TECHNISCHE
UNIVERSITÄT
DRESDEN

Dep. of Computer Science Institute for System Architecture, Database Technology Group

Master Thesis

RETRIEVAL OF FORMULAS FROM ARQMATH CHALLENGE DATASET USING JOINT EMBEDDINGS

Vidya Chandrashekar
Matr.-Nr.: 4815240

Supervised by:
Dr.-Ing. Claudio Hartmann

and:
Prof. Dr.-Ing. Wolfgang Lehner

Submitted on 20 Dec 2021

CONFIRMATION

I confirm that I independently prepared the thesis and that I used only the references and auxiliary means indicated in the thesis.

Vidya. S

Vidya Chandrashekar

Dresden, 20 Dec 2021

ABSTRACT

Investigating methods and techniques for making mathematical knowledge accessible to information retrieval tools is extremely necessary in today's world of the information-rich era. Achieving this goal requires methods that can reliably extract mathematical knowledge from technical and scientific documents. In the domain of Natural Language Processing (NLP), various general-purpose text processing methods and tools exist that can be applied to a text to entitle domain-specific extraction tasks. With this regard, ARQMath 2020 Challenge focuses mainly on retrieving answers to mathematical questions (Task 1), and formula retrieval (Task 2). Since this challenge includes both text and formulas, approaches from NLP and Information Retrieval have to be adapted in order to interpret the semantics of mathematical formulas. Hence this thesis report is mainly focused on solving Task 2, where a novel method is proposed for formula retrieval. The method takes formulas in LaTeX format, their context words and uses the method of joint embedding to combine their vector spaces. With the help of a similarity score, a ranked list of formulas is obtained which is compared against the user's query formula. The proposal has two variants: (i) using the entire formulas for training and (ii) splitting the formulas as LaTeX tokens for training. The evaluation metrics used were nDCG', mAP, p@10. The results were compared against the gold dataset (qrel file) which is obtained from human assessment methods of pooling and annotation. The proposed method gave promising results by extracting the relevant formulas either as an exact match or as a subexpression match as that of the query formula. But overall, it performed poorly when compared to the baseline method, Tangent-S. Improvement in terms of searching and ranking methodology could be done for better results. This novel approach opens the door of opportunities to explore more in the area of formula embedding using machine translation models.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

This chapter describes the motivation behind the thesis and the main goal that is intended to be solved. Thesis is mainly focused on solving Task 2 of ARQMath 2020 Challenge using a novel approach.

1.1 MOTIVATION

The Web is a rich repository of mathematical information, mainly technical documents, online paper databases, tutorials and instructional materials, and other publications. A great deal of research has created efficient search engines for large corpora in information retrieval [MRS08]. Fast and precise text search engines are widely used in enterprise and Web environments. As the world of information technology grows, searching data of interest has become one of the most critical tasks. Identification and retrieval of textual information are relatively mature methods, with widespread availability of text-based search engines.

Mathematical Information Retrieval (MIR) [AK20] is one of the areas of research that mainly focuses on information retrieval in the domain of Science, Technology, Engineering, and Mathematics (STEM). Interest in MIR has increased tremendously in recent years among the research communities, for instance, various math retrieval tasks performed at the NTCIR[1] conferences. Mathematical formulas play a vital role in scientific articles, but they are not easy to handle using the existing techniques. They are challenging to work with because every mathematical formula is unique, or most formulas occur only once. An intuitive understanding of formulas is necessary for analyzing and understanding scientific literature. Meaningful representations of formulas help make connections between articles, improve retrieval of scientific texts, and help create tools for exploring and navigating scientific literature.

Mathematics can describe complex relations in a more straightforward and understandable way through mathematical expressions. But there exists a certain level of ambiguity in expressions too. To mitigate ambiguity, shorter explanations and definitions are required to set the proper con-

[1] `http://research.nii.ac.jp/ntcir/index-en.html`

text. For computers, understanding mathematical expressions and analyzing them is challenging because of inherent issues in natural language. Hence a system is required that understands the semantics of mathematical expressions used for various applications, from search engines to recommendation systems [Gre+20]. Understanding mathematical expressions essentially mean comprehending the internal components' semantic value, which can be achieved by linking its elements with their corresponding mathematical definitions.

Mathematical formulas play an essential role in communicating scientific information. Their purpose is not just limited to numerical calculation but also clarifies the definitions or explanations written in natural language text. Mathematical formulas have many qualities in common when compared with natural language. It also contains terms, symbols, grammatical rules for symbols. The significant differences between natural language and formula are that mathematical formula is highly ambiguous. For instance, a variable y can have different meanings in a different context; it's challenging to understand without explanation. Consider, in the formula, $y = 2x$ is totally different from the formula $y = mx + c$. Usually, formulas are recursive or nested in structure, whereas natural language is generally linear in form. formulas are highly structured, represented as layout or tree structure, e.g., LaTeX or Math ML. These characteristics may lead to difficulties applying natural language approaches to mathematical formulas for analysis and understanding. Despite the known importance of math, many conventional information retrieval systems do not support access to users for formulas in technical documents [Gao+17].

Information Retrieval is one of the application areas of Natural Language Processing (NLP) which provides information content to end-users based on their needs. This information can vary from individual to individual, making the design of search engines more challenging. As conventional search engines are loaded with text and multimedia content search capabilities, they still lack the competence in searching mathematical information from technical and scientific documents. Information-rich contents of the scientific papers serve as a crucial input to many scientific and technological research, which are mainly rich mathematical formulas. Conventional text-based indexing and search techniques often fail to retrieve information from such documents [PPG18].

Mathematical Information Retrieval (MIR) mainly focuses on retrieving mathematical formulas which is an important and challenging area in Information Retrieval (IR). Recent years have shown a surge in popularity and number of MIR systems, specifically intended to retrieve mathematical contents (such as formulas and mathematical equations) from scientific documents. MIR and conventional Text Information Retrieval (TIR) is altogether different. Unlike MIR, TIR involves the relatively straightforward lexical matching of terms. In MIR, complex methodology and reasoning are incurred in comprehending the deeper semantics of mathematical formulas. The problem of retrieving math notation is closely related to the problem of recognizing math notation. The system must be able to acknowledge math expressions that the user provides as a query and also must be able to recognize math expressions in the target documents. The math domain challenges the researchers with its complexity. Generally, searching for mathematical formulas is a non-trivial problem because Mathematical notation is context-dependent; they can come in various notations depending on the context. The goal is to retrieve all forms irrespective of the notations. Sometimes, identical presentations can stand for multiple distinct mathematical objects, e.g., an integral expression of the form $Rf(x)dx$ can mean a Riemann Integral, a Lebesgue Integral, or any other form known anti-derivative operators. The ability to restrict the search to the particular integral type interested is challenging. Current MathIR approaches [KTA14] try to extract textual descriptors of the parts that compose mathematical equations.

1.2 GOAL

This thesis aims to accelerate the research in MIR and includes mainly Task 2 of ARQMath Challenge[2]. It involves retrieving relevant formulas that are written in LaTeX notation. These are retrieved from answer posts for a question asked on the Mathematics Stack Exchange[3] (MSE), a platform where users post questions to be answered by the community. The questions and answers are related to mathematical topics. Natural Language Processing (NLP) [VC13] technique called joint embedding method is used to retrieve relevant formulas for a query formula posed. In this approach, the embedding of formulas and context words are done jointly with the help of a translational vector model, which will be described in detail in further sections. A run file is generated for each topic; the systems retrieve a ranked list of various formulas to evaluate the method's performance. A graded relevance scale is used for each retrieved formula for scoring ranging from 0 (not relevant) to 3 (highly relevant). The evaluation metric to test the method's performance against existing methods are nDCG′, mAP, and p@10. The overview is as shown in Figure 1.1.

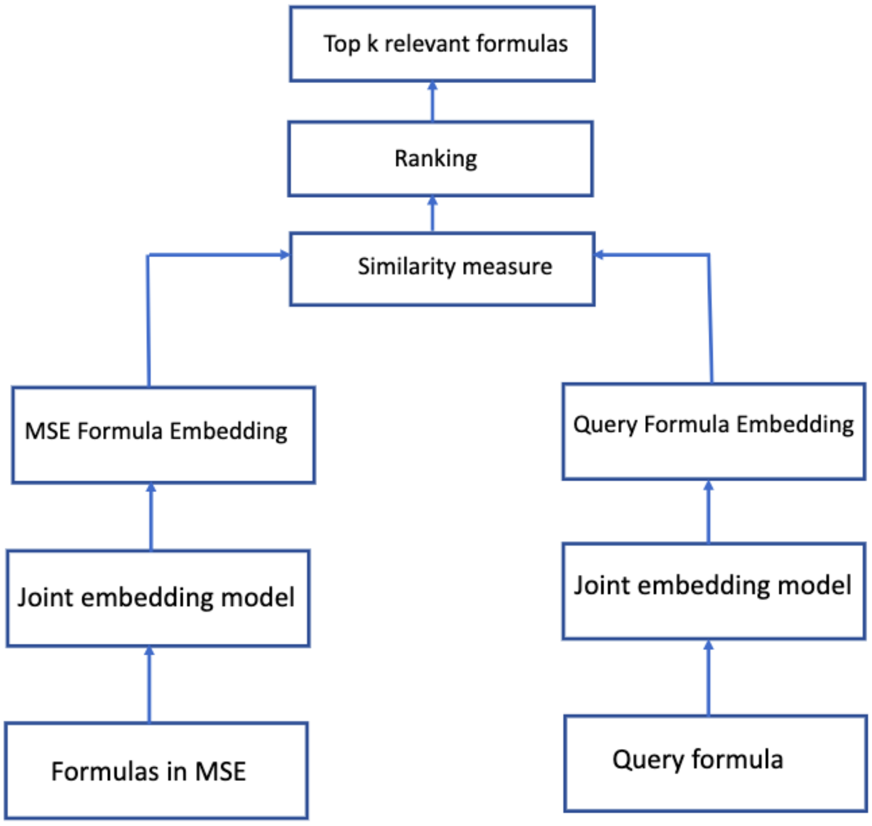


Figure 1.1: Overview of the task

[2]https://www.cs.rit.edu/~dprl/ARQMath/
[3]https://math.stackexchange.com/

2 LITERATURE REVIEW

This section describes the previous related work in the area of formula retrieval task, their approach, and methodologies. Mathematics is usually expressed through imprecise and less accurate descriptions, contributing to machine learning applications for information retrieval in this domain. In scientific literature, searching mathematical formulas usually follows the method of generating string representation of the formulas and uses conventional information retrieval methods. There are various proposed approaches for representing mathematical formulas and their retrieval.

2.1 TREE BASED APPROACH

A symbol layout tree is a representation that encodes the relationship of variables and operators for indexing and retrieving mathematical formulas. It is also seen that large-scale search engines support formula search through LaTeX strings as search terms. Math encodings can be represented as hierarchical where formula syntax is defined in Symbol Layout Tree (SLT) encodings such as LaTeX and formula semantics in Operator Tree (OT) encodings [Figure 2.1].

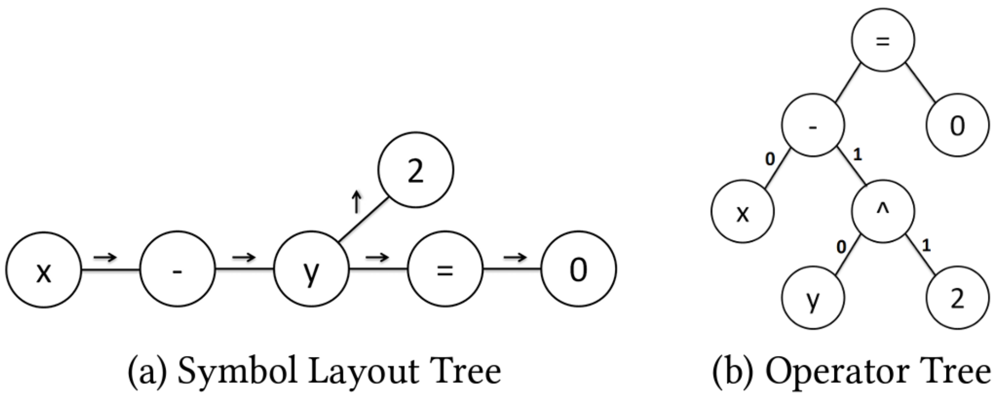


Figure 2.1: Tree representations for the formula $x - y^2 = 0$ [DZ17]

In SLT, the query formulas are tokenized and structured to match its notationally different, sub-formulas, or semantically similar variants. Since math formulas are highly structured, tree-based techniques transform math formulas into tree representation, where the nodes represent operators and the leaves represent operands or variables. A tree representation is traversed in a query formula to detect the structural similarity between the query and indexed formulas. SLTs are modified and normalized for representing operator precedence. LaTeX string is helpful as a baseline math representation and for more fine-grained math representations. Interestingly in one of the methods [Gre+20], it was found that combining both SLT and OT was useful, where the OT captures the formula semantics, and SLT focuses on the visual structure of the formula.

