Three is Better than One
Ensembling Math Information Retrieval Systems

Vít Novotný, Petr Sojka, Michal Štefánik, and Dávid Lupták

Math Information Retrieval Research Group
https://mir.fi.muni.cz/
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
sojka@fi.muni.cz, {witiko,stefanik.m,dluptak}@mail.muni.cz

Abstract. We report on the systems that the Math Information Retrieval group at Masaryk University (MIRMU) prepared for tasks 1 (find answers) and 2 (formula search) of the ARQMath lab at the CLEF conference. We prototyped three primary MIR systems, proposed several math representations to tackle the lab tasks, and evaluated the proposed systems and representations. We developed a novel algorithm for ensembling information retrieval systems that outperformed all our systems on task 1 and placed ninth out of the 23 competing submissions. Out-of-competition ensembles of all non-baseline primary submissions in the competition made available by the participants placed first on task 1 and third on task 2. Our prototypes will help to understand the challenging problems of answer and formula retrieval in the STEM domain and bring the possibility of accurate math information retrieval one step closer.

Keywords: Math information retrieval · question answering · math representations · word embeddings · ensembling

"I do not demand that you make me happy; my happiness does not lie in you." Anthony de Mello

1 Introduction

The Math Information Retrieval (MIR) group at Masaryk University (MIRMU) is interested in challenges of MIR already for more than a decade. The challenges are numerous:

1. How to represent and index the meaning of formulae?
2. How to collate the representation of text and formulae together to grab and disambiguate the meaning of symbols in the formulae?

3. How to pick the canonical representation of formulae in context for formulae retrieval?
4. How to integrate the inference into the information retrieval or question answering system?

In the ARQMath competition, we tackled both task 1 (find answers) and task 2 (formula search). We researched and evaluated several ways of tackling the challenges [41] to push the boundaries of knowledge in MIR.

We prototyped and submitted results for three primary systems, MIaS, SCM, and COMPUBERT, and a secondary system, Formula2Vec.[1] Our main objective was to gain insight into the problems above and we submitted five result lists for both tasks. We think that the key in solving the challenges lies in the accurate representation of meaning of texts and formulae in context—thus we opted to prototype systems that primarily tackle the math representation problem.

We preprocessed the data and represented both given data and other STEM corpora as arXiv to learn the joint representations of text and math. We have compared several math formulae representations and used unsupervised approaches for representation learning. Finally, we ensembled our primary submissions into a committee of MIR systems that was able to achieve better result on task 1 than any of the ensembled method alone.

In this report we report in detail about our achievements: Section 2 describes in detail the resources we used and our math representations. Section 3 reports about using our venerable old MIaS system for ARQMath tasks. In sections 4 and 5, systems based on the embeddings of shallow multilayer perceptrons are evaluated. To push the envelope and catch up with the recent trends in the NLP, we experiment with using the Transformer architecture to produce math representations with our COMPUBERT system in Section 6. The description of our ensembling algorithm can be found in Section 7. Our results, insights we got, and future directions are thoroughly discussed in sections 8 and 9.

> "Don't ask the world to change... you change first."
> Anthony de Mello

2 Methods

In this section, we will describe the math representations ingested by our information retrieval systems, the corpora used for training the models that power our systems, and the relevance judgements we used for parameter optimization, model selection, and performance estimation.

2.1 Math Representations

Our math information retrieval systems ingest math formulae in a variety of formats: LaTeX, Presentation MathML, Content MathML, Symbol Layout Tree

[1] Formula2Vec is considered secondary, because it is a direct application of the Doc2Vec DBOW model [15] without any adaptation to the tasks of the ARQMath 2020 competition. It serves as a baseline for the architecturally-similar primary system of the SCM.

(SLT), Operator Tree (OPT), prefix, and infix. Figure 1 shows how the individual math representations are derived from LaTeX.

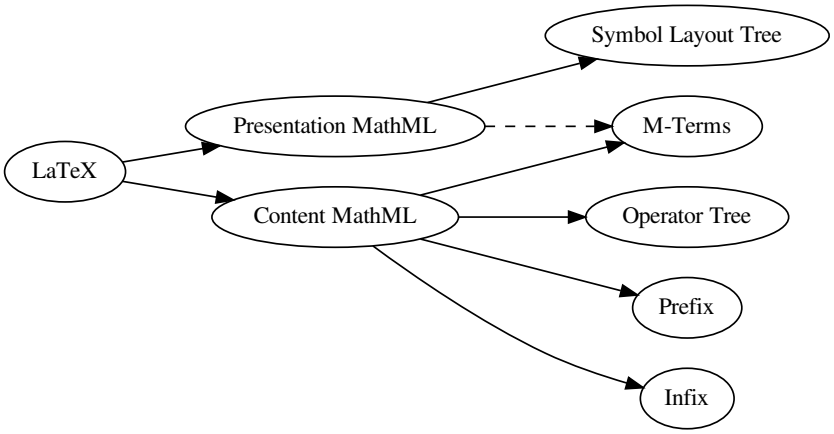

Fig. 1. Dependencies between math representations ingested by our systems.

LaTeX As the most direct math representation, we used LaTeX, which is the standard authoring format for math. Although LaTeX is easy to type, it encodes mostly just the presentation aspects of a math formula, not its content. LaTeX is also a Turing-complete language and therefore impossible to statically parse in the general case. As a result, each formula is represented as a single token in the LaTeX representation.

LaTeX is useful as a baseline math representation and as a basis for deriving more fine-grained math representations described in the following sections. Although having each formula represented as a single token may not seem useful, two of our four systems (the SCM and COMPUBERT) model subwords, which allows them to extract symbols out of the formulae.

To give an example, the formula $x!! - y^2 = 0$ would be represented as a single token $x!! - y^2 = 0$ in LaTeX.

MathML Using LaTeXML v0.8.4[2] [43] and MathML Canonicalizer,[3] [8] we converted math formulae from LaTeX to MathML 3.0, [1] which contains two markups: Presentation MathML (PMML) and Content MathML (CMML).

Like LaTeX, PMML encodes mostly just the presentation aspects of a math formula. Unlike LaTeX, PMML is tokenized to individual math symbols. Unlike LaTeX and like PMML, CMML is tokenized. Unlike PMML, CMML is independent

[2] `https://dlmf.nist.gov/LaTeXML`

[3] `https://github.com/MIR-MU/MathMLCan`

```
<mrow>
  <mi>x</mi>
  <mo>!!</mo>
  <msup>
    <mi>y</mi>
    <mn>2</mn>
  </msup>
  <mo>=</mo>
  <mn>0</mn>
</mrow>
```

```
<apply>
  <eq/>
  <apply>
    <minus/>
    <apply>
      <csymbol cd="latexml">double-factorial</csymbol>
      <ci>x</ci>
    </apply>
    <apply>
      <csymbol cd="ambiguous">superscript</csymbol>
      <ci>y</ci>
      <cn type="integer">2</cn>
    </apply>
  </apply>
  <cn type="integer">0</cn>
</apply>
```

Fig. 2. The PMML (left) and CMML (right) representations of the math formula $x!! - y^2 = 0$. Notice how the double factorial operator is recognized and how the numerical constants (<cn>) are differentiated from variable names (<ci>) in CMML.

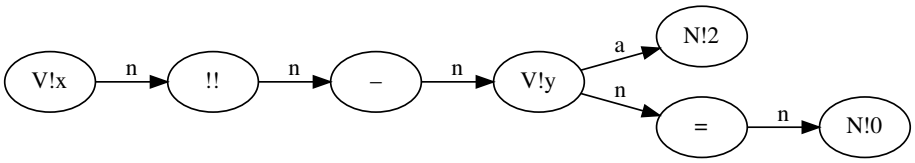

Fig. 3. The SLT representation of the math formula $x!! - y^2 = 0$, where V! prefixes variable names, N! prefixes numerical constants, n stands for next, and a stands for above.

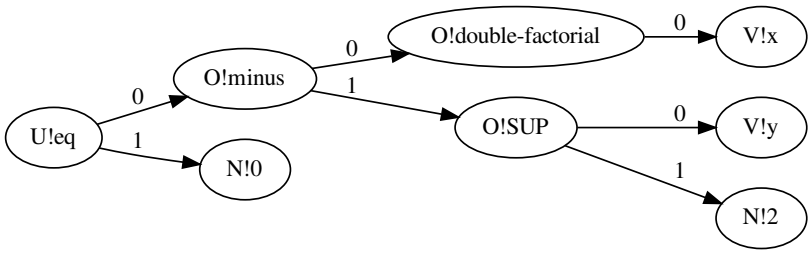

Fig. 4. The OPT representation of the math formula $x!! - y^2 = 0$, where U! prefixes commutative operators, O! prefixes non-commutative operators, V! prefixes variable names, N! prefixes numerical constants, 0 stands for the first operand, and 1 stands for the second operand.

of the presentation aspects of a formula and encodes visually distinct but semantically equivalent formulae the same.

To give an example, the formula $x!! - y^2 = 0$ would be represented as the PMML and CMML documents in Figure 2.

