 = (a^2 - b^2 - c^2 + d^2)^2$ • $(2ab)^2 + (2cd)^2 = (a^2 + b^2 - c^2 - d^2)^2$	$*^\star + *^\star = *^\star$	• angle • square • theories • height • trigonometric • sum • root • number • function • set

Figure 9: Example analysis of LDA algorithm of two NTCIR-12 queries

Discussion
Figure 9 shows the top topic results for the two NTCIR-12 queries on the formula form for Pythagorean Theorem. As can be seen, for both the queries, Elastic search was able to retrieve correct match for both the specialized as well as generalized case. For the first query, the top topic's surrounding words are related in case of specialized model but partially related in case of generalized model. Similar is the interpretation for second query, except for its generalized case where it actually considered top topic corresponding to word "proof" rather than considering contextual part also: the query formula. These observations are a consequence of below limitations which when resolved would improve the overall performance of LDA-based scoring. These limitations could not be resolved in the current setup due to infrastructure issues and time concerns, as discussed earlier.

1. The number of LDA topics are too low (topics = 500) for the amount of data considered. Once an appropriate topic count is considered (using perplexity score), one would expect "proof" & "$x^2 + y^2 = z^2$" to come under same topic

2. Both specialized as well as generalized forms to be considered under single LDA model. This would increase the co-occurrence of similar structure formulae thereby giving better results for generalized form

2.5 Pattern-based Scoring

We observe in NTCIR-12 dataset that certain NLP-based patterns are frequently repeated and hence, applied pattern-based approach to identify such patterns. The pattern-based approach is similar to [1] but differs in its usage. Of all the patterns, we identified two prominent patterns from dataset based on the techniques discussed later in this section.

In the offline part, we develop NLP rules based on Stanford Dependency and POS (Stanford core NLP v3.5.2 package) parsers for mapping the mathematical equation to its definition. We then index the definition to equation mapping in the elastic search in an equation-definition index ([ES_INDX6]). Finally, [ES_INDX6] index is used for expanding the initial query before the expanded query is given to the other ES indexes ([ES_INDX1 $\cdots$ ES_INDX5) for fetching ranked documents.

2.5.1 *Identified Patterns*

We identified two patterns to identify the mapping of mathematical equation to definition.

1. *Nsubj based pattern*: In this pattern, if a token is associated as a dependent in an nsubj relation and the governor of the nsubj relation (which is an equation) is also the governor of a copula relation, the governor of the nsubj relation is the extracted mathematical equation and the dependent of the nsubj relation is its definition. Definition is filtered using POS tag. It should be a noun phrase like (NN, NNS, NNP, and NNPS). An example is shown in fig. 10.

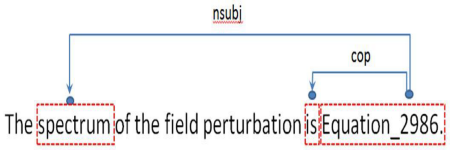

Figure 10: Nsubj based pattern

2. *NSubjPass based pattern*: In this pattern, if a token is associated as a dependent in an nsubjpass relation and the governor of nsubjpass relation is also the governor of an nmod_agent relation whose dependent is an equation then the dependent of the nmod_agent relation is the extracted mathematical equation and the dependent of the nsubjpass relation is its definition. Again the definition is filtered using POS tag .It should be a noun phrase like (NN, NNS, NNP, and NNPS). An example is shown in fig. 11.

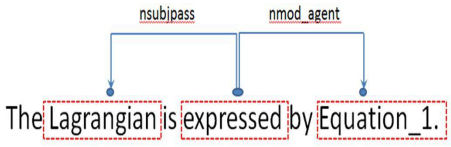

Figure 11: NSubjPass based pattern

2.5.2 *Pattern Indexing and Scoring*

We index the equation and its definition in the elastic search. The both definition as well as equation are stored as an indexable string which is analyzed using a regular expression ("<.*?>|[^<^>]+", acts as a tokenizer to create sub-components from a mathml expression) and filtered using a "lowercase" filter.