2.2 NEURAL NETWORK APPROACH

Recently, neural network approaches are significant for textual content representation, which can discover distributed representations of words and capture the semantic similarity between the words. Compared to tree-based approaches, the embedding model can learn more abstract formula representations as they do not depend on paths or sub-trees, Therefore they can provide better approximations for formula retrieval. Meanwhile, textual information retrieval is done using various available neural methods where some have achieved significant performance. The formula described by mathematical language has many common qualities concerning natural languages: grammatical rules for combining the symbols. However, there are substantial differences between formula and natural language. formulas always have recursive structures, while natural language is usually linear. These characteristics make it challenging to apply the natural language approaches to mathematical formulas.

A novel structure-based approach [KAB12] was proposed for the Mathematical Expression (ME) retrieval problem. In this approach, the query given in LaTeX format is preprocessed to eliminate extraneous keywords while retaining the structure information. MEs in the database are also preprocessed and stored in the same way. A database of MEs was created in LaTeX form that covers different branches of mathematics like Trigonometry, Algebra, Calculus, Logrithmic, etc. The preprocessed query is matched against the database of preprocessed MEs using Longest Common Subsequence (LCS) [ZW13] algorithm. In this algorithm, the order of keywords is preserved in the preprocessed MEs. Structure information is incorporated into the LCS algorithm, and a measure-based approach is proposed for ranking MEs in the database. As the proposed approach exploits structural information, it is closer to human intuition. Retrieval performance has been evaluated using standard precision measures and found to be performing well.

A new protocol [SZ15] for evaluating math formula retrieval systems was proposed where the Tangent search engine uses an inverted index in math expressions. Each key in the index contains a pair of symbols with their relative distance and vertical displacement. The matched expressions are ranked by using the mean percentage of symbol pairs matched in the query and in the candidate expression. The method was found fast enough for use in real-time and found partial matches well. In this experiment, similarity ratings were done using a 5-point Likert scale [(1) Very dissimilar; (2) Dissimilar; (3) Neutral; (4) Similar; (5) Very Similar], evaluating expressions in a randomized order to avoid bias in search result lists.

2.3 EQUATION EMBEDDING

As equations are unique, they are difficult to analyze. An unsupervised approach was proposed for discovering the semantic representations of mathematical equations called equation embeddings (EqEmb) [KB18]. This proposed approach treats the equation as a singleton word. The surrounding text around the equation provides the information to develop a better representation of the equation. These neighboring words are helpful to learn its embedding vector. The resulting representation, when compared to other equation representations, helps to find both related equations and words Figure 2.2. Given a dataset of words and equations, the goal of the

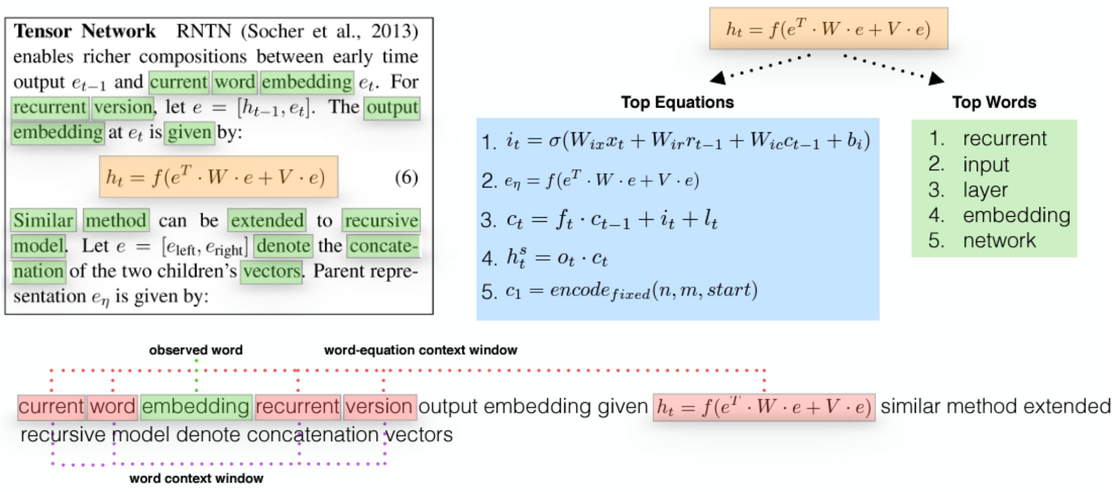

Tensor Network RNTN (Socher et al., 2013) enables richer compositions between early time output e_{t-1} and current word embedding e_t. For recurrent version, let $e = [h_{t-1}, e_t]$. The output embedding at e_t is given by:
$h_t = f(e^T \cdot W \cdot e + V \cdot e)$ (6)
Similar method can be extended to recursive model. Let $e = [e_{\text{left}}, e_{\text{right}}]$ denote the concatenation of the two children's vectors. Parent representation e_η is given by:
$h_t = f(e^T \cdot W \cdot e + V \cdot e)$
Top Equations
1. $i_t = \sigma(W_{ix}x_t + W_{ir}r_{t-1} + W_{ic}c_{t-1} + b_i)$
2. $e_\eta = f(e^T \cdot W \cdot e + V \cdot e)$
3. $c_t = f_t \cdot c_{t-1} + i_t + l_t$
4. $h_t^s = o_t \cdot c_t$
5. $c_1 = encode_{fixed}(n, m, start)$
Top Words
1. recurrent
2. input
3. layer
4. embedding
5. network
observed word
word-equation context window
current word embedding recurrent version output embedding given $h_t = f(e^T \cdot W \cdot e + V \cdot e)$ similar method extended recursive model denote concatenation vectors
word context window

Figure 2.2: Equation embedding model [KB18]

EqEmb models is to derive a semantic representation of each equation along with the context of words. EqEmb is based on the idea that a good semantic representation of equations could be discovered by expanding the original word context to include any similar equations that appear in a possibly larger window around it. EqEmb is based on specifically Bernoulli embeddings (B-embs) [Rud+16]. B-emb is an exponential family embedding model where the conditional distribution is a Bernoulli:

$$p(w_i|w_{ci}) = Bernoulli(b_w)$$

where $p(w_i|w_{ci})$ is the conditional distribution of the word w_i given its context w_{ci}. The parameter b_w is defined using the word embedding ρ_{w_i} and the word context α_{w_j} vectors:

$$b_w = \sigma(\rho_{w_i} \sum_{j=1}^{|c_i|} \alpha_{w_j})$$

where σ is the logistic function.

2.4 NEURAL MODELS

Recently an implementation of Long Short Term Memory (LSTM) [HS97] based MIR system significantly outperformed the conventional text search engine. The primary motivation behind this work was that it witnessed the excellent performance of LSTM networks when applied to sequence-to-sequence tasks, such as machine translation, text summarization, entailment prediction, and so on. Secondly, an LSTM based architecture is highly simplified, fast, and effective compared to the complex MIR systems known till date. Even though the role of the LSTM network is less investigated in the context of information retrieval and MIR (in particular), such networks are likely to deliver good results. An LSTM network generates sentence embeddings for IR. The last hidden state of the LSTM network generates a semantic representation for the whole sentence. Automatic keyword detection and the ability to attenuate unimportant words let the network perform document retrieval [PPD19].

The majority of the neural models rely on generating representations for queries and documents for IR. A language-independent Convolutional neural network (CNN) [Yin+17] model captures matching patterns of two natural language sentences. The model also accounts for hierarchical structures of the sentences. Neural Vector Space Model (NVSM) uses the optimization of gradient descent for learning low-dimensional representations for words and documents. The similarity between document and query representations is used to rank the documents. Some weakly supervised neural network models [Deh+17] have been used to rank the search results of information retrieval. Using different input representations, the implemented system is compared with varying ranking architectures.

To represent math formulas in the form of vectors, the document retrieval system of the NTCIR-12 MathIR task uses the Bag of Words version of Paragraph Vector (PV-DBOW), a form of doc2vec algorithm [LM14]. First, the math formulas inside Wikipedia and arXiv documents are preprocessed to remove content MathML and LaTeX encodings and preserve only the presentation MathML encoding. Second, the math formulas are represented in the form of a two-dimensional math expression tree, wherein the non-leaf nodes correspond to operators, and the leaf nodes correspond to operands. Third, the formulas are tokenized and represented in the form of real-valued dense vectors using PV-DBOW. Consequently, the documents containing structurally similar formulas appear closer in vector space.

An architecture for scientific document retrieval, containing three different Text-Text, Text-Math, and Math-Math entailment modules, is proposed in [PS14]. The architecture supports searching for queries that contain text and mathematical contents. Text Entailment (TE) module matches the text of the query to the text contents, Math Entailment (ME) module matches the math part of the query to the math formulas, and the Text Math Entailment (TME) module matches the text part of the query to names of the math formulas.

2.5 FORMULA EMBEDDING APPROACH

A formula embedding approach [PPG18] for MIR transforms math formula into a binary vector, wherein each bit designates the presence of a mathematical entity or its position in a formula. The normalized count of the matching 1-bit positions is indicative of the extent of similarity between two formula vectors. The similarity score guides ranking and retrieval steps. Gao et al. [Gao+17] introduced embedding for both symbols (symbol2vec) and formulas (formula2vec). Symbol2vec was based on a Continuous Bags-of-Words (CBOW) architecture using negative sampling, while formula2vec uses a distributed memory model of paragraph vectors. The preliminary experimental results of "symbol2vec" revealed several characteristics of mathematical formulas that indicated the natural language embedding technologies could be potentially helpful for the formula embedding task. A non-linear formula tokenizer (i.e., tree-based) might be more helpful in finding the appropriate context for a formula token.

A Bidirectional Encoder Representations from Transformer (BERT) based formula embedding model [DPB21] is used to facilitate formula retrieval in ARQMath tasks. To represent the performance of the pre-trained model for mathematical language processing tasks, the proposed model is trained on the math exchange corpus of the ARQMath. The pre-trained bidirectional encoder representations are used from the transformers model for encoding the formula. It takes the LaTeX formulas as input and produces the fixed dimensional embeddings. The embeddings of the formulas the queried formula are compared, and cosine similarity is estimated. For similarity measure, the cosine similarity was used. The performance was tested using the MSE corpus of ARQMath 2020. The obtained results have shown a remarkable contribution to the task of formula retrieval.

A new framework for learning formula representations using tree embeddings [WLB21] was used to facilitate search and similar content retrieval in textbooks containing mathematical formulas. Each symbolic formula (such as a math equation) was represented as an operator tree. Since mathematical formulas are hierarchical in nature, tree structures are an appropriate representation. The symbolic tree representation of the formula has the advantage of encoding both the semantics and the hierarchical structure of a formula. The framework consists of a tree encoder and a decoder, that encodes the formula's operator tree into a vector and vice versa. To improve the quality of the formula tree generation, a novel tree beam search algorithm was developed. The framework was validated on a formula reconstruction task on a real-world dataset of over 900k formulas collected online. The experimental results were promising and significantly outperformed the various baselines.

2.6 DATA ANALYSIS TECHNIQUE

A lattice-based approach [NHC12] for math search was proposed based on Formal Concept Analysis (FCA). It is one of the powerful data analysis techniques. In the approach, math expressions are converted into their MathML representation, to extract math features. These features are used for constructing a mathematical concept lattice [CLN14]. At query time, the query expression is processed and inserted into this lattice. Then the relevant expressions are retrieved and used for ranking purpose. The search results can be visualized by a dynamic graph and can be navigated due to the lattice structure. This approach was benchmarked against the conventional best match retrieval techniques.

2.7 TOPIC MODELS

Topic models are one of the powerful tools to extract the semantic information of the math formula. This uses the technique of multinomial distributions over words. Topic models have been used mainly in scientific articles and documents [Teh+04]. A recent work called TopicEq [YL19] was proposed to incorporate the joint modeling of equations via a Recurrent neural network (RNN). It boosts the performance of the model for scientific texts. Recent work [Cao+15] was proposed for neural topic models that leverages the flexibility and representation power of neural networks.

3 THEORETICAL BACKGROUND

This chapter describes the concepts and theory used for implementing the proposed methods stating their advantages and disadvantages.

3.1 WORD EMBEDDINGS

Word embeddings have been one of the powerful tools widely used in many natural language processing tasks like sentiment analysis [Yu+18], information retrieval, searching, question answering [Zho+16], machine translation [Che+18], etc. Learning high-quality representation of words is extremely important in many NLP applications. Word embeddings are low-dimensional vector representations of words by embedding semantic and syntactic meanings obtained from a large corpus. It has recently gained much traction in various semantic tasks. A wide range of methods for generating such embeddings has been studied in machine learning and knowledge representation literature. The demand for word embedding is growing significantly as extensive NLP tasks emerge in recent times. The following word embedding methods are proposed and categorized based on their application and underlying techniques [Wan+19].