M-Terms Using MIaSMath,[4] we converted math formulae from MathML to mathematical tokens (M-Terms) by ordering, tokenization, and unification. [42] Although M-Terms can be extracted from both PMML and CMML, the CMML representation has produced the best results in previous MIR competitions. [30]

To give an example, the formula $x!! - y^2 = 0$ would be represented as the following comma-separated bag of tokens in MathML: $x!! - y^2 = 0$, $x!! - y^2$, 0, $=$, $x!!$, y^2, $-$, x, $!!$, y, 2, $\mathrm{id}_1!! - \mathrm{id}_2{}^2 = 0$, $\mathrm{id}_1!! - \mathrm{id}_2{}^2$, $\mathrm{id}_1!!$, $\mathrm{id}_2{}^2$, id_1, id_2, $x!! - y^{\mathrm{const}} = \mathrm{const}$, $x!! - y^{\mathrm{const}}$, y^{const}, $\mathrm{id}_1!! - \mathrm{id}_2{}^{\mathrm{const}} = \mathrm{const}$, $\mathrm{id}_1!! - \mathrm{id}_2{}^{\mathrm{const}}$, $\mathrm{id}_2{}^{\mathrm{const}}$.

Symbol Layout Tree and Operator Tree Using our fork of Tangent-CFT[5] [19] that recognizes all math operators in our corpora (see Section 2.2), we converted math formulae from PMML and CMML to the Symbol Layout Tree (SLT) and Operator Tree (OPT) math representations, respectively.

SLT corresponds to PMML with positional relations between math symbols. In our systems, we tokenize SLT into paths in depth-first-search order. A path in the SLT is represented as a 4-tuple (initial node of the path, terminal node of the path, spatial relations between symbols on the path, spatial relations between symbols on the path from the SLT root to the initial node).

OPT corresponds to CMML with additional information about the types of math symbols. In our systems, we tokenize OPT into paths in depth-first-search order. A path in the OPT is represented as a 4-tuple (initial node of the path, terminal node of the path, edge numbers on the path, edge numbers on the path from the SLT root to the initial node).

To give an example, the formula $x!! - y^2 = 0$ would be represented as the SLT tree in Figure 3 and the OPT tree in Figure 4.

Prefix and Infix To linearize the OPT, we converted math formulae from OPT into the prefix and infix notations. The prefix notation corresponds to the list of visited nodes in the OPT in depth-first-search order, i.e. the topological sorting of the OPT. By parenthesizing operands and by recognizing infix operators, we also produce the infix notation.

Like LaTeX, the prefix and infix notations are easy to type. Unlike LaTeX, the prefix and infix notations are tokenized into math symbols and independent on the presentation aspects of a formula. In TF-IDF-based systems, the prefix and infix notations are virtually equivalent.

To give an example, the formula $x!! - y^2 = 0$ would be represented as the following space-separated list of tokens in the prefix notation: U!eq O!minus

[4] `https://github.com/MIR-MU/MIaSMath`

[5] `https://github.com/MIR-MU/TangentCFT`

O!double-factorial V!x O!SUP V!y N!2 N!0 and as the following space-separated list of tokens in the infix notation: ((O!double-factorial (V!x) O!minus O!SUP (V!y , N!2)) U!eq N!0).

2.2 Corpora

For training our models, we used the arXMLiv and Math StackExchange corpus. Our data preprocessing code is available online.[6]

ArXMLiv The arXMLiv 08.2019 corpus [9] contains 1,374,539 articles from the arXiv.org open-access archive converted from LaTeX to HTML5 and MathML. The corpus is split into four subsets: no_problem (150,701 articles), warning_1 (500,000 articles), warning_2 (328,127 articles), and error (395,711 articles), according to the severity of errors encountered when converting LaTeX to HTML5. For training our models, we only used the no_problem, warning_1, and warning_2 subsets of the corpus.

Math StackExchange The Math StackExchange corpus V1.2 provided by the organizers of the ARQMath 2020 competition contains 2,466,080 posts (questions and answers) from the Math StackExchange question answering website in HTML5 and LaTeX. In our conversion from LaTeX to MathML, we successfully converted 28,320,920 math formulae from LaTeX to 26,705,527 math formulae in CMML (94.3% success rate) and to 27,232,230 math formulae in PMML (96.16% success rate).

Posts in the Math StackExchange corpus are layered and contain not only the body text, but also the title, tags, comments, up- and downvotes, view count, authorship information, etc. Each of our systems used different parts of the corpus, which will be specified in detail in the corresponding sections.

2.3 Competition Tasks

In the ARQMath 2020 competition, we competed in tasks 1 and 2.

Task 1: Find Answers Given a posted question as a query, task 1 is to search all answer posts in the Math StackExchange corpus and return relevant answers. There exist 1.4 million answers in Math StackExchange.

For task 1, we submitted three primary submissions, which correspond to the three primary systems that we developed (MIaS, SCM, and COMPUBERT), and two alternative submissions, which correspond to the Formula2Vec secondary system and an ensemble of the three primary submissions.

[6] `https://github.com/MIR-MU/ARQMath-data-preprocessing`

Task 2: Formula Search Given a formula from a question as a query, search all question and answer posts in the Math StackExchange corpus for relevant formulae. There exist 28.3 million math formulae in Math StackExchange.

For task 2, we submitted two primary submissions, which corresponded to the two systems designed with task 2 in mind (SCM and Formula2Vec), and three alternative submissions, which correspond to different configurations of SCM and Formula2Vec, and an ensemble of the two primary submissions.

Both SCM and Formula2Vec were straightforwardly adapted from task 1 to task 2 by changing the indexing unit from an answer to a formula. Neither search engine takes the textual context of the indexed formulae into account.

2.4 Relevance Judgements

In our experiments, we used two sets of relevance judgements, automatic for parameter estimation and model selection, and human-annotated for performance estimation.

Automatic Since there were no relevance judgements for task 1 available during the development of our systems, we produced our own from the Math StackExchange corpus.[7] The judgements served as a proxy for task 1 during parameter estimation and model selection in our systems.

Each question in the Math StackExchange corpus is a topic, and its judged documents are all its answers with max(0, the number of upvotes minus the number of downvotes) as the gain. Answers to different questions are not judged.

Out of all 797,652 questions in the Math StackExchange corpus with answers, we used 628,442 (78.79%) for training, 85,840 (10.76%) for parameter optimization, and 83,370 (10.45%) for model selection, i.e. for deciding which systems will get submitted to the competition.

Human-Annotated After the submission of our systems, official human-annotated task 1 and 2 relevance judgements produced by eight annotators with fair agreement ($\kappa = 0.34$) [13] were released.[8] [47, Section 4.4] We used the relevance judgements for performance estimation of system configurations that we did not submit to the competition.

Task 1 For each of 77 questions excluded from the Math StackExchange corpus, the judged documents are on average 460 answers from Math StackExchange that were evaluated by human annotators with a range from 0 (not relevant) to 3 (highly relevant) as the gain.

The relevance judgements are highly imbalanced in favor of non-relevant answers: Out of all 39,124 judgements, there exist as many as 35,051 judgements (89.59%) with gain 0, 2,269 judgements (5.8%) with gain 1, 1,071 judgements

[7] `https://github.com/MIR-MU/ARQMath-eval`, files `votes-qrels-*.V1.0.tsv`

[8] `https://github.com/MIR-MU/ARQMath-eval`, files `qrel_task{1,2}.tsv`

(2.74%) with gain 2, and only 733 (1.85%) judgements with gain 3. The small number of relevant answers impedes the usefulness of the human-annotated relevance judgements for training supervised models for use in future systems.

Out of all 39,124 judgements, we used 31,541 (80.62%) for training out-of-competition system configurations, 3,560 (9.10%) for parameter optimization, and 4,023 (10.28%) for performance estimation.[9]

Task 2 For each of 45 math formulae from questions excluded from the Math StackExchange corpus, the judged documents are on average 256 formulae from questions and answers in Math StackExchange that were evaluated by human annotators with a range from 0 (not relevant) to 3 (highly relevant) as the gain.

The relevance judgements are slightly imbalanced in favor of non-relevant formulae: Out of all 12,116 judgements, there exist 7,891 judgements (65.13%) with gain 0, 718 judgements with gain 1 (5.93%), 553 judgements (4.56%) with gain 2, and 2,954 judgements (24.38%) with gain 3.

2.5 Evaluation Measures

To measure information retrieval accuracy for parameter optimization, model selection, and performance estimation, we used the Normalized discounted cumulative gain prime (nDCG') and Spearman's ρ.

Normalized Discounted Cumulative Gain Prime The nDCG' [32] is an evaluation measure specifically designed for information retrieval with incomplete relevance judgements. nDCG' is defined as follows:

$$\text{nDCG}' = \operatorname*{avg}_{t \in T} \frac{\text{DCG}'_t}{\text{IDCG}_t},\ \text{IDCG} = \sum_{i=1}^{|\text{REL}_t|} \frac{\text{gain}_t(\text{REL}_{t,i})}{\log_2(i+1)},\ \text{DCG}' = \sum_{i=1}^{|\text{RES}'_t|} \frac{\text{gain}_t(\text{RES}'_{t,i})}{\log_2(i+1)}, \quad (1)$$

where T are the topics for a task, REL_t is a list of relevant documents for topic t in the descending order of their gain up to position 1,000, RES_t is a list of results produced for topic t our system up to position 1,000, $\text{RES}'_t = \text{REL}_t \cap \text{RES}_t$, and $\text{gain}_t(R)$ is the gain of result R for topic t as specified by relevance judgements.

To see if two nDCG' values are significantly different, we construct two 95% Student's t [44] confidence intervals from the values of $\left(\text{DCG}'_t/\text{IDCG}_t\right)_{t \in T}$. If the intervals do not overlap, we say that the higher nDCG' value is *significantly better*.