Once the expanded query is generated (using the method described earlier), it is used to retrieve ranked list of documents from other ES indexes with default ES scoring. The scored documents then become pattern-based scored documents.

2.6 Document Re-Ranking

Having obtained different document scores each from different scoring mechanisms, we require a method to re-rank the documents to obtain a consolidated ranking of documents. We use Borda count based ranking mechanism (as discussed earlier) to re-rank the documents. Figure 12 depicts the complete re-ranking system. NTCIR requires scoring for both documents as well as formulae. We individually

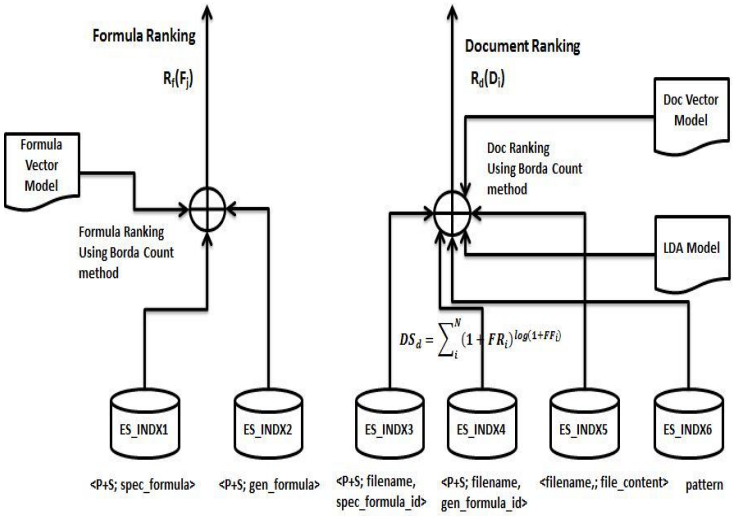

Figure 12: Re-ranking System

describe, in subsequent subsections, the scoring mechanism of each of them.

2.6.1 Document Scoring

The ranking formula used for re-ranking of documents is:

$$\text{Document Ranking} = BC(S_{ES}, S_{D2V}, S_P, S_{LDA}) \quad (6)$$

where $BC(.)$ is the Borda Count Ranking and S_{ES}, S_{D2V}, S_{LDA}, S_P, are scores of Elastic Search, Doc2Vec, LDA and Pattern respectively and are given as

$$S_{ES} = BC(S_K, S_S, S_G) \quad (7)$$

where S_K is the ES score corresponding to Keyword Index, S_S is ES score corresponding to Specialized Index and S_G is ES score corresponding to Generalized Index.

$$S_{D2V} = BC(S_F, S_D) \quad (8)$$

where S_D is Doc2Vec score and S_F is formula vector score.

$$S_P = BC(S_{EK}, S_{ES}, S_{EG}) \quad (9)$$

where S_{EK} is the extended ES score corresponding to Keyword Index for extended query, S_{ES} is ES score corresponding to extended Specialized Index and S_{EG} is ES score corresponding to extended Generalized Index.

$$S_{LDA} = BC(S_{LS}, S_{LG}) \quad (10)$$

where S_{LS} is scoring corresponding to specialized LDA model and S_{LG} is scoring corresponding to generalized LDA model. Depending on the scoring algorithms used, document ranking formula changes (Ex- for ES+D2V, formula becomes Document Ranking $= BC(S_{ES}, S_{D2V})$.