1. **Neural Network Language Model**
   The Neural Network Language Model (NNLM) [Ben+03] together learns a word vector representation along with a statistical language model using a feed-forward neural network that contains a linear projection layer and a non-linear hidden layer. Input to this model is an N-dimensional one-hot vector representing the word where N is the vocabulary size. The input is first projected onto the projection layer. Then a SoftMax operation is performed to compute the probability distribution over all words in the vocabulary. Due to its non-linear hidden layers, the NNLM model is computationally complex. First, an NNLM is trained using continuous word vectors learnt from other simple models to reduce the complexity. Then another N-gram NNLM is trained from these word vectors.

2. **N-gram model**
   The N-gram model is an essential concept in language models. It has been used in many

NLP tasks. The ngram2vec method [Zha+17] incorporates the n-gram model in various baseline embeddings models such as word2vec, GloVe, PPMI, and SVD. The ngram2vec method does word-to-word level co-occurrence and extends the reception window by adding the word-ngram and the ngram-ngram co-occurrence information. Its performance has significantly improved mainly on word analogy and word similarity tasks. It can also learn negative word phrases, which can be difficult for other models.

3. **Word2vec**
   Word2vec [MZ15] is a neural network used for pre-processing the text data and convert into embedding vectors. It includes two learning models, Continuous Bag of Words (CBOW) and Skip-gram. CBOW predicts the word given its context words, and Skip-gram predicts the context given the target word. A text corpus is used as input and Word2vec generates the word vectors. It builds a vocabulary from a training text corpus and learns the vector representations of each word. The similarity between words is calculated using the cosine distance. Word2vec can be used for finding both semantic and syntactic relationships between the words. This model is described in detail in further sections. This method was chosen for our experiment.

4. **FastText**
   Rarely used word embeddings are usually poorly estimated. Hence several methods have been proposed to mitigate this issue. FastText method is an extension of word2vec. FastText methods use the information of sub-word where embedding of rare words can be represented well. It is based on the skip-gram model where each word is represented as a bag of character n-grams or subword units [Boj+17]. Each character n-grams is associated with a vector representation, and the average of these vectors gives the final representation of the word. For example, consider the word "artificial" with n=3, the fastText representation of this word is $\langle ar, art, rti, tif, if, fic, ici, ial, al \rangle$, where the angular brackets indicate the beginning and end of the word. This helps to capture the meaning of shorter words. A skip-gram model is trained to learn the embeddings once the word has been represented using character n-grams. Like mentioned earlier, FastText works well with rare words even if a word was not seen during training; it can be broken down into n-grams to get its embeddings. This model improves the performance on syntactic tasks significantly compared to semantic tasks.

5. **Dictionary model**
   Extracting linguistic properties into word embedding is challenging for more extensive data. Lexical databases such as WordNet help to learn word embeddings, but labeling those databases are time-consuming task. A dictionary is used to describe words to reduce the complexity. The dict2vec [TGH17] method is used to learn word representations from the dictionary entries of the large databases. Semantically-related words from the dictionary appear closer in high-dimensional vector space. It is proven that dict2vec works well on small corpus. Here, negative sampling is used to filter out odd pairs. Dict2vec models the information by building the pair of strong and weak words to provide a positive and negative sampling.

3.2 WORD2VEC EMBEDDING MODEL

Word2vec [Mik+13b] was created and published in the year 2013 by a team of researchers led by Tomas Mikolov at Google. Generation of word embedding in word2vec occurs in two ways: Continuous Bag-of-Words (CBOW) and Skip-Gram (SG). Though both use the same structure of the network, they rely on different input and output variable management [Ben+03]. These models are unsupervised models that take in massive textual corpora to create a vocabulary. Then generate the dense word embeddings for each word in the vector space. The performance of word2vec varies with the number of epochs. The behavior of the learning rate depends on each epoch. CBOW predicts the current word based on the given context, i.e., neighboring words in the window. It contains three layers: the input layer corresponds to the context: the hidden layer corresponds to the projection of each word from the input layer into the weight matrix; the output layer is projected from the third weight matrix. The final step is the comparison between its output and the input word. Based on the backpropagation of the error gradient [NCG17], correctness can be achieved. Considering an example sentence, "the quick brown fox jumps over the lazy dog.", this can be pairs of (*context_window*, *target_word*) with a window size 2, the pairs are ([quick, fox], brown), ([the, brown], quick), ([the, dog], lazy) and so on. Thus the model tries to predict the *target_word* based on the *context_window* words [Figure 3.1].

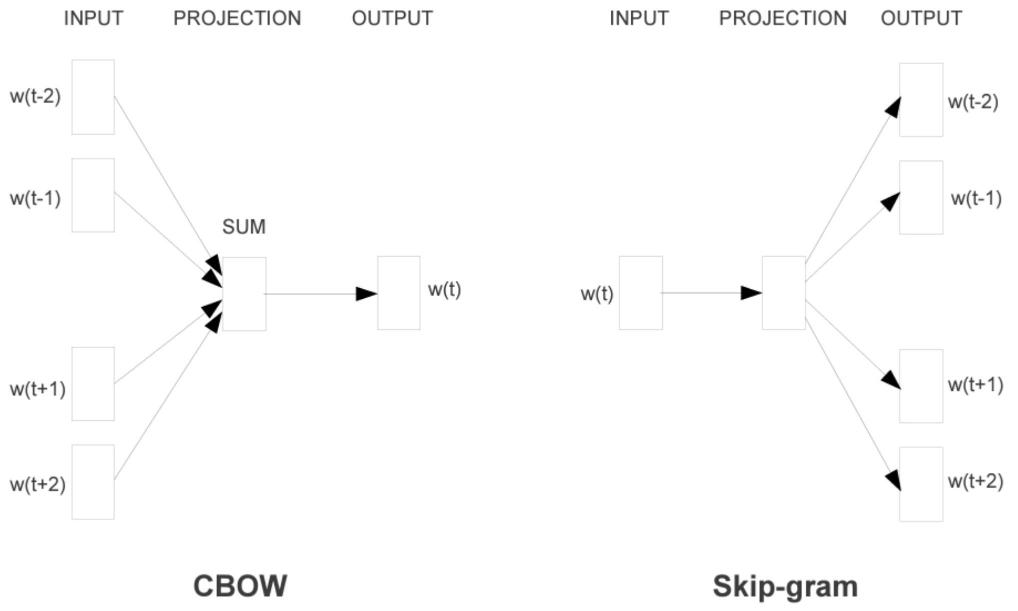


Figure 3.1: Word2vec architecture [Mik+13b]

SG is the opposite of the CBOW models. Here, the input layer corresponds to the target word, and the output layer corresponds to the context. Therefore, Skip-Gram predicts the context given the target word. The final step is the comparison between its output and each word of the context. Considering the simple sentence from earlier, "the quick brown fox jumps over the lazy dog." The task is to predict the context [quick, fox] given the target word 'brown' or [the, brown]. Thus, the model predicts the *context_window* words based on the *target_word* [Figure 3.1].

Each of these models has its advantages. SG model is more efficient for small corpus and also for rare words. On the other hand, CBOW works well for larger corpus and frequent words.

However, learning the output vectors from these two methods can be complex and challenging. To mitigate this, a negative sampling algorithm can be used. The main idea of this algorithm is to limit the number of output vectors that need to be updated. Thus, only a sample of these vectors is updated based on a noise distribution, a probabilistic distribution used in the sampling process. Negative sampling is more efficient with low dimensional vectors, and it works better with frequent words [NCG17].

The following are some of the important parameters in training of word2vec model:

1. **Context window:** The size of the context window is used to determine the numbers of words before and after a given word. The recommended value for skip-gram is ten and for CBOW it is five.

2. **Dimensionality:** Usually, the dimensionality of the vectors is between 100 and 1,000. The quality of word embeddings increases with higher dimensionality. After a threshold value, marginal gain diminishes.

3. **Subsampling:** High-frequency words often provide less information. Therefore words with a frequency above a certain threshold are usually subsampled to speed up the training process.

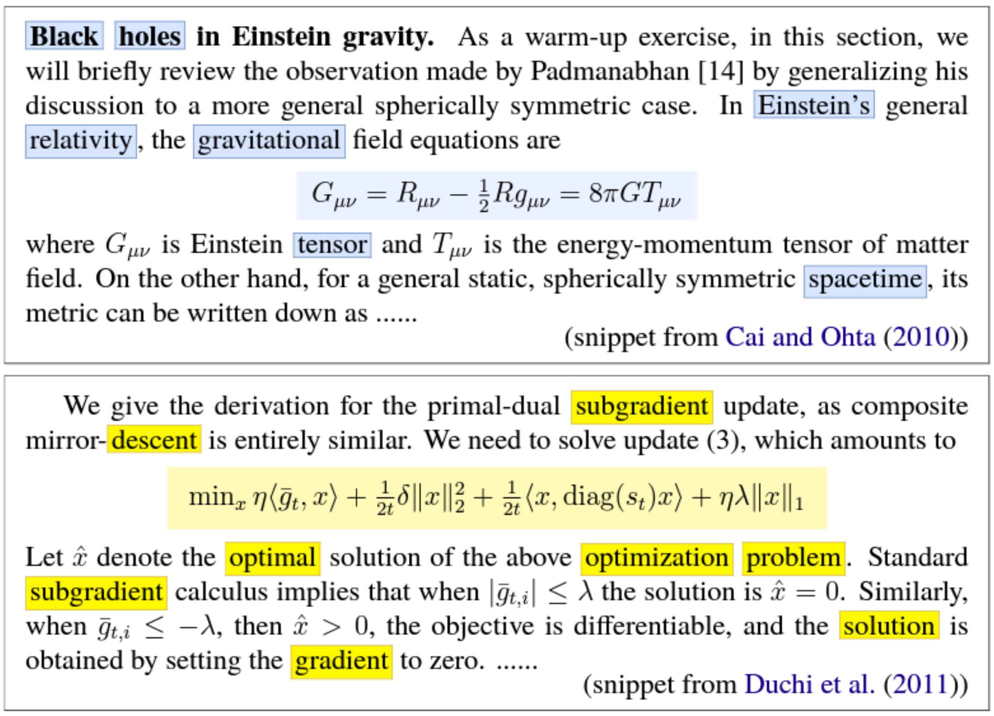
4. **Training algorithm:** Word2vec model can be trained with hierarchical softmax and/or negative sampling. The hierarchical softmax method uses Huffman coding [Mof19] to reduce calculation. The negative sampling method approaches the maximization problem by minimizing the log-likelihood [OAM20] of sampled negative instances. Usually, hierarchical softmax works better for rare words while negative sampling works better for frequent words.

5. **Training samples:** Prime task is to generate training data. The target word is referred to as a central word. The neighboring words are called context words based on the context window. The data pairs are generated, where each pair is composed of the central word and a context word. The total number of unique words from the corpus is considered as this determines the size of the embedding vocabulary. This constitutes the training sample.

6. **Neural network:** The architecture of the feedforward network is simple. The input layer takes a word w and returns its one-hot encoded vector representation, x. One-hot encoded vectors are vectors with all zeroes but 1 one, and they are used to indicate one member in a set. For each pair, w is the central word considered. The size of the input vector V is the same as the vocabulary size. There is no activation function in this layer. There is a matrix of V rows and E columns in the hidden layer that assigns to each input word x its embedded representation h. Each of the i-th rows represents its embedding, and the output is a vector of size E. No activation function is used. The output layer is the opposite of the previously mentioned network. It is a transposed matrix of E rows and V columns, in which the i-th column represents the embedding of the i-th context word. Given an embedding h, the multiplication performed with the mentioned matrix returns a V-sized vector y. An activation function, called softmax is applied to y. A vector z is generated, where the k-th element indicates its probability in the context of w.

3.3 JOINT EMBEDDING METHOD

The proposed model jointly embeds equations and their context text (surroundings words) in posts of ARQ Math challenge data set (MSE data set) and demonstrates that the model can effectively retrieve relevant formulas for a given query formula. The intuition behind the model is illustrated in the sample passages in Figure 3.2, which shows how the topic of the word context is often indicative of the distinctive types of equations used, and vice versa. For instance, equations appearing in the topic of relativity (with context words like "back hole", "Einstein") tend to involve a series of tensors like G and T. While equations used in the topic of optimization (context words "gradient", "optimal") may use norms like the min operator and often their combinations. Ideally, the strings of mathematical symbols in the equations should aid the training of models, and the context words should aid the modeling and understanding of the equations. For the


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

$$G_{\mu\nu} = R_{\mu\nu} - \frac{1}{2}Rg_{\mu\nu} = 8\pi G T_{\mu\nu}$$

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

(snippet from Cai and Ohta (2010))

We give the derivation for the primal-dual subgradient update, as composite mirror-descent is entirely similar. We need to solve update (3), which amounts to

$$\min_x \eta\langle \bar{g}_t, x\rangle + \frac{1}{2t}\delta\|x\|_2^2 + \frac{1}{2t}\langle x, \mathrm{diag}(s_t)x\rangle + \eta\lambda\|x\|_1$$

Let $\hat{x}$ denote the optimal solution of the above optimization problem. Standard subgradient calculus implies that when $|\bar{g}_{t,i}| \leq \lambda$ the solution is $\hat{x} = 0$. Similarly, when $\bar{g}_{t,i} \leq -\lambda$, then $\hat{x} > 0$, the objective is differentiable, and the solution is obtained by setting the gradient to zero. ......