Spearman's ρ Spearman's ρ is a general non-parametric measure of rank correlation. Spearman's ρ between random variables X and Y corresponds to Pearson's r between the rank values rg_X and rg_Y of the random variables:

$$\rho = \frac{\text{cov}(\text{rg}_X, \text{rg}_Y)}{\sigma_{\text{rg}_X} \cdot \sigma_{\text{rg}_Y}} \text{ and } \text{cov}(\text{rg}_X, \text{rg}_Y) = \text{E}\big[(\text{rg}_X - \text{E}[\text{rg}_X]) \cdot (\text{rg}_Y - \text{E}[\text{rg}_Y])\big]. \quad (2)$$

[9] `https://github.com/MIR-MU/ARQMath-eval`, files `qrel_task1-*.tsv`

Unlike nDCG', Spearman's ρ can not handle incomplete relevance judgements. Therefore, we only used Spearman's ρ to evaluate our systems with automatic relevance judgements, which are always complete for all answers of a question.

Throughout the paper, we refer to both nDCG' and Spearman's ρ as measurements of our systems' *accuracy*. This should not be confused with the binary classification accuracy measure, which is computed as the proportion of correct predictions to all predictions and which we do not use.

> "When you get rid of your fear of failure, your tensions about succeeding... you can be yourself. Relaxed. You'll no longer be driving with your brakes on." Anthony de Mello

3 Math Indexer and Searcher

In 2011, we have developed and open-sourced the Math Indexer and Searcher (MIaS) system [40] based on Apache Lucene: [2] a high-performance full-text search engine library. We have deployed MIaS in the European Digital Mathematical Library[10], making it historically the first MIR system deployed in a digital mathematical library. [46] The architecture of MIaS is shown in Figure 5.

In this section, we will describe MIaS and its results on task 1 of the ARQMath 2020 competition. Our experimental code is available online.[11]

[10] `https://eudml.org/search`

[11] `https://github.com/MIR-MU/SCM-at-ARQMath`, file `MIaS-at-ARQMath.ipynb`

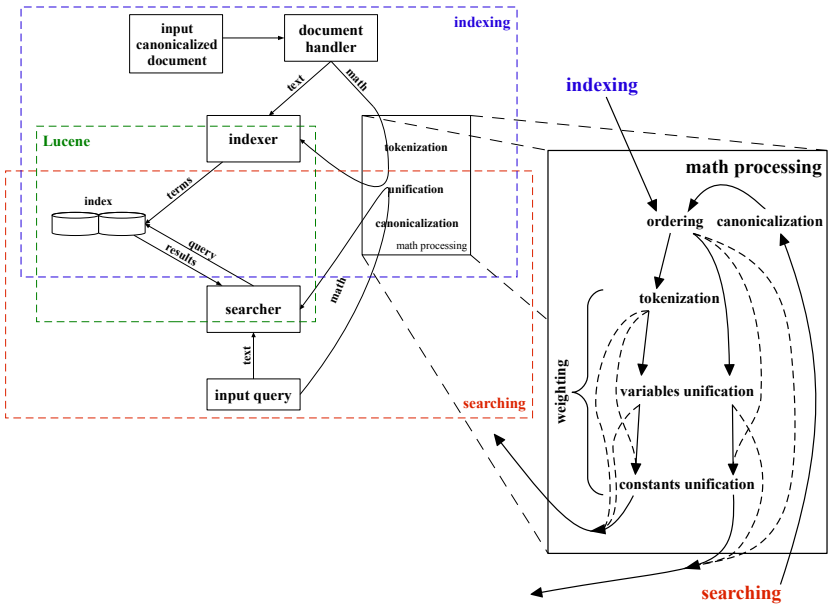

Fig. 5. Top-level architecture of our Math Indexer and Searcher (MIaS) system.

3.1 Configuration

We represented answers from the Math StackExchange corpus as the text and math in their body text and comments, as produced by the `HtmlGenerator.generate_answer` method of the code provided by the ARQMath 2020 organizers.[12] We represented math in answers using M-Terms extracted from CMML with structural unification disabled, since experimental results have shown that the it is not suitable for general use. [28, 31]

Since the TF-IDF scoring function of MIaS [40, Section 3.1] is best suited for short and precise queries due to a lack of logarithmic TF factoring, [38, Section 2] we represented topics as the text and math in their title, ignoring the body text and tags. Like in the indexing of answers, we represented math in topics using M-Terms extracted from CMML. To improve the recall of MIaS, we use the Leave Rightmost Out (LRO) querying strategy [30, 16, 28] that expands the original query and produces a set of subqueries by leaving out text keywords and math subformulae. After expansion, we submit the subqueries to MIaS and interleave the partial results into a final result list.

We submitted the described configuration of MIaS as a primary submission.

3.2 Results

In this section, we will discuss the accuracy of the MIaS using nDCG′ on the official human-annotated relevance judgements for task 1, and the speed of MIaS.

Accuracy In the described configuration, the MIaS system achieved the nDCG′ 0.155 on task 1, which is far from the best competing system, but close to the Tangent-S system that served as a baseline. [47, Table A1]

Speed For task 1, the average run time of MIaS over all topics was 1.24 seconds, with the minimum of 0.1 seconds for topic A.55, and the maximum of 7.27 seconds for topic A.80.

MIaS ran on a machine with 2 TiB free disk space, eight Intel Xeon™ X7560 2.26 GHz processors with a total of 32 CPU cores, 252 GiB RAM and no GPUs.

"You have to understand, my dears, that the shortest distance between truth and a human being is a story." Anthony de Mello

4 Soft Cosine Measure

Since the seminal work of Mikolov et al., [21] unsupervised word embeddings have become the preferred word representations for many natural language processing tasks. Document similarity measures extracted from TF-IDF and unsupervised word embeddings, such as the Soft Cosine Measure (SCM), [36] are

[12] `https://github.com/ARQMath/ARQMathCode`, file `generate_html_file.py`

fast [22] and achieve strong performance on semantic text similarity, [6] information retrieval, [10] entrance exam question answering, [36] and text classification [23] tasks, among others.

In this section, we will describe the Soft Cosine Measure (SCM) system, which combines TF-IDF with unsupervised word embeddings to produce interpretable representations of math documents and math formulae that enable fast and interpretable information retrieval. We will also report the results of the SCM on tasks 1 and 2 of the ARQMath 2020 competition. Our experimental code is available online.[13]

4.1 Orthogonalized Joint Word Embeddings

In mathematical discourse, math formulae are often more important than words for understanding. [14] Using a single representation for both text and math makes it possible to exploit zettabytes of text for training high-quality unsupervised word embeddings that can capture the relations between text with math.

Drawing inspiration from the existing work about unsupervised word embeddings for math, [12, 19] we produce unsupervised joint word embeddings for text and math tokens by training a fastText skipgram model [3] on text and math in a corpus. For the Math StackExchange corpus, we use the body text of posts. For the arXMLiv corpus, we use the full texts. We experiment with different representations of math: LaTeX, SLT, OPT, prefix, and infix.

Unlike the embeddings produced by Krstowski and Blei [12] and Mansouri et al., [19] our fastText embeddings are share weights between tokens that have subwords in common. This speeds up training convergence [3] and enables inference of embeddings for text and math tokens not seen during training. To differentiate the token type (text or math) of subwords, we use a crude heuristic that lower-cases text tokens and upper-cases math tokens.

Following the work of Novotný et al., [23, Section 4.2] we orthogonalize the word embeddings in order to improve retrieval accuracy and speed.

4.2 Document Similarity Measure

The TF-IDF vector space model (VSM) [33] is a distributional semantics model that is fundamental to information retrieval. The VSM represents documents as coordinate vectors relative to an orthogonal basis, where the similarity between documents or formulae in tasks 1 or 2, respectively, is the cosine similarity between their coordinate vectors.

Although the VSM is fast and interpretable, it is highly susceptible to polysemy, since distinct text and math terms correspond to mutually orthogonal basis vectors. Therefore, documents that use different terminology will always be regarded as dissimilar.

To better model the polysemy, we use the TF-IDF soft vector space model (soft VSM). [36, 22] Unlike the VSM, the soft VSM assumes that documents are

[13] `https://github.com/MIR-MU/SCM-at-ARQMath`, file `json_to_fasttext_and_scm.py`

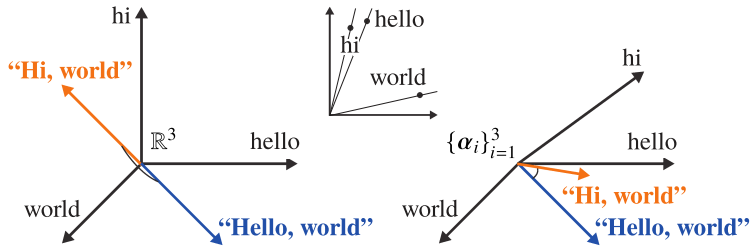

Fig. 6. The representation of two documents, "Hi, world" and "Hello, world" in the TF-IDF vector space model (VSM, left) and in the TF-IDF soft vector space model (soft VSM, right). In the VSM, different terms correspond to orthogonal axes, making the document representations distant despite their semantic equivalence. In the soft VSM, different terms correspond to non-orthogonal axes, where the angle between the axes is proportional to the similarity of terms in a word embedding space (middle).

represented in a non-orthogonal basis, i.e. similar text and math terms correspond to basis vectors that make acute angles, see Figure 6.

Following the work of Novotný et al., [23, Section 3.2] we use the orthogonalized joint word embeddings as the source of similarity between terms. We use the implementation of fastText and the soft VSM in Gensim.[14] [25]

For task 1, we represent answers from the Math StackExchange corpus as the text and math in their body text. For math, we experiment with different representations: LaTeX, SLT, OPT, prefix, and infix. For task 2, we represent formulae from questions and answers in the Math StackExchange corpus as the math in the optimal representation selected for task 1. For both tasks 1 and 2, we represent topics as the text and math in their title, body text, and tags.