Since ES contains index of formula rather than documents, we converted the scoring of formula to scoring of documents. During ES scoring, we were boosting the score for exact match and hence, we first normalize the score by using: ES_Score/Max_ES_Score and denote it by FR(formula Relevance). For document scoring, we do the following:

$$DS_d = \sum_i^N (1 + FR_i)^{log(1+FF_i)} \quad (11)$$

where DS_d is the document score for document d, FR_i (> 0 and ≤ 1) is the formula relevance score (normalized ES score) for i^{th} matched result out of r formula retrieved results and FF_i is the frequency of that formula in file d. The choice of formula for formula-to-document scoring is based on the following requirement:

1. High FR and high FF => Highest rank for that document
2. High FR and low FF => Medium-to-low rank for that document
3. Low FR and high FF => Low-to-medium rank for that document
4. Low FR and low FF => Lowest rank for that document

2.6.2 Formula Scoring

Formula scoring is obtained by applying Borda Count-based Ranking over Doc2Vec (for formula) and ES scores as follows:

$$\text{Formula Ranking} = BC(S_{ES}, S_{F2V}) \quad (12)$$

3. RESULTS & EVALUATION

3.1 System Setup

We use a basic set up of three normal desktop PCs with one PC handling LDA and ES internal queries, another for doc2vec internal queries and the third one for re-ranking, result generation & also, acts as a spring framework server for the queries. Because of this minimal infrastructure, the response time for all the NTCIR-12 queries both for ArXiv as well as Wikipedia dataset is high. We expect it to reduce to acceptable figures once it is deployed in a distributed environment.

3.2 Results Analysis

As discussed earlier, our ArXiv submission was partial due to time and infrastructure constraints. We submitted top 1000 results from the output of only Elastic Search index for all the 50 queries. As can be seen from fig. 13 for the best comparison results, the performance of ES alone is fairly good, as far as partially relevant results are concerned. In contrast to other systems, the performance of ES does not degrades much as the precision@N increases.

For Wikipedia task, we submitted four runs each with top 1000 results: (i) ES only (ii) ES and Doc2Vec(D2V) (iii) ES, D2V and Pattern(P) and (iv) ES+D2V+P+LDA. Figure 14 shows the comparison for all the four variants. We observe that ES+D2V+P performs best followed by ES+D2V. It clearly shows that ES alone cannot assist in retrieving relevant results though it does help in increasing the number of partial results (fig. 15), an observation similar to ArXiv dataset. Additional ranking systems do assist in improving the overall rank of the relevant documents.

One of the reasons why variant with LDA did not perform as good as other variants is because of the choice of the number of topics. Because of infrastructure limitations, we use 500 topics which is way too less as compared to the number of topics usually used (2000-10000 topics or more depending on the vocabulary size) to give better granularity to topics. Additionally, some parameter tweaking is required

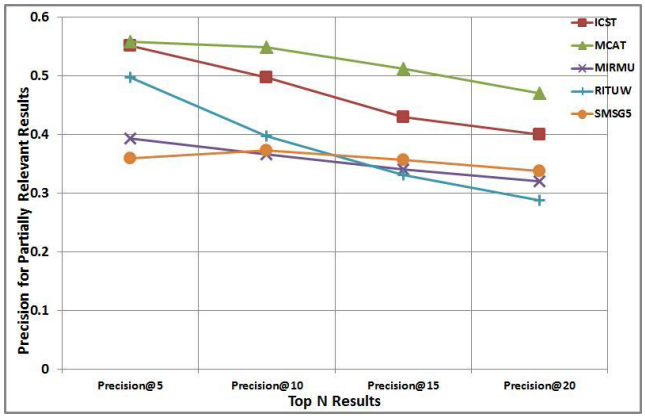

Figure 13: Comparison among the 5 systems for their best results for partial relevance using treceval