(snippet from Duchi et al. (2011))

Figure 3.2: Top topic: Relativity; bottom topic: Optimization. [YL19]

joint embedding model, the Translational vector model [MLS13] was used. It is mainly used for language translations. The method consists of two simple steps. First, monolingual models of languages are built using large text corpus. Then, a small bilingual dictionary learns a linear projection between the languages. During testing, any word can be translated that has been seen in the monolingual corpora by projecting its vector representation from the source language space to the target language space. Once the vector is obtained in the target language space, the most similar word vector is the output. The similarity distances between vectors are calculated via the cosine distance. The models learn word representations using a neural network that predicts the context of a word. Due to its simplicity, the SG and CBOW models can be trained on a large text data: word2vec implementation can train the model to learn from billions of words in hours. When trained on a large dataset, they capture a substantial amount of semantic information.

Consider a set of word pairs and their associated vector representations $x_i, z_{i=1}^n$, where $x_i \epsilon R^{d_1}$ is the distributed representation of word i in the source language, and $z_i \epsilon R^{d_2}$ is the vector representation of its translation. The main goal is to find a transformation matrix W [MLS13] such that Wx_i approximates z_i. In practice, W can be learned by the following optimization problem, which is solved using stochastic gradient descent [Rob07]. The gradient is computed using the backpropagation rule [RHW86].

$$min_W \sum_{i=1}^{n} ||Wx_i - z_i||^2$$

At prediction time, for any given word along with its continuous vector representation x, it can be mapped to the another language space by computing $z = Wx$. Then the word that is closest to z in the target language space is found, using cosine similarity.

4 DATA

This chapter describes about the ARQMath Challenge, dataset used for training and testing phase in the proposed approach. It also focuses on experimental setup, scripts and gold dataset used for evaluation.

4.1 ARQMATH CHALLENGE

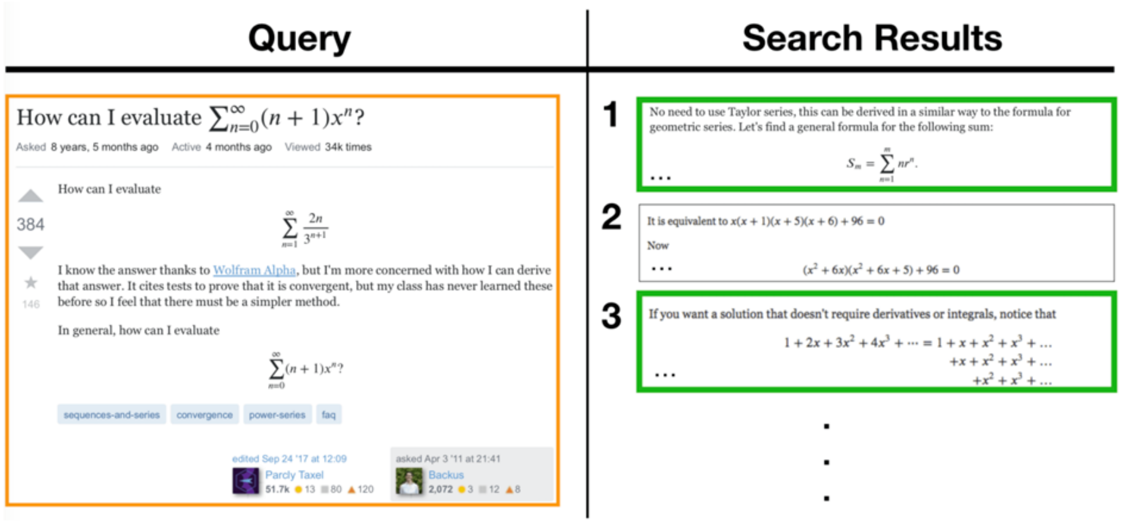
As math is intrinsically tricky, people usually consult forums like Community Question Answering (CQA) sites such as MSE[1] (MSE) and Math Overflow[2] to find answers to their math questions. There is evidence [Zan+20] that people search more math queries on conventional search engines compared to general queries. This gives the opportunity to develop a math-based search engine that processes formulas and keywords effectively to provide more accurate and better results. Finding answers to mathematical questions is one of the challenges in the field of AI that has gained a lot of attention and limelight among researchers in the IR and NLP communities. This has led to a lot of research in the growing community of researchers working on MIR. To address this, the ARQMath evaluation exercise was introduced, aiming to search advanced math-related queries to analyze and understand the semantics of mathematical notation and texts. This platform is helpful to develop computational methods to support math-related answers to humans. ARQMath has two tasks, (i) answer retrieval and (ii) formula retrieval. However, the tasks are challenging because both questions and answers can be a combination of natural language and mathematical language involving words, terms, sentences, and formulas.

Existing models for formula retrieval can be broadly classified as text-based, tree-based, and embedding-based models [Zan+20]. Text-based models use the representation of string for formula indexing and retrieval. Tree-based models use hierarchical/tree representation(s) of formula appearance and operator syntax using paths traversed. Embedding-based models use vector space to project the formulas and are also helpful in identifying neighboring formulas based on closer appearing formulas in the vector space. It mainly uses the bag- of-words approach or

[1]`https://math.stackexchange.com/`
[2]`https://mathoverflow.net/`

skip-gram approach to embed equations composed of variables, operators, and other symbols to convert into equivalent vectors.

Figure 4.1 illustrates the first task in ARQMath 2020 and Figure 4.2 illustrates the second task. The task is mainly in the domain of mathematics involving formulas, but it could be extended to a different domain (e.g., chemistry or biology), which employs other types of special notation. The questions and answers for the task are taken from the platform of MSE, an online platform with a host of Q&A forums. It is one of the largest and most trusted online communities that provide its content publicly available in XML format for developers to learn and share their knowledge. It roughly consists of around 150+ Q&A communities in different fields like computer queries, math, physics, etc. Users can rank questions and answers according to their quality assessment by voting them up or down. For ARQMath tasks, the collection from MSE comprises Q&A postings extracted from data dumps from the Internet Archive. For task 1, rather than generating new answers, answers provided by a community are selected and ranked. For task 2, an individual formula is used as the query, and the system returns a ranked list of other potentially useful instances of formulas found in the collection. Here, the relevance is determined by the usefulness of a retrieved formula based on user's knowledge. Formulas were returned by their identifiers within the collection, along with their associated post identifiers. Finally, the results are analyzed using a manual consistency and quality check [Man+21].



How can I evaluate $\sum_{n=0}^{\infty}(n+1)x^n$?

Asked 8 years, 5 months ago Active 4 months ago Viewed 34k times

How can I evaluate

$$\sum_{n=1}^{\infty} \frac{2n}{3^{n+1}}$$

I know the answer thanks to Wolfram Alpha, but I'm more concerned with how I can derive that answer. It cites tests to prove that it is convergent, but my class has never learned these before so I feel that there must be a simpler method.

In general, how can I evaluate

$$\sum_{n=0}^{\infty}(n+1)x^n?$$

No need to use Taylor series, this can be derived in a similar way to the formula for geometric series. Let's find a general formula for the following sum:

$$S_m = \sum_{n=1}^{m} nr^n.$$

...

It is equivalent to $x(x+1)(x+5)(x+6)+96=0$

Now

... $(x^2+6x)(x^2+6x+5)+96=0$

If you want a solution that doesn't require derivatives or integrals, notice that

$$\begin{aligned} 1+2x+3x^2+4x^3+\cdots &= 1+x+x^2+x^3+\ldots \\ &\quad +x+x^2+x^3+\ldots \\ &\quad +x^2+x^3+\ldots \end{aligned}$$

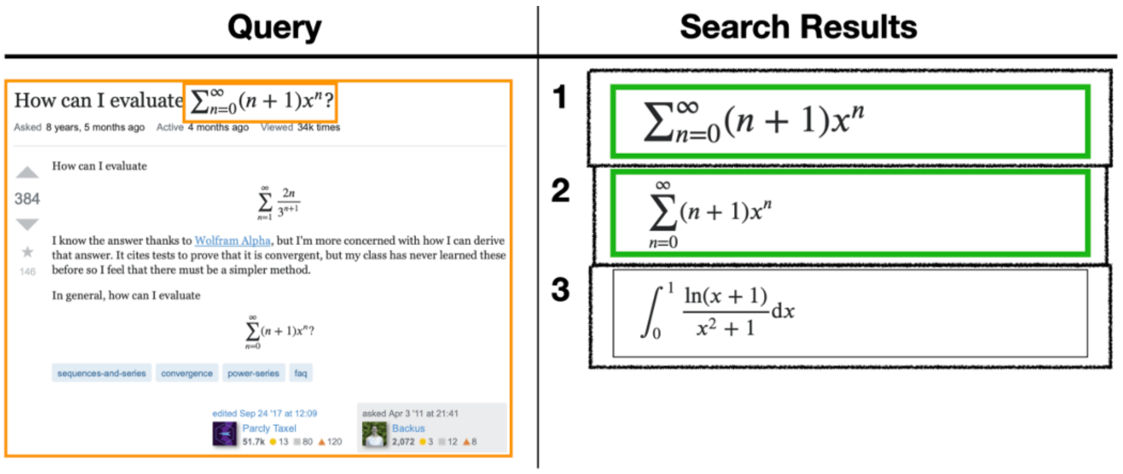
...

Figure 4.1: Task 1: Answer Retrieval - Query indicates the question and Search Results indicate the answers to the question [Zan+20]

4.2 ARQMATH DATA COLLECTION

The ARQMath data collection contains posts of questions and answers from Math Stack Exchange (MSE), that are published from the year 2010 to 2018, with a total of about 1 million questions and 1.4 million answers. In ARQMath 2020 Challenge, posts from 2019 were used for the construction of topics. The dataset collection is available on Google Drive[3]. The collection contains the following:

[3]`https://drive.google.com/drive/folders/1ZPKIWDnhMGRaPNVLi1reQxZWTfH2R4u3?usp=sharing`


Query
Search Results
1
$\sum_{n=0}^{\infty}(n+1)x^n$
2
$\sum_{n=0}^{\infty}(n+1)x^n$
3
$\int_0^1 \frac{\ln(x+1)}{x^2+1}dx$
384
146
sequences-and-series
convergence
power-series
faq

How can I evaluate $\sum_{n=0}^{\infty}(n+1)x^n$?

Asked 8 years, 5 months ago Active 4 months ago Viewed 34k times

How can I evaluate

$$\sum_{n=1}^{\infty}\frac{2n}{3^{n+1}}$$

I know the answer thanks to Wolfram Alpha, but I'm more concerned with how I can derive that answer. It cites tests to prove that it is convergent, but my class has never learned these before so I feel that there must be a simpler method.

In general, how can I evaluate

$$\sum_{n=0}^{\infty}(n+1)x^n?$$

edited Sep 24 '17 at 12:09 Parcly Taxel 51.7k ● 13 ■ 80 ▲ 120

asked Apr 3 '11 at 21:41 Backus 2,072 ● 3 ■ 12 ▲ 8

Figure 4.2: Task 2: Formula Retrieval [Zan+20]

1. **Posts**: Each MSE post has a unique identifier and a field to indicate whether it is a question or answer post. Questions have a title and a body (the content of the question), while answers have only a body. Each answer has a 'parent_id' indicating the answer to its related question. Additional information such as the score, the post owner id, and creation date are also available. A snapshot of *posts.xml* file is shown below:

```
<posts>
 .....
  <row
    Id="7"
    PostTypeId="2" (1: Question, 2: Answer)
    ParentId="5" (present only if PostTypeId is 2)
    Body="<p>You use a proof by contradiction. Basically, you suppose that
    <span class="math-container" id="63">\sqrt{2}</span> can be written as
    <span class="math-container" id="64">p/q</span>. Then you know
    that <span class="math-container" id="65">2q^2 = p^2</span>.</p> "
    CommentCount="10"
    CreationDate="2010-07-20T19:21:52.240"
    AcceptedAnswerId="17"
    AnswerCount="7"
    OwnerUserId="45"
    Tags="<real-analysis> <algebra-precalculus> <decimal-expansion>"
    ViewCount="40630"
    Score="74" />
   ....
</posts>
```

2. **Comments**: MSE users can comment on posts. Each comment has a unique identifier and a 'post_id' indicating for which post the comment was written.

3. **Post links**: Sometimes, there may be duplicate or related questions that are identified previously. A 'post_link_type_id' of value '1' indicates related posts, while value '3' indicates duplicates.

4. **Tags**: Questions can have one or more tags describing the domain of the question.