4.3 Parameter Optimization

To select the optimal parameters for the SCM on task 1, we used the nDCG' on our relevance judgements (see sections 2.4 and 2.5) as the objective. Since we lacked relevance judgements for task 2, we used the optimal model for task 1 in both tasks.

Initialization To produce the word embeddings, we reproduced the experimental setup of Bojanowski et al.: [3, Section 4] hash table bucket size $2 \cdot 10^6$, 300 vector dimensions, negative sampling loss with five negative samples, initial learning rate 0.05 with a linear decay to zero, sampling threshold 10^{-4}, a window size of five, a minimum of five occurences of a term, and character n-grams with n in $\{3, \ldots, 6\}$. We did not model phrases[15] [20, Section 4] and

[14] `https://radimrehurek.com/gensim/models/fasttext.html` and `https://radimrehurek.com/gensim/similarities/termsim.html`

[15] `https://radimrehurek.com/gensim/models/phrases.html`

trained on the text and the no_problem subset of arXMLiv in the prefix notation for five epochs.

To produce the orthogonalized word embeddings and the term similarity matrix, we used the optimal parameters suggested by Novotný et al.: [23, Table 2] symmetric (Sym $=$ ✓) and strictly diagonally dominant (Dom $=$ ✓) term similarity matrix with a maximum of $C = 100$ non-zero elements in a row/column, and term similarity with threshold $t = -1$ and exponent $o = 4$.

For the TF-IDF soft VSM, we used the dtn.dtn SMART weighting scheme.

Tree Search To optimize the parameters, we performed tree search: assuming the parameters were mutually independent, we varied only one of the parameters at each step of the search, and we selected the best value of the parameter using the nDCG' on our relevance judgements. In each step, we used the optimal values for parameters from previous steps and the default values (see Section 4.3) for parameters from future steps. For each step, we report the best parameter value together with the corresponding nDCG' score *italic*. None of the differences in nDCG' were statistically significant.

In the first step, we selected the optimal representation of math from removing math tokens (0.7600), LaTeX (0.7602), OPT (0.7606), SLT (0.7607), and *prefix/infix (0.7612)*. We were delighted to learn that whereas removing math tokens and just using LaTeX representation produced the worst results, the novel prefix and infix notations led to improvements in the accuracy.

In the second step, we selected the optimal maximum length of phrases from one (0.7612), two (0.7613), *three (0.7614)*, four (0.7612), five (0.7610), and concatenating all consequent math tokens into a single math token (0.7603). The optimal value of three reproduces the result of Mikolov et al. [20, Section 4] who suggest using 2–4 iterations of bigram merging when modeling phrases. Notice also the similar accuracy received when concatenating all math tokens in prefix/infix notation (0.7603) and when using LaTeX (0.7602).

In the third step, we optimized the hash bucket size from $1 \cdot 10^6$ (0.7612), $2 \cdot 10^6$ *(0.7614)*, $4 \cdot 10^6$ (0.7612), $8 \cdot 10^6$ (0.7611). Despite our worries, the default bucket size of $2 \cdot 10^6$ appears to be sufficient to store character n-grams for both text and math tokens.

In the fourth step, we performed an exhaustive grid search of the Sym $\in \{✓, ✗\}$, Dom $\in \{✓, ✗\}$, and $C \in \{0, 50, 100, 200, 400, 800, 1600\}$ parameters, see Table 1. To break the tie, we selected the default parameters: *Sym* $=$ ✔, *Dom* $=$ ✔, *and* $C = 100$ *(0.7614)*.

4.4 Evaluation

We trained the tiny, small, medium, and large models, and used them to produce the competition submissions.

Tiny model For the task 2 alternative submission, we used the model produced via parameter selection.

Table 1. The results of a grid search of the Sym $\in \{\checkmark, \times\}$, Dom $\in \{\checkmark, \times\}$, and $C \in \{0, 50, 100, 200, 400, 800, 1600\}$ parameters using the nDCG′ on our relevance judgements as the objective. Optimal parameter values are **bold**. However, only the values that are *italic* were available before the competition due to time constraints.

Sym	Dom	C	nDCG′	Sym	Dom	C	nDCG′	Sym	Dom	C	nDCG′	Sym	Dom	C	nDCG′
✓	✓	0	0.7613	✗	✓	0	0.7613	✓	✗	0	0.7613	✗	✗	0	0.7613
✓	✓	*50*	*0.7614*	✗	✓	50	0.7613	✓	✗	50	0.7613	✗	✗	50	0.7612
✓	✓	*100*	*0.7614*	✗	✓	100	0.7613	✓	✗	100	0.7610	✗	✗	100	0.7610
✓	✓	*200*	*0.7614*	✗	✓	200	0.7613	✓	✗	200	0.7611	✗	✗	200	0.7610
✓	✓	400	0.7612	✗	✓	400	0.7613	✓	✗	400	0.7613	✗	✗	400	0.7609
✓	✓	*800*	*0.7614*	✗	✓	800	0.7613	✓	✗	800	0.7609	✗	✗	800	0.7609
✓	✓	1600	0.7612	✗	✓	1600	0.7613	✔	✘	**1600**	**0.8104**	✗	✗	1600	0.7609

Small model For the task 1 and 2 primary submissions, we trained fastText for ten epochs on both the Math StackExchange corpus and the no_problem subset of arXMLiv, and we used character n-grams with n in $\{4, 5\}$ shown to be optimal for English by Bojanowski et al. [3, Table 4] We also used the dtb.nnn SMART weighting scheme with slope $s = 0.2$ for the TF-IDF soft VSM instead of dtn.dtn, as suggested by Singhal, [39, Table 1] and we replaced cosine normalization in the soft VSM document similarity function with a normalization function that maintains the ℓ_2 norm of a document vector during a change of basis,[16] which we theorize is a better fit for the pivoted normalization [37] in the dtb.nnn SMART weighting scheme.

Medium and large models After the submission deadline, we trained fastText for two epochs on both the Math StackExchange corpus and the no_problem and warning_1 subsets of arXMLiv (medium), and for ten epochs on both the Math StackExchange corpus and the no_problem, warning_1, and warning_2 subsets of arXMLiv (large). The medium and large models were not submitted to the competition and only serve for comparison.

For task 1, the results are the closest 1,000 answers in the soft VSM. For task 2, the results are the closest 1,000 formulae in the soft VSM that did not originate from a comment.

4.5 Results

In this section, we will discuss the accuracy of the SCM using nDCG′ on the official human-annotated relevance judgements for tasks 1 and 2, and the speed of the SCM.

[16] `https://github.com/RaRe-Technologies/gensim/pull/2783`

Accuracy For task 1, only the small model was submitted out of the tiny (0.178), small (0.224), *medium (0.237)*, and large (0.231). We conjecture that the poorer performance of the large model was due to training on the warning_2 subset of arXMLiv, which contains highly malformed LaTeX documents.

For task 2, both the tiny and small models were submitted out of the *tiny (0.119)*, small (0.059), medium (0.078), and large (0.075). We conjecture that the poorer performance of all models larger than tiny was either due to the change from cosine normalization to pivoted normalization, or due to the reduced range of modeled character *n*-gram sizes.

Speed For task 1, the average run time of the small, medium, and large models over all topics was 58.46 seconds with the minimum of 30.52 seconds for topic A.88 and the maximum of 502.84 seconds for topic A.35. For task 2, the average run time of the small, medium, and large models over all topics was 108.86 seconds with the minimum of 54.81 seconds for topic B.88 and the maximum of 2720.14 seconds for topic B.25. For the tiny model, the speed may be different due to the different normalization in the soft VSM document similarity function.

The SCM ran on a machine with 2 TiB of free disk space, eight Intel Xeon™ X7560 2.26 GHz processors with a total of 32 CPU cores, 252 GiB RAM and no GPUs. For the SCM, all documents were stored in the RAM, and queries were processed using matrix multiplication on one CPU core without any index. In a production system, the SCM can be reduced to the run time of a TF-IDF-based system, [27, 29, 22] see the performance of MIaS in Section 3.2.

"Thought can organize the world so well that you are no longer able to see it."
Anthony de Mello

5 Formula2Vec

Many machine learning algorithms require the input to be represented as a fixed-length feature vector. Architecturally similar to the fastText model used in Section 4 by the SCM, the Doc2Vec DBOW model [15] is a staple in the area of text classification. The shallow multilayer perceptron can be trained on large volumes of text and infers embeddings for unseen retrieval units.

In this section, we will describe the Formula2Vec system that uses Doc2Vec to infer document and formula embeddings for tasks 1 and 2, respectively. We will also report the results of Formula2Vec on tasks 1 and 2 of the ARQMath 2020 competition. Our experimental code is available online.[17]

5.1 Document Similarity Measure

In Formula2Vec, documents and formulae are represented by document and formula embeddings produced by traning the Doc2Vec DBOW model [15] on text and math in a corpus. For the Math StackExchange corpus, we use the body

[17] `https://github.com/MIR-MU/SCM-at-ARQMath`, file `json_to_doc2vec.py`

text of posts. For the arXMLiv corpus, we use the full texts. To differentiate the token type (text and math) of *n*-grams extracted by fastText, we lower-case all text tokens and upper-case all math tokens.

For task 1, we represent answers from the Math StackExchange corpus as the text and math in their body text. For task 2, we represent formulae from questions and answers in the Math StackExchange corpus as the math. For both tasks 1 and 2, we represent topics as the text and math in their title, body text, and tags.