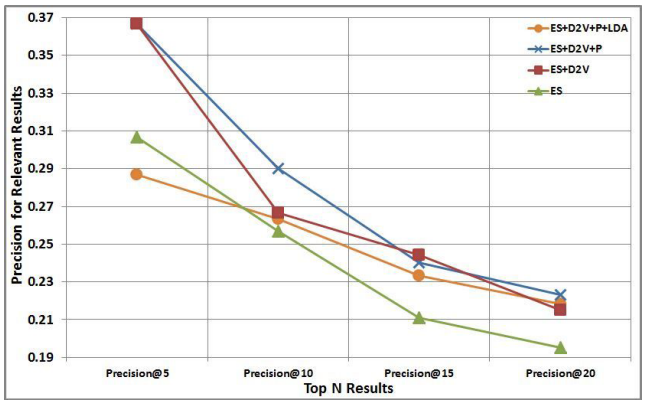

Figure 14: Comparison among the four variants of our Math search engine for relevant results using treceval

to achieve best performance. For our case, we considered the LDA parameters as: $\alpha = 0.1$, $\beta = 0.1$, $chunksize = 5000$, $passes = 5$ under Gensim tool. It is also observed that for some queries, ES's top most result was partially relevant which indicates some improvements are required in ES's best result retrieval mechanism.

The observation somewhat changes in case of partially relevant results. As can be seen from fig. 15, ES performs way much better as compared to ES+D2V+P. It indicates that in general, ES performs well when it comes to retrieving partially relevant results and requires other systems to improve its relevant result score. One of the reasons why we think ES works well for partial relevance is in its capability of partial matches for formulae and tf-idf based keyword match scoring.

4. CONCLUSION & FUTURE WORK

In this paper, we present a Math Search engine by considering the co-occurrence of formula and keywords. We use Borda count-based ranking system to re-rank the documents.

As part of the future work, we plan to work on the following:

1. Introduce normalization: apply operator and/or variable ordering to improve results
2. Improve the infrastructure to get better query response time, consider higher LDA topics and improve doc2vec models
3. Consider Content MathML instead of Presentation MathML as it has been shown that in general, Content MathML performs better
4. Our choice of (nested) Borda count-based re-ranking is based on its simplicity and faster re-ranking capability. Though it works well, weights across different scoring mechanisms are treated equally. That is, the different knowledge bases are considered equally important which may not be the case always. For example, in some queries, it is found the keywords are not that relevant whereas in others, they help in retrieving the documents. Hence, a machine learning approach like RankSVM would help in further improving the systems performance.

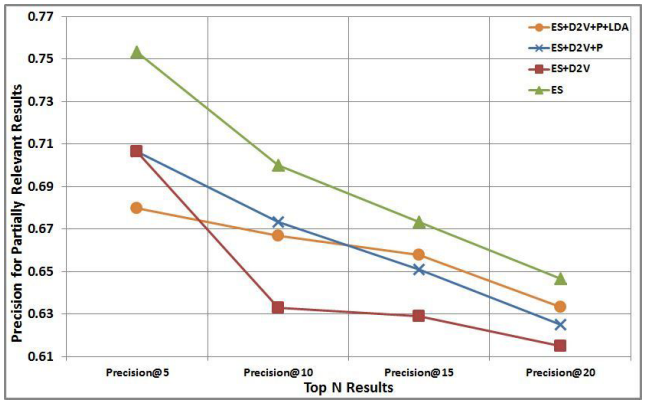

Figure 15: Comparison among the four variants of our Math search engine partially relevant results using treceval

5. REFERENCES

[1] A. Aizawa and et al. Extracting textual descriptions of mathematical expressions. In *Proceedings of Scientific Papers D-Lib Magazine*, 2014.

[2] J. A. Aslam and M. Montague. Models for metasearch. In *24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 276–284, 2001.

[3] V. der Maaten, Laurens, and G. Hinton. Visualizing data using t-sne. In *Journal of Machine Learning Research 9.2579-2605 (2008): 85*, 2008.

[4] Le, Q. V., and T. Mikolov. Distributed representations of sentences and documents. In *arXiv preprint arXiv:1405.4053*, 2014.

[5] M. Tomas and et al. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 2013.

[6] R. Zanibbi, A. Aizawa, M. Kohlhase, I. Ounis, G. Topić, and K. Davila. NTCIR-12 mathir task overview. In *NTCIR*. National Institute of Informatics (NII), 2016.