5. **Votes**: Each vote has a 'vote type id' for the vote type and a 'post_id' for the associated post. The post score is the difference between up votes and down votes, and there are also other vote types such as 'offensive' or 'spam'.

6. **Users**: Registered MSE users have a unique id. Each user has a reputation score, which can be increased either by posting an appropriate answer or by receiving up votes to the posted question.

7. **Badges**: Each user can receive a list of badges. There are three classes of badges, namely 1-gold, 2-silver, 3-bronze.

4.2.1 Test data set

Formulas for Task 2 were selected based on three criteria: complexity, elements, and text dependence.

Formulas: In MSE, the mathematical formulas are located under the $\langle span \rangle$ tag where class attribute is set to "math-container" and also as LaTeX string placed between single or double '$' signs. Overall, 28,320,920 formulas were detected and annotated. Every identified instance of a formula was assigned a unique identifier, FID as shown below:

$$\langle span\ id = FID\ class = "math - container" \rangle ... \langle /span \rangle$$

In the formula retrieval task, around 98 mathematical formulas are provided from the questions of Task 1 in the form of Topics[4] (topics.xml) and expected to find the relevant formulas from answers posts in the collection. Each topic is located in a $\langle Topic \rangle$ tag with a unique topic id. All topic ids are in the format of "B.x", where x shows the topic number. There is a correspondence between topic id in tasks 1 and 2. For instance, the topic id "B.99" indicates that the formula is selected from the topic "A.99" in task 1 and so on.

Each topic has 5 parts:

1. $\langle Formula\_Id \rangle$: id of formula query.
2. $\langle Latex \rangle$: LaTeX representation of formula query.
3. $\langle Title \rangle$: question title of the post from which the formula query is selected.
4. $\langle Question \rangle$: question body from which the formula is selected.
5. $\langle Tags \rangle$: comma-separated tags of the question.

The snapshot of the *topics.xml* file is as shown:

[4] `https://drive.google.com/drive/folders/1DFvfNObb1T8AnOYkCvpOo6XmfX-9J60B`

```
<Topics>
 .....
 <Topic number="B.17">
      <Formula_Id> q_97 </Formula_Id>
      <Latex> \int _{x=0}^{\infty} \frac{\sin(x)}{x} </Latex>
      <Title> Calculate with the function </Title>
      <Question>
      ......
      </Question>
      <Tags> complex-analysis,improper-integrals </Tags>
 </Topic>
   ....
</Topics>
```

4.3 GOLD DATASET

Gold dataset (also known as qrel file) strictly adheres to the Text REtrieval Conference (TREC) qrel format [5], which contains a set of human assessed documents for each query in the query set. Gold dataset has the format as shown below:

Query ID	Iteration	Formula ID	Relevance

where, Query ID indicates a specific query, Iteration is an irrelevant field which is set to 0, and ignored by the trec eval tool; Formula ID specifies post ID of the retrieved formula, and Relevance designates relevance score. A formula is judged by the human assessor as relevant, even if it contains a sub expression of the query formula. The generation of this file is described in Experiment setup chapter.

[5]https://trec.nist.gov/data.html

5 METHODS

This chapter describes the data preprocessing and proposed methods used to retrieve relevant ranked list of formulas from the dataset.

5.1 DATA PREPROCESSING

For preprocessing, the official tool provided by ARQMath was applied to read the posts from the posts.xml [4.2] file. The file contains several rows within $\langle row \rangle$ tag. From each $\langle row \rangle$ tag , *Body*, *PostTypeId* and *Id* attribute values are extracted. The *Body* contains the question or the answer depending on the *PostTypeId* (1 = question, 2 = answer), and *Id* presents the unique post id. In *Body*, the mathematical formulas are located under the $\langle span \rangle$ tag where class attribute is set to "math-container" and also as LaTeX string placed between single or double '$' signs. The text and formulas are extracted from *Body* and stored in a TSV file. The TSV file mainly contains id, question, question_context_words (question_CW), answer, and answer_context_words (answer_CW). As discussed earlier, if the PostTypeId is 1, the formula and text are inserted in the question column; else, it was inserted in the answer column. The context window size is set to ten, i.e., five words before the formula and five words after the formula. The stop words are removed using the standard NLTK[1] library. The stopwords are the most common words in data. These words are not useful to describe the topic of the content. Examples of stop words in English are "a", "the", "is", "are", "such", and etc. The generated TSV file [Table 5.1] is the input to the proposed models. The models were trained from scratch with this input file.

ID	question	question_CW	answer	answer_CW
5	\sqrt{2}	['proofs', 'irrational', ..., 'square', 'root']	p^q	['proof', 'contradiction.'..., 'written', 'p/q']
57	R \rightarrow T	['natural', 'Symmetric', ..., 'fraction', 'relation']	xRy	['reflexive', 'relation',..., 'finite', 'count']

Table 5.1: Input TSV file

[1] https://www.nltk.org/

Three variations of formula embedding methods are proposed. They are described in detail as follows:

1. Formula embedding model using tokens,
2. Formula embedding model using whole formula as words,
3. Joint embedding model using Translational vector model.

5.2 FORMULA EMBEDDING MODEL USING TOKENS

The system architecture of the proposed method is as shown in Figure 5.1. The proposed system architecture is inspired by the existing word2vec [Mik+13b] formula embedding approach where each module works interdependently to make the faster retrieval and accurate search.

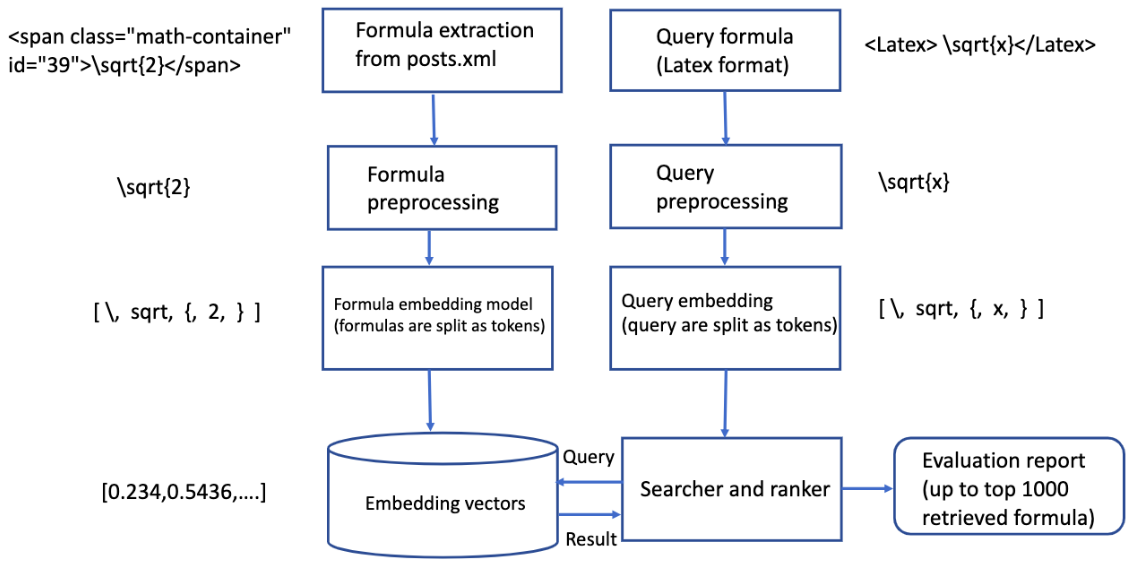

<span class="math-container" id="39">\sqrt{2}</span>
Formula extraction from posts.xml
Query formula (Latex format)
<Latex> \sqrt{x}</Latex>
\sqrt{2}
Formula preprocessing
Query preprocessing
\sqrt{x}
[\, sqrt, {, 2, }]
Formula embedding model (formulas are split as tokens)
Query embedding (query are split as tokens)
[\, sqrt, {, x, }]
[0.234,0.5436,....]
Embedding vectors
Query
Result
Searcher and ranker
Evaluation report (up to top 1000 retrieved formula)

Figure 5.1: Proposed architecture for formula embedding model using tokens

5.2.1 Formula extraction

The formulas are extracted from the posts of MSE from 2010-2018 in the LaTeX format. To investigate the ability of the proposed approach in the formula search task, only the formulas in LaTeX format are used. Some examples of the formulas are shown in Table 5.2.

post	**formula**
<span class="math-container" id="39">\sqrt{2}</span>	\sqrt{2}
<span class="math-container" id="67">p^2</span>	p^2
<span class="math-container" id="151">R=\frac{Q}{D}</span>	R=\frac{Q}{D}

Table 5.2: Examples of LaTeX format formulas extracted from MSE post

5.2.2 Formula Preprocessing

The prime task of the formula preprocessing module is to transform the formulas into a unified form by removing irrelevant elements and attributes. In this process, the preprocessing module trims the irrelevant tags like *span* and attributes like *class* before storing in the TSV file. The formulas from the column ′question′ and ′answer′ will be taken from the input TSV file [Table 5.1] and are split into valid LaTeX tokens using *tokenize.generate_tokens()*[2] method of Python library. This method reads each input line of text and returns 5-tuples with following members: the token type; the token string; a 2-tuple (s_row, s_col) of integers specifying the row and column where the token begins in the source; a 2-tuple (e_row, e_col) of integers specifying the row and column where the token ends in the source; and the line on which these tokens were found (input line). While splitting the tokens, irrelevant punctuation, white spaces were removed. For example, if the formula is $F(n)^2$, the tokens are ′F′, ′(′, ′n′, ′)′, ′∧′, ′2′. These tokens are stored in a list.

5.2.3 Formula embedding

The tokens are fed as input to the word2vec model to generate the respective embedding vector. Each token is represented as a vector.

5.2.4 Query preprocessing

The prime task of the query preprocessing module is to extract the formulas from the Topics file 4.2.1 and remove unwanted tags. The query formula is represented in the $< Latex >$ tag. The topic_id, formula_id, and the LaTeX formula are extracted and stored in the TSV file as shown in Table 5.3. The LaTeX formulas are split into valid tokens and stored as lists and is used for testing.

topic_number	formula_id	latex
B.1	q_4	f(x)= \frac{x^2 + x + c}{x^2 + 2x + c}
B.1	q_9	\frac{df}{dx} = f(x+1)
B.1	q_22	\sum_{k=0}^{n} \binom{n}{k} k

Table 5.3: Some example of extraction done from topics file

5.2.5 Query Embedding

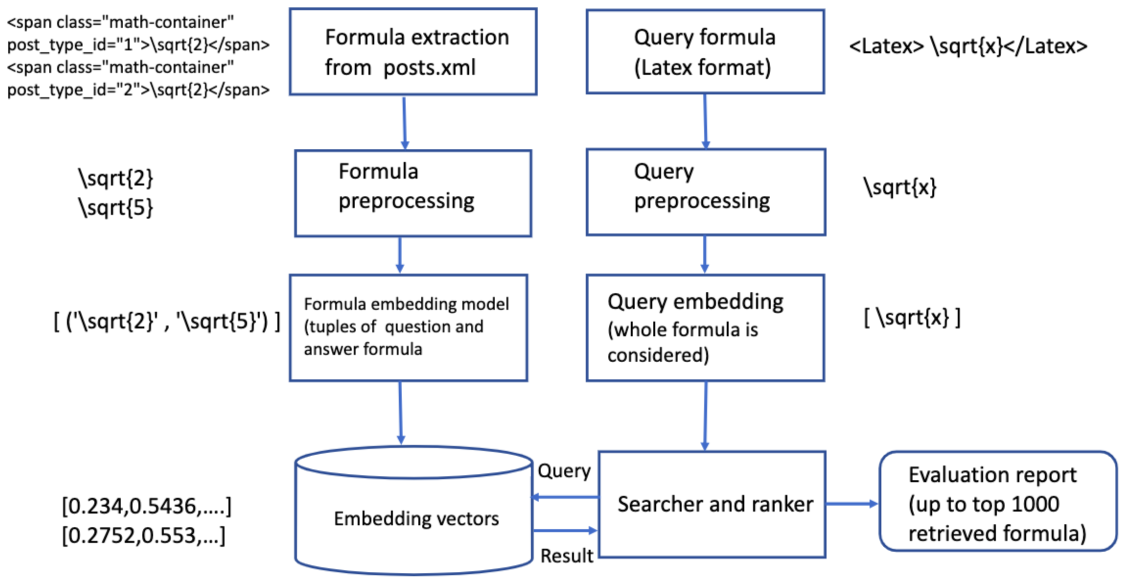
Query embedding module converts the preprocessed LaTeX tokens list into a fixed size vector using the word2vec model in testing phase.