5.2 Parameter Optimization

Due to the architectural similarities between the SCM and Doc2Vec, we reused the results of the parameter optimization of the SCM, see Section 4.3: 300 vector dimensions and a minimum of five occurences of a term. We modeled phrases with a maximum length of two, and trained on the no_problem subset of arXMLiv in the prefix notation for five epochs.

The remaining parameters were taken from the searchisko information retrieval system:[18] 300 vector dimensions, negative sampling loss with twelve negative samples, initial learning rate 0.1 with a linear decay to zero, and a window size of eight.

We use the implementation of Doc2Vec in Gensim.[19] [25]

5.3 Evaluation

We trained the tiny, small, medium, and large models, and used them to produce the competition submissions.

Tiny model For the task 2 alternative submission, we used the model produced via parameter selection.

Small model For the task 1 alternative submission and the task 2 primary submission, we trained Doc2Vec for ten epochs on both the Math StackExchange corpus and the no_problem subset of arXMLiv.

Medium and large models After the submission deadline, we trained Doc2Vec for two epochs on both the Math StackExchange corpus and the no_problem and warning_1 subsets of arXMLiv (medium), and for ten epochs on both the Math StackExchange corpus and the no_problem, warning_1, and warning_2 subsets of arXMLiv (large). The medium and large models were not submitted to the competition and only serve for comparison.

For task 1, the results are the closest 1,000 answers in the soft VSM. For task 2, the results are the closest 1,000 formulae in the soft VSM that did not originate from a comment.

[18] `https://github.com/searchisko/searchisko`, file `Doc2Vec_wrapper.py`

[19] `https://radimrehurek.com/gensim/models/doc2vec.html`

5.4 Results

In this section, we will discuss the accuracy of Formula2Vec using nDCG' on the official human-annotated relevance judgements for tasks 1 and 2, and the speed of Formula2Vec.

Accuracy For task 1, only the small model was submitted out of the *tiny (0.101)*, small (0.050), medium (0.054), and large (0.074). We conjecture that the poorer performance of all models larger than tiny was due to training on the arXMLiv corpus, which contains scientific articles that are significantly longer and lexically different compared to the short-form questions and answers in the Math StackExchange corpus.

For task 2, both the tiny and small models were submitted out of the tiny (0.077), *small (0.108)*, medium (0.063), and large (0.078). We conjecture that the poorer performance of the tiny and medium models is due to the small number of epochs (five and two, respectively) used for their training. In the Gensim implementation of Doc2Vec, the same number of epochs is also used to infer embeddings of unseen retrieval units. By contrast, both the small and large models used ten epochs for both training and inference. We conjecture that the poorer performance of the large model was due to training on the warning_2 subset of arXMLiv, which contains highly malformed LaTeX documents.

Speed For task 1, the average run time of the large model over all topics was 3.23 seconds with the minimum of 3.14 seconds for topic A.11 and the maximum of 7.87 seconds for topic A.1. For task 2, the average run time of the large model over all topics was 164.5 seconds with the minimum of 61.61 seconds for topic B.26 and the maximum of 5448.65 seconds for topic B.20. For the other models, the speed may be different due to the different number of epochs used for inference.

Like the SCM, Formula2Vec ran on a machine with 2 TiB free disk space, eight Intel Xeon™ X7560 2.26 GHz processors with a total of 32 CPU cores, 252 GiB RAM and no GPUs. Like for the SCM, all documents were stored in the RAM, and queries were processed using matrix multiplication on one CPU core without any index. In a production system, Formula2Vec can be reduced to the run time of a TF-IDF-based system, [27] see the performance of MIaS in Section 3.2.

"If what you seek is Truth, there is one thing you must have above all else." "I know. An overwhelming passion for it." "No. An unremitting readiness to admit you may be wrong." Anthony de Mello

6 CompuBERT

Our CompuBERT system aims to utilize the expressive power of pre-trained Transformer models [45] and the results of applying the Transformer architec-

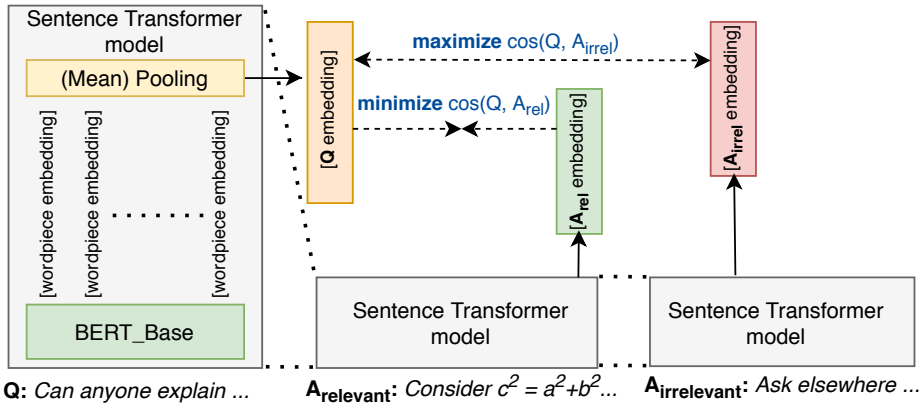

Fig. 7. CompuBERT model and Triplet objective, introduced by Reimers and Gurevych: [26] Informally, the system combines the Wordpiece embeddings [7] to create static representations of questions, that are similar to the representations of relevant answers, while dissimilar to the representations of irrelevant answers.

ture to complex math-related tasks, such as computing derivatives and first-order differential equations by representing math formulae in the prefix notation. [17]

In this section, we will describe CompuBERT and its results on task 1 of the ARQMath 2020 competition. Our experimental code is available online.[20]

6.1 Matching Questions with Answers

In addition to math representation, we have to contend with additional challenges characteristic to information retrieval but alien to Transformers: While the original Transformer architecture [45] builds upon the Wordpiece text segmentation [35] that optimizes the representation of subwords (not unlike fastText in the SCM), we also need to uniformly represent long spans of text.

We address this challenge with an approach introduced by Reimers and Gurevych [26] and shown in Figure 7. The underlying idea of their *Sentence Transformers* is to adjust the pre-trained language model so that it reflects the pairwise similarity of pieces of text, where this similarity is known.

This approach has shown to reach state-of-the-art results on several document classification tasks, where it minimizes the pairwise distance of documents of the same class and maximizes the distance between documents in different classes.

Similarly, in order to get a unified representation of spans of texts representing both questions and answers, the Transformer architecture is extended with an additional pooling layer. The training of the system mimics the Siamese net-

[20] `https://github.com/MIR-MU/CompuBERT`

work architecture [34] with a *Triplet* objective to minimize the cosine distance of the similar blocks of text, as shown in Figure 7.

COMPUBERT uses the Triplet objective as a proxy to the objective of task 1: to minimize the cosine distance of questions to their high-ranking answers, while maximizing the distance to their low-ranking answers. Specifically, we use the following objective:

$$\text{minimize} \quad \sum_{i=1}^{|Qs|} \sum_{j=1}^{|A_i|} \left| \left(1 - \cos(q_i, a_{ij})\right) - \text{dist}_{\text{exp}}(q_i, a_{ij}) \right|, \tag{3}$$

where the expected distance dist_{exp} of an answer a_{ij} to the question q_i is the number of upvotes given to a_{ij}, standardized with respect to all answers to q_i.

Using a similar objective, we can fine-tune an arbitrary embedding model to respect either inherent ranking provided by a feedback of the users, or an explicit relevance ranking provided by relevance judgements.

Adapting the embeddings of the Transformer architecture to Information Retrieval is an active field of research. [24, 18, 5] However, COMPUBERT is, to our knowledge, the first application of pairwise, multimodal (joint text and math) embedding optimization applied in Information Retrieval.

6.2 Model Training

We represent topics and questions and answers from the Math StackExchange corpus as the text and math in their body text. For math, we experiment with different representations: LaTeX, prefix, and infix.

In order to train the system to minimize the distance of relevant answers to a question, we used the Math StackExchange corpus (see Section 2.2) and our relevance judgements (see Section 2.4) with the nDCG' and Spearman's ρ as our objectives (see Section 2.5). We represent math in each question and answer using either LaTeX, the prefix notation, or the infix notation (see Section 2.1). Finally, we iterate over the data set 3–4 times until we reach the convergence with respect to our objectives.

We performed 35 experiments with different configurations of COMPUBERT, and we submitted the configuration of COMPUBERT with the highest nDCG' as a primary submission. See Table 2 for the nDCG' of selected COMPUBERT configurations. See Figure 8 for the learning curves of the submitted COMPUBERT configuration.

6.3 Results

In this section, we will discuss the accuracy of COMPUBERT using nDCG' on the official human-annotated relevance judgements for task 1, and the speed of COMPUBERT.

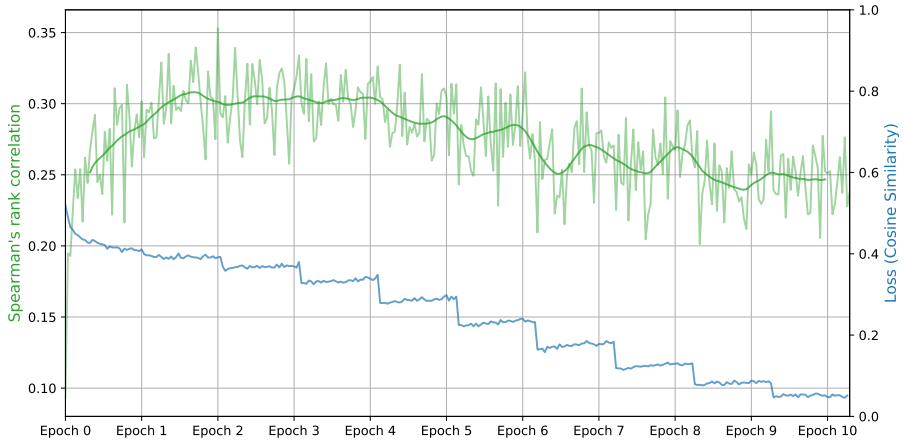

Fig. 8. Spearman's ρ and the corresponding loss (averaged over 20 batches) of training the submitted CompuBERT configuration. Based on the monitoring, we chose to use the model weights from after epoch 4, before the performance started to degrade.