[2] https://docs.python.org/3/library/tokenize.html#module-tokenize

5.2.6 Searcher and Ranker Module

The main objective of the searcher module is to search for relevant formulas that are syntactically or semantically similar to the query formula. Searching for relevant information is a time-consuming process and requires effective formula representation. In the proposed architecture, using the multiprocessing technique, the searcher module compared the query formula vector with all the input formula vectors and computed each formula vector's cosine similarity. For similarity calculation, the proposed approach is compared with combined query vectors with the combined formula vectors. After a successful comparison and similarity calculation between the formula vectors and query vector, the ranker module retrieves and ranks the post (the post which contains the search formula) based on the higher similarity score. The MSE posts are assigned higher priority if it contains more similar formulas as that of query formula. As a final search result to query formula, at most top 1000 posts of MSE are retrieved, which contains the relevant formulas for each query formula.

5.3 FORMULA EMBEDDING MODEL USING WHOLE FORMULA AS WORDS



<span class="math-container" post_type_id="1">\sqrt{2}</span>
<span class="math-container" post_type_id="2">\sqrt{2}</span>
Formula extraction from posts.xml
Query formula (Latex format)
<Latex> \sqrt{x}</Latex>
\sqrt{2}
\sqrt{5}
Formula preprocessing
Query preprocessing
\sqrt{x}
[('\sqrt{2}' , '\sqrt{5}')]
Formula embedding model (tuples of question and answer formula
Query embedding (whole formula is considered)
[\sqrt{x}]
[0.234,0.5436,....]
[0.2752,0.553,...]
Embedding vectors
Query
Result
Searcher and ranker
Evaluation report (up to top 1000 retrieved formula)

Figure 5.2: Proposed architecture for formula embedding considering whole formula

The system architecture of the proposed method is as shown in Figure 5.2. This architecture is inspired by considering the whole formulas as words and these words are used for training the word2vec model. The Formula extraction module and Query preprocessing module are same as described in Section 5.2.

5.3.1 Formula Preprocessing

In this module, the formulas from the 'question' and 'answer' columns are retrieved from the input TSV file that are stored as a list. The main difference compared to earlier method is that instead of feeding the model with tokens of formula split, the whole formula is considered as words to generate embeddings. The list contains tuples of question and answer pairs which were used to train the model. For example, the list contains [('$/sqrt2$', 'p / q'), ('$/sqrt2$', '$2q^2 = p^2$'), ('$/sqrt2$', '$2q^2$'), ...]. This kind of tuple based representation is used because intrinsically word2vec models are always trained as tuple of target word and context word in textual data for better prediction.

5.3.2 Formula embedding

The list of tuples are fed as input to the word2vec model to generate the respective embedding vector. Each formula is represented as a vector as shown in Figure 5.3.

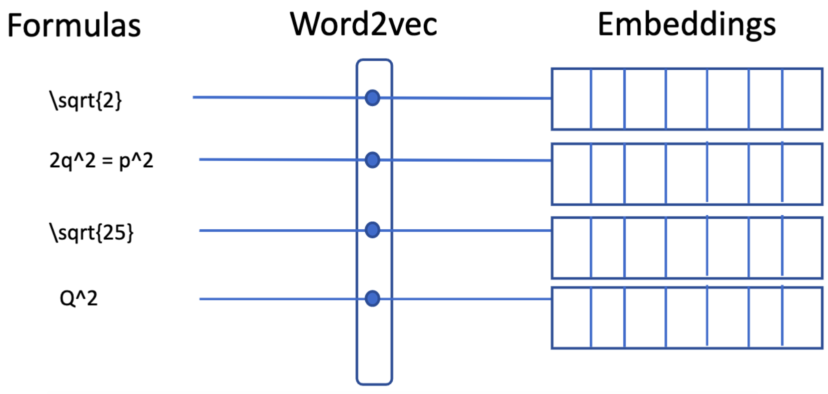


Figure 5.3: Formula to vector

5.3.3 Query Embedding

Query embedding module converts the whole LaTeX formula into a fixed size vector using the word2vec model.

5.3.4 Searcher and Ranker Module

In the proposed architecture, using the multiprocessing technique, the searcher module compares the query formula vector with all the input formula vectors and computes the cosine similarity between them. The formulas with a maximum similarity score are returned. After a successful comparison and similarity calculation, the ranker module retrieves and ranks the post containing the whole formula based on the higher similarity score rather than tokens as compared to previous method.

5.4 JOINT EMBEDDING MODEL USING TRANSLATION VECTOR MODEL

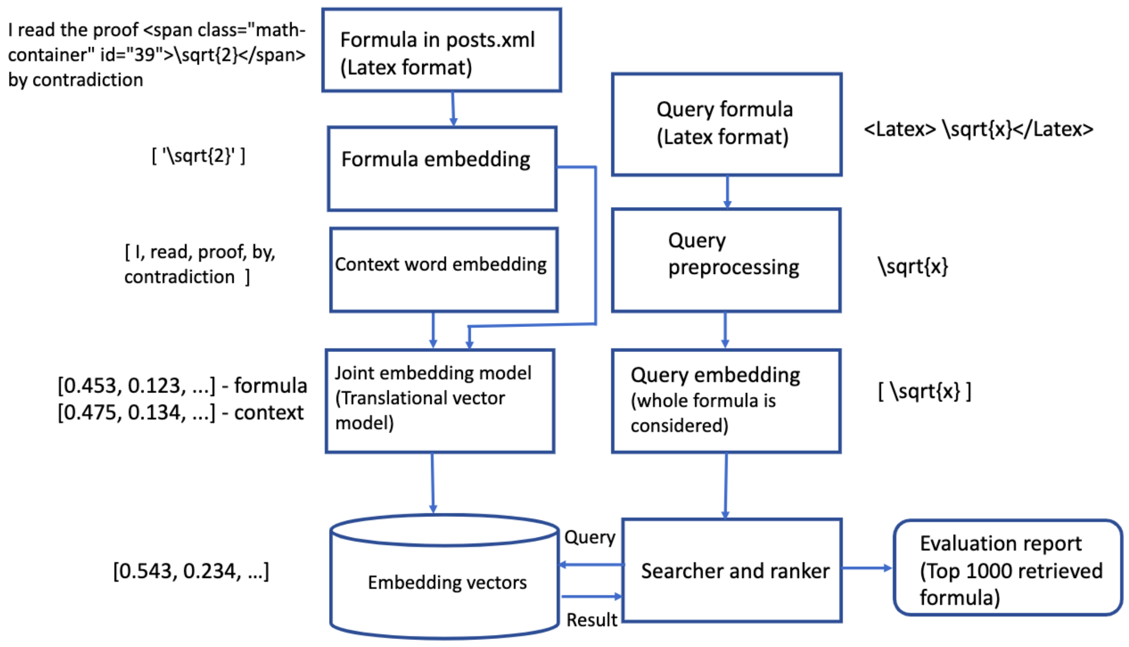

I read the proof <span class="math-container" id="39">\sqrt{2}</span> by contradiction
Formula in posts.xml (Latex format)
['\sqrt{2}']
Formula embedding
[I, read, proof, by, contradiction]
Context word embedding
[0.453, 0.123, ...] - formula
[0.475, 0.134, ...] - context
Joint embedding model (Translational vector model)
[0.543, 0.234, ...]
Embedding vectors
Query formula (Latex format)
<Latex> \sqrt{x}</Latex>
Query preprocessing
\sqrt{x}
Query embedding (whole formula is considered)
[\sqrt{x}]
Query
Result
Searcher and ranker
Evaluation report (Top 1000 retrieved formula)

Figure 5.4: Proposed architecture for joint embedding method using formula and context words

The system architecture of the proposed method is as shown in Figure 5.4. The proposed system architecture was inspired by considering both formulas and their context words. The formulas and context words were trained independently by separate word2vec models. These embeddings were combined by joint embedding method using the Translation vector model. The Formula extraction module and Query preprocessing module are same as described in Section 5.2.

5.4.1 Formula Preprocessing

In this module, the formulas from the ′question′ and ′answer′ columns were retrieved from the input TSV file that are stored as a list. The context words from the column ′question_CW′ and ′answer_CW′ were also retrieved and stored as a list. Both lists were used as inputs to train the word2vec model separately.

5.4.2 Formula embedding

In this module, only formulas are fed as input to the word2vec model as explained earlier in Section 5.4. Each formula is represented as a vector.

5.4.3 Context words embedding

In this module, only the context words retrieved from the input file are fed to a separate word2vec model. Each context word is represented as a vector.

5.4.4 Joint embedding model

The vector representation of formulas and context words are combined to generate a new vector representation using the Translation vector model. It builds the translation matrix from mapping the formula model's vector and the context word model's vector. The Translation Matrix works by assuming the two entities share similarities in the geometric arrangement in both vector space. With this assumption, the algorithm learns a linear mapping from the one vector representation to the another vector representation. During translation, it project the input word to be translated into the target space, and returns words with a representation closest in the target space. In this module, it combines formula embedding vector and context vector based on cosine similarity, to measure their closeness in the same vector space. Once the model is trained, it is tested with query embedding model. Then using cosine similarity, the formulas that are closet to the query formulas are retrieved.

5.4.5 Query Embedding

The query embedding module converts the whole LaTeX formula into a fixed size vector using the word2vec model.

5.4.6 Searcher and Ranker Module

In the proposed architecture, using the multiprocessing technique, the searcher module compares the query formula vector with all the input joint embedded vectors and computes the cosine similarity between them. After a successful comparison and similarity calculation, the ranker module retrieves about 1000 similar formulas. It ranks the post (containing the whole formula) based on the higher similarity score.

6 EXPERIMENTAL SETUP

This chapter mainly focuses on experimental setup, evaluation metrics, methods used by other participating teams in ARQMath Challenge, configurations, hyperparameters, scripts, and tools used for evaluation.

6.1 EVALUATION METRICS

Ranking plays a vital role in the Formula Retrieval task. In general, users inspect only the top few results returned by a query search and therefore precision and relevance factors strongly determines the ranking function. When the model retrieves the candidates for a query formula, correctly ranking the results will reduce the number of invalid or irrelevant formulas. The ranking criterion is based on the cosine similarity of formulas with respect to the query, and it contributes to determine the relevance score of the results. The ranked list of relevant formulas are evaluated based on three metrics: nDCG′, mAP and p@10.

6.1.1 Normalized Discounted Cumulative Gain (nDCG')

The primary measure for the task is $nDCG'$ measure (read as "nDCG-prime") introduced by Sakai and Kando [SK08]. It is a measure of ranking quality. In IR, the effectiveness of search engine algorithms is measured using this. The lists of search results vary depending on the query. A search engine's performance from one query to the next cannot be achieved by DCG alone. Hence the cumulative gain at each position p should be normalized across queries. This can be achieved by sorting all relevant documents by their relative relevance. Then, produce the maximum possible DCG through its position p, called Ideal DCG ($IDCG$). The $nDCG$ for each query is computed as:

$$nDCG_p = \frac{DCG_p}{IDCG_p}$$

where $IDCG$ is ideal discounted cumulative gain,

$$IDCG_p = \Sigma_{i=1}^{REL_p} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

and REL_p represents the list of relevant documents (ordered by their relevance) in the corpus up to position p. The average performance of a model's ranking algorithm can be obtained by averaging the $nDCG$ values for all queries.

6.1.2 Precision at k (p@10)

The number of relevant results from the top 10 retrieved documents is considered. Precision is defined as the ratio of the retrieved documents that are relevant to user's query over the retrieved documents. Precision formula for information retrieval [Tan21] is as shown below:

$$precision = \frac{|relevant\ documents| \cap |retrieved\ documents|}{|retrieved\ documents|}$$

By default, precision considers all the retrieved documents for calculation. But, it can be evaluated at a given threshold number, known as cut-off rank. The model is only assessed by considering only its top-most query results. This measure is called p@k.

6.1.3 Mean Average Precision (mAP)

mAP for a set of queries is measured as the mean of the average precision scores for each query. It is one of the popular metrics used to measure the performance of models in IR tasks. It is also used to quantify the performance of the model in the query search tasks.

$$mAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q}$$

where Q is the number of queries and $AP(q)$ is the average precision (AP) for a given query, q. Area under the precision-recall curve measures AP. But, for practical purpose, it is a finite sum over every position in the ranked sequence of documents given as,

$$AP = \sum_{k=1}^{n} P(k)\Delta r(k)$$

where k is the rank in the sequence of retrieved documents, n is the total number of retrieved documents, $P(k)$ is the precision at cut-off k in the list, and $\Delta r(k)$ is the change in recall from items $k-1$ to k.

6.2 METHODS USED BY OTHER PARTICIPATING TEAMS

The evaluation of the proposed model was tested against three different teams' results from the ARQ Math challenge. They are Baseline (Tangent -S), Document and Pattern Recognition Lab from the Rochester Institute of Technology (DPRL Systems) [MOZ20a] and Ensembling Math Information Retrieval Systems for Math Information Retrieval group at Masaryk University (MIRMU).