Accuracy We expected the nDCG' and Spearman's ρ to reflect the same objective and we've observed that to be the case for the most cases: Out of 18 configurations with both the nDCG' and Spearman's ρ measured, the ranking based on Spearman's ρ diverged from the nDCG' only for some three systems. However, the automatic and human-annotated relevance judgements for task 1 have proven to be divergent. The best configuration, reaching 0.78 nDCG' on our relevance judgements, reached only 0.009 nDCG' on the human-annotated relevance judgements for task 1, which is not significantly better than zero.

After the competition, we experimented with additional configurations:

1. We trained on the human-annotated relevance judgements, using weighted sampling to correct class imbalance (see Section 2.4).
2. We replaced the standardization of dist_{exp} with linear scaling to $[0; 1]$.
3. We replaced the Triplet objective with the objective of a related Transformer model that produces embeddings for questions and answers. [11, Section 4.a]
4. We assigned questions and answers different token types.

None of the configurations achieved significantly better nDCG' than zero on the human-annotated relevance judgements for task 1.

Speed The submitted system was trained in 4 iterations over 1.4 million pairs of question-answer bodies with math represented in LaTeX. Each iteration took 6 hours and 30 minutes to finish, using NVIDIA GTX 2080 Ti, with 11 GiB of VRAM. We were able to fit batches of 25 question-answer pairs in-memory in a training step and hence we adjusted weights based on such amount of data.

Table 2. The nDCG′ on our relevance judgements for selected CompuBERT configurations. The bert-base-cased configurations were pre-trained as described by Devlin et al. [7] and the bert-base-wikipedia-sections-mean-tokens and bert-base-nli-mean-tokens configurations were pretrained using similar objective to ours, as described by Reimers and Gurevych. In the explicit token_type configuration, we assigned a distinct mask for math tokens on input, often used to distinguish between different input languages. [7]

#	Pre-trained checkpoint	Representation	Text preprocessing	Epochs	nDCG′
1	bert-base-cased	LaTeX	tag removal	4	0.7796
2	-wiki-sections-mean-tokens	LaTeX	tag removal	5	0.7688
3	-nli-mean-tokens	LaTeX	tag removal	4	0.7669
4	bert-base-cased	LaTeX	tag removal	9	0.7659
5	bert-base-cased	LaTeX	none	4	0.7653
6	bert-base-cased	Prefix	tag removal	4	0.7651
7	bert-base-cased	Prefix	tag removal	8	0.7645
8	bert-base-cased	Infix	tag removal	4	0.7603
9	bert-base-cased	LaTeX	explicit token_type	4	0.7598

We fine-tuned the pre-trained Bert-base-cased snapshot of the Bert-base architecture. [7] Following the example of Reimers and Gurevych, [26] we choose the top-layer representation to be a 768-float vector.

The trained system was able to index all 1.4 million answers by inferring the static embeddings of their content, preprocessed in the same manner. Indexing using the same GPU takes roughly four times less than training: 1 hour and 44 minutes. Once all the answers are indexed, the system consumes 27 GiB of RAM to keep the embeddings in-memory. This can be further optimized using vector databases, if needed.

Retrieval for a single topic requires an inference of topic embedding and the retrieval of a ranked list of answers, based on the similarity of the embeddings. We use exact nearest neighbor search and compute the cosine similarity between the topic embedding and each of the indexed answers. Single embedding inference takes 0.004 seconds on GPU and an exact-search retrieval of requested 1,000 ranked questions – just as *all* ranked answers – takes 3.426 seconds on average, 3.667 seconds maximum and 3.198 seconds minimum.

> "There are two ways to wash dishes: One is to wash them in order to make them clean; the other is to wash them in order to wash them."
>
> Anthony de Mello

7 Ensemble

Different information retrieval systems can agree on a small portion of the most relevant documents, but each individual system will miss the great majority of relevant documents. With ensembling, we can combine the strenghts of different information retrieval systems to produce more accurate results.

In this section, we describe a parameter-free algorithm for ensembling an arbitrary number of result lists into a single result list. The algorithm is agnostic to the scoring functions used by the different systems and only uses the ranks of results. Our experimental code is available online.[21]

7.1 Algorithm

Definition 1 (Frequency f). *Let T be a topic. Let R be a result for T that appears in exactly f result lists. The frequency of R is f.*

Definition 2 (Median Inverse Rank M^{-1}). *Let T be a topic. Let R be a result for T that appears in any of the N result lists. Let M be the median of $rank_1, rank_2, \ldots, rank_N$, where $rank_i, 0 \leq rank_i \leq 1000$ is the zero-based rank of R in the i-th result list if R appears in the i-th result list, and 1000 otherwise. The median inverse rank M^{-1} of R is $(1000 - M)/1000$.*

Definition 3 (Striped Inverse Rank S^{-1}). *Let T be a topic. Let R be the k-th result for T (according to some criterion). Let $S = rank_i$, where $i \equiv k \pmod{f}$ and $rank_i, 0 \leq rank_i \leq 1000$ is the zero-based rank of R in the i-th result list of the f result lists in which R appears. The striped inverse rank S^{-1} of R is $(1000 - S)/1000$.*

For each topic T, we combine the result lists of the systems, and we lexicographically order them using the four-tuple $(M^{-1}, f, S^{-1}, \text{result name})$. The top 1,000 ordered results are the ensembled result list for topic T.

The rationale of the algorithm is the following: If multiple results with the same M^{-1} exist, then f is used to break the ties. If ties still exist, the tied results are interleaved using S^{-1} to ensure fairness. The remaining ties are broken arbitrarily using the result name.

In our implementation, we only reported M^{-1} as the score in the ensembled result lists. Since the trec_eval software[22] used for evaluation disregards the reported ranks of results and infers them from the reported scores instead, the ensembled result lists were effectively reordered using the two-tuple $(M^{-1}, \text{result name})$ during evaluation. This flaw in trec_eval has only a limited effect on the nDCG', since the four-tuple $(M^{-1}, f, S^{-1}, \text{result name})$ has already decided which 1,000 results would appear in the result list. See also the discussion of tie-breaking in information retrieval by Cabanac et al. [4]

7.2 Evaluation

For task 1, we ensembled our three primary submissions: MIaS (0.155, see Section 3), the small model of the SCM (0.224, see Section 4), and COMPUBERT (0.009, see Section 6). To assess the capabilities of our ensembling algorithm, we also ensembled all non-baseline primary submissions submitted to task 1

[21] `https://github.com/MIR-MU/SCM-at-ARQMath`, file `combine_serps.py`

[22] `https://github.com/usnistgov/trec_eval`

by the ARQMath participants: alpha05 (0.278), PSU2 (0.228), SCM (0.224), MIaS (0.155), COMPUBERT (0.009), zbMATH (0.101), and DPLR2 (0.051). [47, Table A1]

For task 2, we also ensembled our two primary submissions: the small model of the SCM (0.059, see Section 4), and the small model of Formula2Vec (0.108, see Section 6). To assess the capabilities of our ensembling algorithm, we also ensembled all non-baseline primary submissions submitted to task 2 by the ARQMath participants: TangentCFTED (0.420), Formula2Vec (0.108), and SCM (0.059). [47, Table A2] The primary submission of formulaembedding (0.026) was not included in the ensemble, since it was not shared by the participants.

7.3 Results

In this section, we will discuss the tie-breaking performance of ensembling, and the accuracy of ensembling using nDCG' on the official human-annotated relevance judgements for task 1. Since the ensembling was applied on the result lists as a post-processing step, speed can not be meaningfully reported.

Breaking Ties Table 3 shows how successful our algorithm was in breaking ties. 59.29% of results in task 1 and 47.56% of results in task 2 only had their ties broken using their names. A fairer tie-breaking strategy would have been to combine and lexicographically order the results R using the four-tuple (M^{-1}, f, S^{-1}, index of R's originating result list), so that tied results from different input result list are interleaved.

Table 3. Percentages of ties in the submitted ensembled result lists when ordering the result lists by just a first few ordering criteria.

	(M^{-1},	f,	S^{-1},	result name)
Task 1	92.76%	92.75%	59.29%	0.00%
Task 2	48.33%	47.72%	47.56%	0.00%

Accuracy For task 1, the ensemble of our three primary submissions achieved the highest accuracy of all our systems (0.238). The ensemble of all non-baseline primary submissions[23] received *the highest accuracy in the competition (0.419)*, significantly better than the alpha05noReRank alternative submission of the MathDowsers team (0.345). [47, Table A1]

For task 2, the ensemble of our two primary submissions achieved a slightly lower accuracy (0.100) than Formula2Vec (0.108). The ensemble of all non-baseline primary submissions[24] received *the third highest accuracy results in the competition (0.327)*.

[23] `https://github.com/MIR-MU/SCM-at-ARQMath`, file `ensemble-task1.tsv`

[24] `https://github.com/MIR-MU/SCM-at-ARQMath`, file `ensemble-task2.tsv`

There was no difference in nDCG′ for either task when using the fairer tie-breaking strategy described in the previous section. This is because of the arbitrary reordering of results by trec_eval discussed in Section 7.1: the fairer tie-breaking did not alter the top 1,000 results present in the ensembled result lists.