6.2.1 Baseline

Tangent-S [DZ17] is considered as baseline. It is the extended model of Tangent-3 [Zan+16a]. It works with OPTs and uses a stricter unification model to avoid matching functions to variable names. Tangent-S derived the first two stages from Tangent-3, which uses a two-stage Symbol Layout Tree (SLT) model for formula retrieval. First, top-k candidates are identified using a CBOW model, using symbol pairs in SLTs as 'words'. Later, the top-k candidates are re-ranked after aligning query and candidate SLTs. Candidates are re-ranked using the harmonic mean of symbol and relationship recall (the Maximum Subtree Similarity) [CWW21] and two tie-breakers: symbol precision after unification and symbol recall without unification. An additional third stage is added, where a linear combination of similarity scores is computed from the second layer to produce a final re-ranking of the top-k candidates. SLTs and OPTs are retrieved independently to obtain stronger formula retrieval results and then linearly combined with their similarity scores. This supports the view of OPTs (Operator trees) and SLTs as complementary for formula retrieval.

6.2.2 DPRL Systems

They built three systems and submitted one run from each. All three methods use formula embeddings produced from strings generated after parsing formulas with Tangent-S [DZ17], which has both SLT and OPT representations, namely Tangent-CFT, Tangent-CFT-Plus, Tangent-CFT with Tree Edit Distance.

1. **Tangent-CFT:** Tangent-CFT (Tangent Combined with fastText) [Man+19] is an embedding model for mathematical formulas. This model uses both SLT and OPT representations of formulas to consider the visual appearance and the syntax of formulas. To embed mathematical formulas, mathematical formulas are first linearized, and then a text embedding model is applied. Mathematical formulas are usually represented in LaTeX or MathML format. Using Tangent-S [DZ17], these are converted into SLT and OPT encodings, and a depth-first traversal is used to generate a tuple sequence. After converting to a tuple sequence, each tuple can be considered a word. 'Words' and 'characters' are defined using an n-gram embedding model. Each character (token) is encoded using a unique identifier. Tokenizing the tuple elements can provide good insight into the formula structure. Then, use the fastText [Boj+17] n-gram embedding model to embed the formula. Each encoded tuple is considered a word, and the context window for a tuple T is defined by nearby tuples in the linearized tuple sequence. The context window size is also a hyperparameter in the model, set to the

default value, 5. After the model is trained, each tuple is assigned a d dimensional vector defined before training. The vector representation for formula F with set of n tuples,TF, is given by:

$$formulaVec(F) = \frac{1}{n} \sum_{t \epsilon T_f} tupleVec(t)$$

The similarity of the two formulas is computed using the cosine similarity of their vector representations.

2. **Tangent-CFT-Plus:** Tangent-CFT Plus [MOZ20b] is an extension to the Tangent-CFT embedding model where only the structural similarity of formulas are considered during embedding. To decide the relevance of the formula, context words play a vital role. Each formula is represented as the concatenation of Formula and Text vectors. Therefore, each formula has two vector representations:

   (a) Formula Vector: Vector representation of formulas obtained by Tangent-CFT. The vector size is 300.

   (b) Text Vector: Vector representation of the surrounding words of the formula that are trained by the fastText model. The vector size is 100 by default.

3. **Tangent-CFT with Tree Edit Distance:** From the experiment results on the NTCIR-12 Wikipedia Formula Browsing task test collection [Zan+16b], it was found that Tangent-CFT did better on partial matching compared to full matching. Tangent-CFT was extended to re-rank the retrieval results based on tree-edit distance to address this issue. Tree-edit distance (TED) [PA16] is the minimum cost of converting one tree to another. Three edit operations that are considered here are insertion, deletion, and substitution. These operations can have different weights when calculating the tree-edit distance cost. The ranking score for each retrieved formula is calculated as:

$$sim(T_1, T_2) = \frac{1}{TED(T_1, T_2)}$$

The weights for each edit operation were turned over the range [0, 1] with step size 0.05. Since there are two tree representations for formulas (OPT and SLT), the re-ranking is done on each representation separately, and then the results were linearly combined as :

$$Score_q(f) = \alpha \cdot SLT\_Score_q(f) + (1 - \alpha) \cdot OPT\_Score_q(f)$$

with constant α = 0.95.

6.2.3 MIRMU

They prototyped and submitted results for three primary systems, Formula2Vec, SCM, and Ensemble. They opted to prototype systems that primarily tackle the math representation problem because the accuracy of solving the task of formula retrieval lies in the correct representation of formulas and context text. They preprocessed the data and represented it as arXiv to learn the joint representations of text and math. They compared several math formulas representations and used unsupervised approaches for representation learning. At last, they ensembled primary submissions into a committee of MIR systems that achieved better performance on task 2 results compared to the individual ensembled method. They used two sets of relevance judgments, automatic for parameter estimation and model selection, and human-annotated for performance estimation [Nov+20]. To measure information retrieval accuracy for parameter optimization, model selection, and performance estimation, they used the $nDCG'$ and Spearman's ρ [08], a general non-parametric measure of rank correlation.

6.3 GENSIM MODELS

Genism[1] is a open-source Python library used to perform large-scale NLP tasks. It helps to represent natural language words as semantic vectors and find similar scientific documents. It is the fastest library available for the training of vector embeddings. The core algorithms in Gensim use highly optimized and parallelized C routines. It can process arbitrarily large corpora using data-streamed algorithms. Genism was the best choice since there was a limitation of the hardware and could not use any high-performance computing machine due to time constraints. It is platform-independent and could be tested on Linux or Windows. With thousands of companies using Gensim every day, over 2600 academic citations, and 1M downloads per week, Gensim is one of the most popular ML libraries. The Gensim community also publishes pre-trained models for specific domains like legal or health, etc. All Gensim source code is hosted on Github under the GNU LGPL license, maintained by its open-source community. Hence easy to install and access.

6.4 CONFIGURATION

Processor	1.8GHz dual-core Intel Core i5
Memory	8GB of 1600MHz LPDDR3 onboard memory
Storage capacity	128GB SSD
Startup Disk	Macintosh HD
Graphics	Intel HD Graphics 6000

Table 6.1: System Configuration

Genism's word2vec and translational vector model are used in the implementation. Word2Vec is one of the widely used algorithms based on neural networks. For this experiment, large amounts of unannotated math text from the ARQ Math Challenge dataset are used. The intention was that word2vec learns relationships between words and formulas automatically. The output is vectors where each vector represents a word, that gives remarkable linear relationships to compare similar formulas based on cosine similarity metrics [SW15].

The three proposed methods as described in Chapter 5 were implemented in Python using PyTorch[2] IDE and Jupyter[3] notebook. The system used for the experiment was MacBook Air (13 inches, 2017 model), with the configuration as shown in Table 6.1.

[1] https://radimrehurek.com/gensim/index.html
[2] https://pytorch.org/
[3] https://jupyter.org/

6.5 RELEVANCE SCORE

Relevance can be defined as the usefulness for the searcher who searches for a given formula query. The search engine may retrieve one or more instances of the given query formula. The relevance score is defined in four levels to judge the usefulness of the retrieved formula, as shown in the table below.

Score	Rating	Definition
3	High	As good as finding an exact match to the query formula
2	Medium	Useful but not as good as the original formula
1	Low	Some chance of being useful
0	Not Relevant	completely different than the query formula

Table 6.2: Relevance scores

6.6 HYPERPARAMETERS SETTINGS

6.6.1 Negative sampling

To accelerate the training of the word2vec model, negative sampling was introduced as optimization for word2vec [MLS13]. While predicting a word with a certain window size (e.g. 5), the neighbouring words in the window are considered as positive words. However, it is also necessary to have negative cases to reduce the number of weight matrices, thereby reducing the training time and having a better prediction result. Therefore, the non-surrounding words (referred as negative word) beyond the window size are used as negative cases. For example, if there are ten positive words and five negative words for predicting one target word, then the total number of neuron weight updating operations is eleven (10 positive word + 1 target word) instead of updating whole corpus's neuron weight. Using negative sampling, the calculation of the loss function is only performed for a subset of all entries. This speeds up the calculations and results in a smaller number of elements that need to be adjusted in the weight matrices of the hidden and output layer of the neural network. The size of negative sampling is set to three [Mik+13a].

6.6.2 Embedding size

The dimensionality of the word vector increases the computational complexity with increase in the size of training dataset. For better results, embedding size ranges from 300-1,000 [Mik+13a]. In this experiment, it is set to 500.

6.6.3 Minimum count (w_{min})

These are words that are to be included in vocabulary only if its occurence is greater than configured w_{min} value, else they are replaced by a token 'UNK' for unknown. In this experiment, w_{min} is set to one.

6.6.4 Epochs

The number of epochs controls the total number of times model goes over each item of the dataset, which has a direct impact on the duration and the quality of the training. The choice often depends on the constraints imposed by the size of the training dataset, the computational complexity and available resources. In our experiment, the number of epochs was set to ten to minimise the training loss, but due to hardware constraints, larger epoch values were not considered.

6.6.5 Window size

Increasing the size of context window has been reported [Jan17] to boost accuracy through a larger number of training samples, but also increases training time. The window size is usually sampled randomly for each training sample with probability of $\frac{1}{C}$, where C is the maximum context window size. It signifies the wide gap between two items in a sequence, such that they are still considered in the same context. The default value is set to 5 in Gensim [HER+10]. Some authors claim that it is best to use a "infinite" window size [BK16], meaning that the whole session is considered as one context. In our experiment, the context window was set to ten (five words before and after the target formula) so that enough context words are taken into consideration during training and evaluation.

6.6.6 Subsampling

Higher-frequency items are randomly sub-sampled. Subsampling the frequent words reduces the number of training examples [Mik+13a]. For each frequent word in the training text, there is a chance that it will be effectively deleted from the text. The default value of the sub sampling parameter is set to 10^{-3} in Gensim [HER+10] and the same value is retained in the experiment.

6.7 RUN FILES

Run file indicates the output file of the proposed methods. According to the evaluation protocol of ARQMath Challenge, runs for each proposed methods should be submitted for evaluation by all the participating teams. Submitted runs can be either manual or automatic. In a manual run, manual intervention will produce results for query formulation and result selection. In an automatic run, results must be produced algorithmically by computer, without user intervention.

The runs are evaluated using the Python scripts as described below, and the results are displayed in the form of TSV with evaluation metrics 6.1.

The submitted run file should be in tab-separated variable (TSV) format, and their names should use the following convention:

```
[group]-[task]-[run-type]-[data-used]-[eval].tsv

    * [group]: CLEF-registered team name.
    * [task]: task1 / task2
    * [run-type]: manual / auto (manual or automatic)
    * [data-used]: text / math / both (text-only, formula-only, formula and text)
    * [eval]: P / A (primary run, alternate run)


Examples:   TeamX-task1-auto-both-P.tsv
            TeamY-task1-auto-text-A.tsv
            TeamZ-task2-manual-math-P.tsv
```

The content of each run file submitted should follow the guidelines of ARQMath 2020 Protocol[4]. The retrieval results submitted by the each participating team have the following columns:

Query_Id	Formula_Id	Post_Id	Rank	Score	Run_Number
B.1	282	24312	1	0.98	Run_0
B.1	398	7623	2	0.95	Run_0
B.1	405	8508	3	0.81	Run_0

Table 6.3: Example of run file

The *Query_Id* is the id of each query formula from the *topics.xml* as discussed in Section 4.2.2. The *Formula_Id* is the id of the retrieved formula and the *Post_Id* is the id of the question or the answer in which the retrieved formula appears. Note that the formulas should be in a question (title and/or body) or an answer in the *posts.xml*. *Rank* is generated in increasing order up to 1000 because there are 1000 retrieved formulas. The *Score* is the normalized similarity measure that is usually float in nature. Finally, the *Run_Number* is a trivial field where a string that characterizes system run is indicated; since there are three methods proposed, it will be named as Run_0, Run_1, and Run_2 respectively.

6.8 PYTHON SCRIPTS USED FOR EVALUATION RESULTS

There are two python scripts to generate the evaluation results. The file *"de_duplicate_2020.py"* generates the prime-deduplicated results for task 2. Deduplication is elimination of duplicate or redundant information from the file. The redundant formulas are eliminated using pooling

[4]https://drive.google.com/drive/u/0/folders/1IQDVkBbJO4IhUMSjgs5uDx5KPlOcdEDr

and annotation methods described in Section 6.9. *"task2_get_results.py"* is used to get evaluation measures from trec_val tool.

de_duplicate_2020: This script is used to generate the deduplicated results for the submissions in task 2. The four inputs should be provided as the example command shown below:

```
python3 de_duplication_2021.py
    -qre "qrel_task2_2020.tsv"
    -tsv "/latex_representation_v1/"
    -sub "/ARQMath 2020 Submission/Task2_2020/"
    -pri "/ARQMath 2020 Submission/Task2_2020_Prime/"
```

The first input is the qrel file provided on google drive[5]. The second input is the directory in which the latex TSV files are located. The third input is the directory in which the all participating teams runs for Task 2 are located. The last input is the directory in which the de-duplicated prime results get stored.

task2_get_results: After creating the de-duplicated result files, this script can be used to calculate the effectiveness measures. Please note that nDCG′ and mAP and p@10 are calculated on the deduplicated files. Here is the sample command:

```
python3 task2_get_results.py
-eva "/usr/local/bin/trec_eval"
-qre "qrel_task2_all.tsv"
-pri "Task2_2020_Prime/"
-res 2020_task2_results.tsv
```

The first input is the path to the trec_eval tool. The second is the path to the qrel file. The third input is the directory in which the prime files are located. The last entry is the file path to save the final results.