Figure 9 shows the percentage of judged results at different ranks for all our submissions[25] and offers an alternative interpretation of the surprisingly high accuracy of the ensemble: Due to the pooling of submitted results before annotation, [47, Figure 2] the ensembled result lists have significantly more judged results than any other submission. Since there is no penalty to the nDCG′ for adding non-relevant results (i.e. $\text{gain}(\text{RES}'_i) = 0$) at the end of a result list (see Section 2.5), having more judged results is beneficial even if few are relevant. However, the high mean relevance of the first ten results (see Figure 10) and the best P@10 among our systems [47, Tables A1 and A2] indicate that this is not the main cause of the high accuracy of the ensemble.

"Meaning is only found when you go beyond meaning."
Anthony de Mello

8 Results

In this section, we will describe the accuracy and the speed of all our systems on tasks 1 and 2.

[25] `https://github.com/MIR-MU/ARQMath-eval`, file `mean-relevance-and-percentages-judged.ipynb`

Table 4. The nDCG′ for the primary and secondary submissions of all our systems on tasks 1 (left) and 2 (right) as well as the best nDCG′ among the submitted and out-of-competition system configurations. Best and second best systems in a row are **bold** and *italic*, respectively.

	MIaS	SCM	F2Vec	CBRT	Ens.
Best	0.155	*0.237*	0.101	0.009	**0.419**
Primary	*0.155*	**0.224**		0.009	
Alternative			*0.050*		**0.238**

	SCM	F2Vec	Ens.
Best	*0.119*	0.108	**0.327**
Primary	*0.059*	**0.108**	
Alternative	**0.119**	0.077	*0.100*

Table 5. The average, minimum, and maximum run time in seconds for all our systems to produce results for a task 1 (left) and 2 (right) topic. Best and second best results in a row are **bold** and *italic*, respectively.

	MIaS	SCM	F2Vec	CBRT
Minimum	**0.1**	30.52	*3.14*	3.2
Average	**1.24**	58.64	*3.23*	3.43
Maximum	*7.27*	502.84	7.87	**3.67**

	SCM	F2Vec
Minimum	**54.81**	*61.61*
Average	**108.86**	*164.5*
Maximum	**2720.14**	*5448.65*

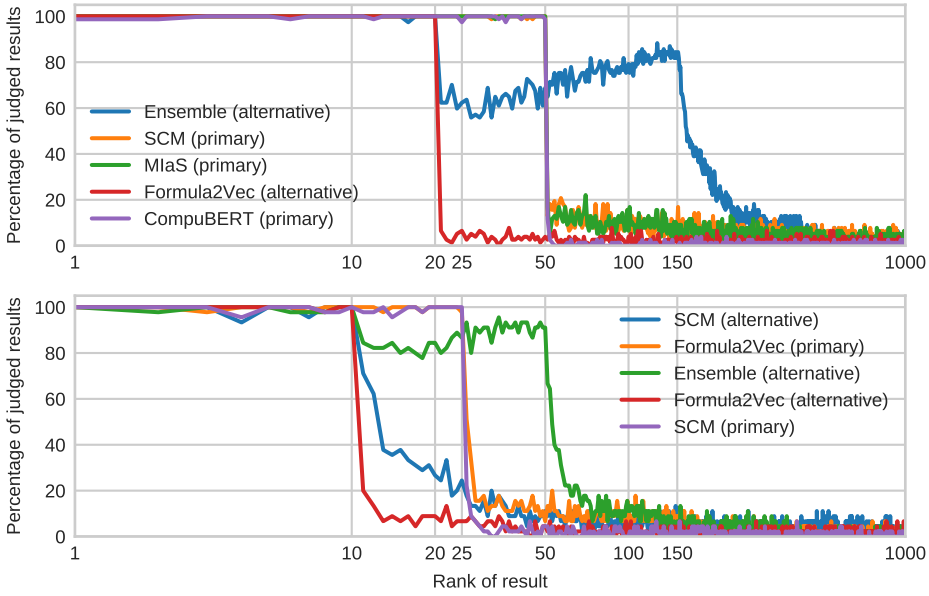

Fig. 9. The percentage of judged results at different ranks for all our task 1 (above) and 2 (below) submissions. Whereas other alternative submissions only have the first 20 and ten results judged for tasks 1 and 2, respectively, due to the pooling of submissions, the ensemble has significantly more judged results, which may put it at an unfair advantage.

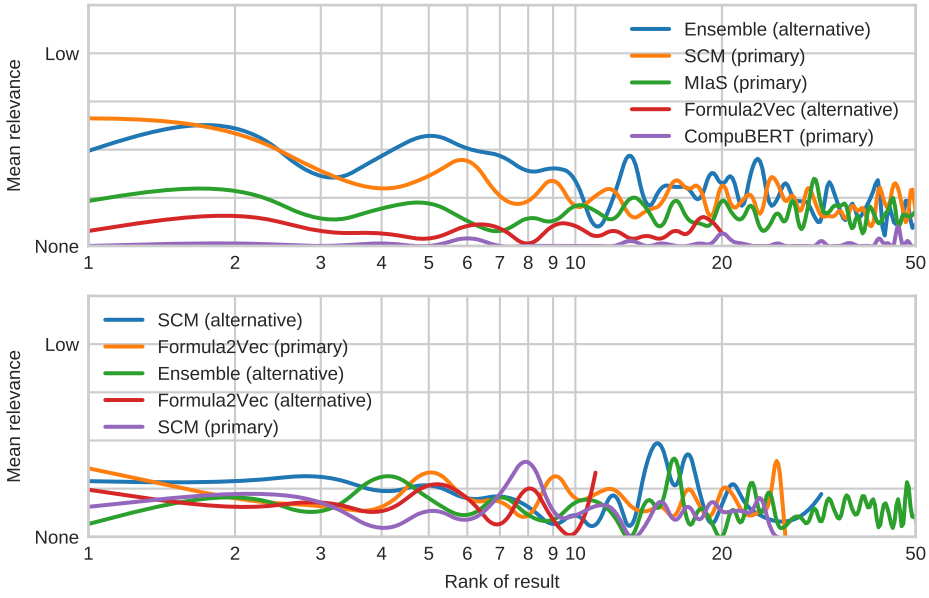

Fig. 10. The mean relevance (gain) for all our task 1 (above) and 2 (below) submissions. For task 1, the ensemble has the highest mean relevance at ten, except for ranks 1 and 3.

8.1 Accuracy

Table 4 shows the nDCG' of all our systems on both tasks. The primary and alternative rows only show the results of our submissions, whereas the best row also include the results of our out-of-competition system configurations.

On task 1, the best configuration of the SCM is significantly better than MIaS, Formula2Vec, and COMPUBERT. On both tasks, the best configuration of the ensemble is significantly better than the ensembled systems.

8.2 Speed

Table 5 shows the run times of all our systems on task 1 and 2 topics. In task 1, MIaS was the fastest, closely followed by Formula2Vec and COMPUBERT. In task 2, SCM was faster than Formula2Vec, which is likely due to the larger number of indexing units and the fact that SCM uses sparse matrix multiplication, which becomes significantly faster than dense matrix multiplication as the matrices grow large.

"Wisdom tends to grow in proportion to one's awareness of one's ignorance."
Anthony de Mello

9 Conclusion and Future Work

In our work, we have introduced three significantly different MIR systems.

The methods based on the vector space model (SCM and MIaS) have shown useful transferability to other tasks as well as interpretability, as it contains the solid ground in relying on concrete term matching.

While COMPUBERT has demonstrated an outstanding ability to learn a specific task of guessing the number of votes of an answer based on its textual and mathematical content, we've also observed an unexpected sensitivity to the training objective, where COMPUBERT was able to significantly outperform others on our relevance judgements but finished last on task 1 of the ARQMath 2020 competition. The possible reasons include overfitting, underfitting, and divergent objectives.

Thanks to the ARQMath competition, we will be able to identify such failures and further fine-tune our systems to better regard the provided human-annotated relevance judgements, that as we believe, better reflect the quality of a MIR system, than our artificial relevance judgements that we had to rely on when developing our systems.

Our most surprising finding is that *three is better than one*: In terms of accuracy, ensembling the results of our primary submissions outperformed all our three primary systems on task 1 of the ARQMath 2020 competition. Ensembling all non-baseline primary submissions for task 1 received the best score in the competition at a statistically significant margin.

Question answering in a complex domain such as STEM is a very challenging task. Looking at it from diverse viewpoints is a good thing and so is compounding and merging diverse approaches and their results! Ensembling worked for the ARQMath tasks, and it may work for other complex tasks in general.

Acknowledgements First author's work was graciously funded by the South Moravian Centre for International Mobility as a part of the Brno Ph.D. Talent project. We also thank the three anonymous reviewers for their insightful comments. Last but not least, we extend our gratitude to the ARQMath 2020 organizers for their great professionalism and patience.

References

1. Ausbrooks, R., Buswell, S., Carlisle, D., Chavchanidze, G., Dalmas, S., Devitt, S., Diaz, A., Dooley, S., Hunter, R., Ion, P., Kohlhase, M., Lazrek, A., Libbrecht, P., Miller, B., Miner, R., Rowley, C., Sargent, M., Smith, B., Soiffer, N., Sutor, R., Watt, S.: Mathematical markup language (MathML) version 3.0 2nd edition (2014), https://www.w3.org/TR/MathML3/Overview.html
2. Białecki, A., Muir, R., Ingersoll, G., Imagination, L.: Apache Lucene 4. In: SIGIR 2012 workshop on open source information retrieval. p. 17 (2012)
3. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017)
4. Cabanac, G., Hubert, G., Boughanem, M., Chrisment, C.: Tie-breaking bias: effect of an uncontrolled parameter on information retrieval evaluation. In: International Conference of the Cross-Language Evaluation Forum for European Languages. pp. 112–123. Springer (2010)
5. Chang, W.C., Yu, F.X., Chang, Y.W., Yang, Y., Kumar, S.: Pre-training Tasks for Embedding-based Large-scale Retrieval. arXiv e-prints arXiv:2002.03932 (Feb 2020)
6. Charlet, D., Damnati, G.: Simbow at semeval-2017 task 3: Soft-cosine semantic similarity between questions for community question answering. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). pp. 315–319 (2017)
7. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
8. Formánek, D., Líška, M., Růžička, M., Sojka, P.: Normalization of digital mathematics library content. In: Davenport, J., Jeuring, J., Lange, C., Libbrecht, P. (eds.) 24th OpenMath Workshop, 7th Workshop on Mathematical User Interfaces (MathUI), and Intelligent Computer Mathematics Work in Progress. pp. 91–103. No. 921 in CEUR Workshop Proceedings, Aachen (2012), http://ceur-ws.org/Vol-921/wip-05.pdf
9. Ginev, D.: arXMLiv:08.2019 dataset, an HTML5 conversion of arXiv.org (2019), https://sigmathling.kwarc.info/resources/arxmliv-dataset-082019/, SIGMathLing – Special Interest Group on Math Linguistics
10. González Barbosa, J.J., Frausto-Solis, J., Villanueva, D., Valdés, G., Florencia, R., González, L., Mata, M.: Implementation of an Information Retrieval System Using the Soft Cosine Measure, vol. 667, pp. 757–766. Springer (Dec 2017), https://doi.org/10.1007/978-3-319-47054-2_50
11. Jernite, Y.: Explain Anything Like I'm Five (June 2020), https://github.com/huggingface/notebooks/blob/master/longform-qa/Long_Form_Question_Answering_with_ELI5_and_Wikipedia.ipynb
12. Krstovski, K., Blei, D.M.: Equation Embeddings. ArXiv e-prints (Mar 2018)

13. Landis, J.R., Koch, G.G.: The measurement of observer agreement for categorical data. biometrics pp. 159–174 (1977)
14. Larson, R.R., Reynolds, C., Gey, F.C.: The abject failure of keyword IR for mathematics search: Berkeley at NTCIR-10 math. In: Proceedings of the 10th NTCIR Conference on Evaluation of Information Access Technologies, NTCIR-10, National Center of Sciences, Tokyo, Japan, June 18–21, 2013 (2013)
15. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. CoRR **abs/1405.4053** (2014), http://arxiv.org/abs/1405.4053
16. Líška, M., Sojka, P., Růžička, 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. NWSearch '15, Association for Computing Machinery, New York, NY, USA (2015), https://doi.org/10.1145/2810355.2810359
17. Lu, Y., Li, Z., He, D., Sun, Z., Dong, B., Qin, T., Wang, L., Liu, T.Y.: Understanding and improving transformer from a multi-particle dynamic system point of view. arXiv preprint arXiv:1906.02762 (2019)
18. Luan, Y., Eisenstein, J., Toutanova, K., Collins, M.: Sparse, Dense, and Attentional Representations for Text Retrieval. arXiv e-prints arXiv:2005.00181 (Apr 2020)
19. Mansouri, B., Rohatgi, S., Oard, D.W., Wu, J., Giles, C.L., Zanibbi, R.: Tangent-CFT: An Embedding Model for Mathematical Formulas. In: Proceedings of the 2019 ACM SIGIR international conference on theory of Information Retrieval. pp. 11–18 (2019)
20. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed Representations of Words and Phrases and their Compositionality. In: Burges, C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 26, pp. 3111–3119. Curran Associates, Inc. (2013)
21. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781 (2013)
22. Novotný, V.: Implementation notes for the soft cosine measure. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18). pp. 1639–1642. Association for Computing Machinery, Torino, Italy (2018). https://doi.org/10.1145/3269206.3269317
23. Novotný, V., Ayetiran, E.F., Štefánik, M., Sojka, P.: Text classification with word embedding regularization and soft similarity measure (2020), https://arxiv.org/abs/2003.05019
24. Qadrud-Din, J., Bah Rabiou, A., Walker, R., Soni, R., Gajek, M., Pack, G., Rangaraj, A.: Transformer Based Language Models for Similar Text Retrieval and Ranking. arXiv e-prints arXiv:2005.04588 (May 2020)
25. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (May 2010). https://doi.org/10.13140/2.1.2393.1847
26. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 3982–3992. Association for Computational Linguistics, Hong Kong, China (Nov 2019). https://doi.org/10.18653/v1/D19-1410, https://www.aclweb.org/anthology/D19-1410
27. Rygl, J., Pomikálek, J., Řehůřek, R., Růžička, M., Novotný, V., Sojka, P.: Semantic vector encoding and similarity search using fulltext search engines. In: Proceedings of the 2nd Workshop on Representation Learning for NLP. pp. 81–

90. Association for Computational Linguistics, Vancouver, Canada (Aug 2017). https://doi.org/10.18653/v1/W17-2611
28. Růžička, M.: Maths Information Retrieval for Digital Libraries. Ph.D. thesis, Masaryk University, Faculty of Informatics, Brno (2018), https://is.muni.cz/th/pxz4q/
29. Růžička, M., Novotný, V., Sojka, P., Pomikálek, J., Řehůřek, R.: Flexible Similarity Search of Semantic Vectors Using Fulltext Search Engines. In: HybridSemStats Workshop Proceedings. vol. 1923, pp. 1–12 (2017)
30. Růžička, M., Sojka, P., Líška, M.: Math Indexer and Searcher under the Hood: History and Development of a Winning Strategy. In: Joho, H., Kishida, K. (eds.) Proc. of the 11th NTCIR Conference on Evaluation of Information Access Technologies. pp. 127–134. NII, Tokyo, Japan (Dec 2014)
31. Růžička, M., Sojka, P., Líška, M.: Math indexer and searcher under the hood: Fine-tuning query expansion and unification strategies. In: Kando, N., Kishida, K., Kato, M.P., Yamamoto, S. (eds.) Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies. pp. 331–337. National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430 Japan, Tokyo (2016)
32. Sakai, T., Kando, N.: On information retrieval metrics designed for evaluation with incomplete relevance assessments. Information Retrieval **11**(5), 447–470 (2008)
33. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing and Management **24**, 513–523 (1988)
34. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: A unified embedding for face recognition and clustering. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 815–823 (2015). https://doi.org/10.1109/CVPR.2015.7298682
35. Sennrich, R., Haddow, B., Birch, A.: Neural Machine Translation of Rare Words with Subword Units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1715–1725. Association for Computational Linguistics, Berlin, Germany (Aug 2016). https://doi.org/10.18653/v1/P16-1162, https://www.aclweb.org/anthology/P16-1162
36. Sidorov, G., Gelbukh, A., Gómez-Adorno, H., Pinto, D.: Soft similarity and soft cosine measure: Similarity of features in vector space model. Computación y Sistemas **18**(3), 491–504 (2014)
37. Singhal, A., Buckley, C., Mitra, M.: Pivoted document length normalization. In: ACM SIGIR forum. vol. 51, pp. 176–184. ACM New York, NY, USA (2017)
38. Singhal, A., Choi, J., Hindle, D., Lewis, D.D., Pereira, F.: AT&T at TREC-7. In: Proceedings of the seventh Text REtrieval Conference (TREC-7). pp. 239–252 (July 1999)
39. Singhal, A., et al.: Modern information retrieval: A brief overview. IEEE Data Eng. Bull. **24**(4), 35–43 (2001)
40. Sojka, P., Líška, M.: The Art of Mathematics Retrieval. In: Proceedings of the ACM Conference on Document Engineering, DocEng 2011. pp. 57–60. Association of Computing Machinery, Mountain View, CA, USA (Sep 2011), http://doi.acm.org/10.1145/2034691.2034703
41. Sojka, P., Novotný, V., Ayetiran, E.F., Lupták, D., Štefánik, M.: Quo Vadis, Math Information Retrieval. In: Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019. pp. 117–128 (2019), https://nlp.fi.muni.cz/raslan/2019/paper11-sojka.pdf
42. Sojka, P., Růžička, M., Novotný, V.: MIaS: Math-Aware Retrieval in Digital Mathematical Libraries. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18).

pp. 1923–1926. Association for Computing Machinery, Torino, Italy (2018). https://doi.org/10.1145/3269206.3269233
43. Stamerjohanns, H., Kohlhase, M., Ginev, D., David, C., Miller, B.: Transforming large collections of scientific publications to xml. Mathematics in Computer Science **3**(3), 299–307 (2010)
44. Student: The probable error of a mean. Biometrika **6**(1), 1–25 (1908)
45. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is All you Need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30. pp. 5998–6008. Curran Associates, Inc. (2017), http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf
46. Wojciechowski, K., Nowiński, A., Sojka, P., Líška, M.: The EuDML Search and Browsing Service – Final (Feb 2013), deliverable D5.3 of EU CIP-ICT-PSP project 250503 EuDML: The European Digital Mathematics Library, revision 1.2 https://project.eudml.eu/sites/default/files/D5_3_v1.2.pdf
47. Zanibbi, R., Oard, D.W., Agarwal, A., Mansouri, B.: Overview of ARQMath 2020: CLEF Lab on Answer Retrieval for Questions on Math (2020)