6.9 HUMAN ASSESSMENT

The gold dataset (qrel file) described in section 4.3 is generated using pooling and annotation methods.

6.9.1 Pooling

Pooling was done based on visually distinct formulas. For each query formula in Task 2 of ARQMath 2020 Challenge, the participants will retrieve up to d related formulas from the answers and its value ranges upto 1000. The pooling is applied to the top-k results ($k < d$) from all the participants as shown in Figure 6.1.

[5] `https://drive.google.com/drive/u/0/folders/1BKk_Q7wKtoezRlfIb1OcoWCiUYuVuwsx`

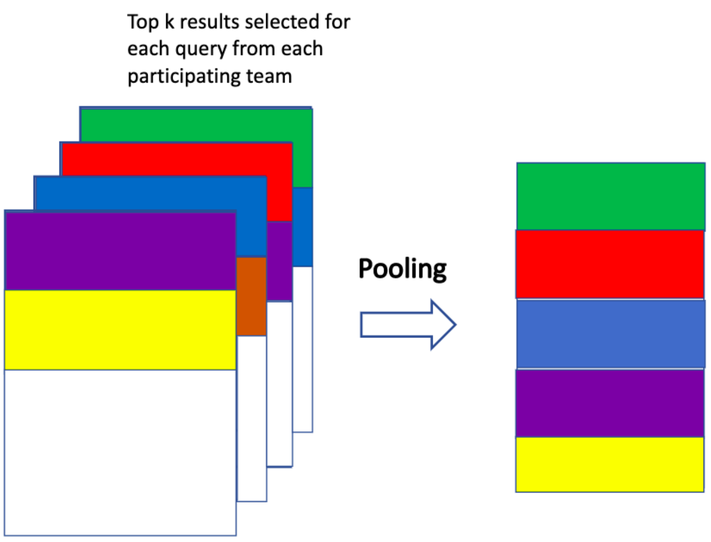

Top k results selected for each query from each participating team
Pooling

Figure 6.1: Creating a pool for a given query from the results

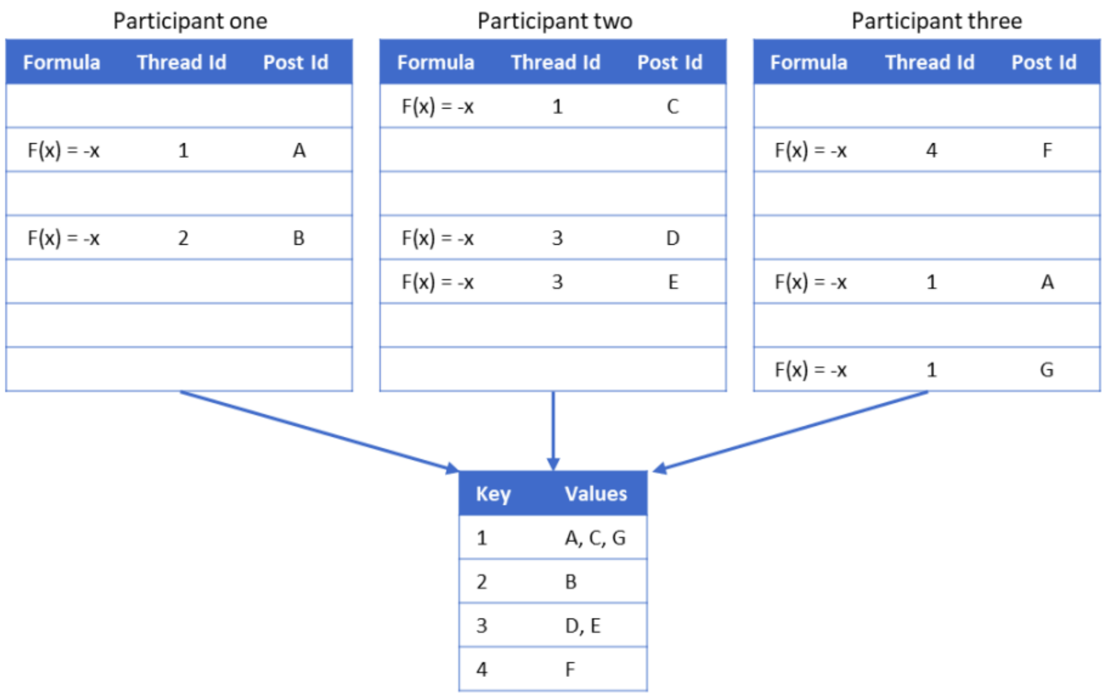
6.9.2 Annotation

After pooling, the next step is annotation. For better understanding consider an example of a given query formula $F(x) = x$ to which, formula such as $F(x) = -x$ is retrieved by different participants. Consider this formula, $F(x) = -x$ is retrieved from a thread (either question or answers) with $id = 1$ by a participant. In contrast, another participant may retrieve the same formula from a different thread with $id = 99$. Therefore, if the formula $F(x) = -x$ is in the pool, a dictionary is created where the keys are the thread ids and the values are the list of answer ids. To decide if the retrieved formula is related to the query, a maximum of N different answers are chosen as shown in Figure 6.2. The choosing criteria are as follows:

1. If the number of answers is less or equal than N, they will be all judged by the annotators.
2. If the number of answers is more than N, but the number of threads is less than N, first from each thread, one answer will be randomly chosen to be judged. Then the rest of the answers are selected randomly.
3. If the numbers of answers and threads are more than N, then N unique threads are chosen randomly, and a random answer is selected for each thread.

To annotate each formula, Turkle[6] tool is used by the annotators as shown in Figure 6.3. Turkle is a locally installed system with functionality similar to Amazon Mechanical Turk[7]. All the retrieved formulas in the pool are shown on the left side of the tool. The formula query is highlighted as shown in Figure 6.3 and its related question is located below it. The question gives a good description of the relevance of the formula. The annotators will look at each formula (highlighted), along with their answers. The retrieved formula is categorized using the four relevance score as described in Section 6.5. To determine the relevance score, the annotator will consider the formula query along with its question and the context of the answer of the retrieved formula.

[6]`https://github.com/hltcoe/turkle`
[7]`https://www.mturk.com/`



Participant one

Formula	Thread Id	Post Id
F(x) = -x	1	A
F(x) = -x	2	B

Participant two

Formula	Thread Id	Post Id
F(x) = -x	1	C
F(x) = -x	3	D
F(x) = -x	3	E

Participant three

Formula	Thread Id	Post Id
F(x) = -x	4	F
F(x) = -x	1	A
F(x) = -x	1	G

Key	Values
1	A, C, G
2	B
3	D, E
4	F

Figure 6.2: Creating a map for annotation from the pool

Apart from the four levels of relevance score, two additional choices were given. 'System failure' indicates system issues such as wrong rendering of formulas or the thread link not working. 'Do not know' indicates that the assessors were not able to decide the relevance degree. The assessors had to leave a comment for these special cases.

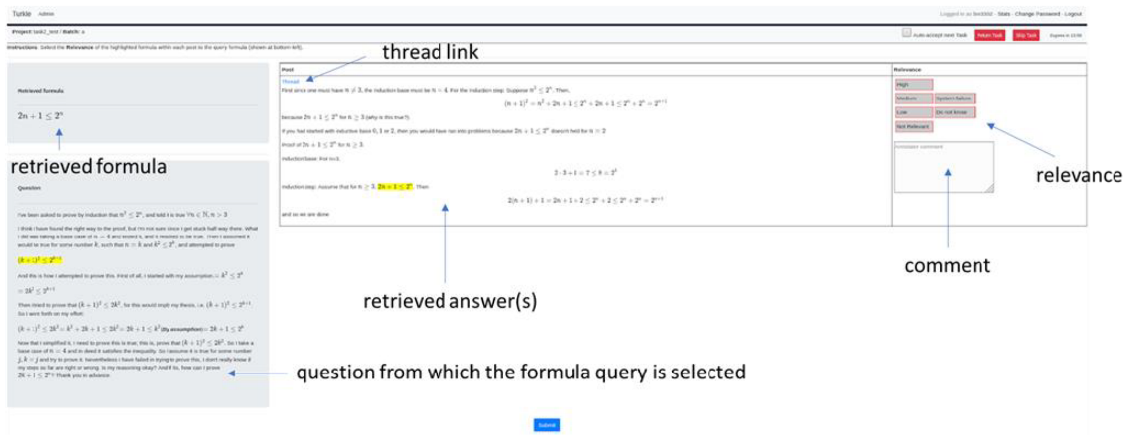


Figure 6.3: Annotation tool

7 EVALUATION

This chapter describes the evaluation, results, and analysis of the three proposed methods on task 2 of the ARQMath 2020 challenge. The implementation code is available in Github[1]. The results are compared with the other participating teams in the ARQMath Challenge.

7.1 OUTPUT FILES

Three output run files were used for evaluating the three proposed methods. They were declared as automatic, meaning that queries were automatically processed from the *topics.xml* file. During testing, the ranked lists for each query were produced with no human intervention. Following are the three run files generated from the proposed approaches for Task 2 of ARQMath Challenge 2020.

7.1.1 Thesis-task2-JointEmbedding-auto-both-P

According to the evaluation protocol of ARQMath 2020 Challenge, the naming convention of the file is followed as described in Section 6.7. This is the primary run file output of the joint embedding method where the context words and formula vectors are combined using a translational vector model as described in Section 5.4. The cosine similarities of the query's vector and all the trained vectors in the collection are computed which is used to rank the results for each topic.

7.1.2 Thesis-task2-tokens-auto-math-A

This alternate run is similar to the primary run, with the difference being that the formula vectors are the vector representation of the tokens of the formula as described in Section 5.2. The formulas

[1] `https://github.com/vidya0c/JointEmbedding`

are split as valid LaTeX tokens and then combined to find relevant formulas. The cosine similarity is calculated between the query formula vector and all combined tokens vector which is used to rank the results for each topic.

7.1.3 Thesis-task2-formulas-auto-math-A

This alternate run is similar to the primary run, with the difference being that the whole formula vector is used for training as described in Section 5.3.For each topic, the cosine similarity is calculated between the query formula vector and all formula vectors which is used to rank the results for each topic.

7.2 RESULTS

For evaluating the performance, the trectool[2] is used, which compares the gold dataset (Qrel file) with the results obtained from the proposed systems. The results for the formula retrieval task are shown in Table 7.1.

Participating Systems	**nDCG′**	**mAP**	**p@10**
Baseline_TangentS_Res_Task2	0.4939	0.2744	0.4553
DPRL-task2-CFTED-auto-P	0.4018	0.2469	0.4809
DPRL-task2-CFT-auto-A	0.3753	0.2096	0.3787
MIRMU-task2-SCM-auto-math-P	0.1308	0.054	0.0553
DPRL-task2-CFTPlus-auto-A	0.1288	0.0447	0.1979
MIRMU-task2-Formula2Vec-auto-math-P	0.103	0.0448	0.0723
MIRMU-task2-Ensemble-auto-math-A	0.0946	0.0314	0.0489
Thesis-task2-tokens-auto-math-A	**0.0939**	**0.0309**	**0.0468**
Thesis-task2-JointEmbedding-auto-both-P	**0.0725**	**0.0264**	**0.0426**
Thesis-task2-formulas-auto-math-A	**0.0541**	**0.0156**	**0.0447**

Table 7.1: Results of proposed methods and participating teams in ARQMath 2020

The results of the all the participants are ranked based on obtained nDCG′ metric. The results are also estimated for generic and more comparative analysis in terms of mAP and p@10. Higher value of metrics indicates the better performance of the models. The proposed model was trained with around 2 million formulas with training time of about 8 hour and 34 minutes with the mentioned configuration of the hardware in Section 6.4. The highlighted rows in the above table are the results of the three proposed methods. From the results, it can be seen that among three, *Thesis-task2-tokens-auto-math-A* performed better compared to the primary run, *Thesis-task2-JointEmbedding-auto-both-P* among all metrics.

The results of the baseline system have achieved the highest score compared to all the participant

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

teams. The obtained evaluation results discloses that the proposed approach requires significant improvement compared to the baseline system for formula search and retrieval tasks.

Topic Id	Query formula	Retrieved Formulas
B.14	y = xy′+ \frac{1}{2}(y′)^{2}	frac{1}{2}y′+(2x+y′) = y y = xy′+\frac{2}{3} y = y′+ \frac{1}{2} (t′-2x)t′=2t-2x^2 x^2+y=t

Table 7.2: Best relevant retrieved search results

Overall, the metric indicates the *token based* approach performed better than the other two methods. But for some topics, the performance of *joint embedding* method was better than the *token based* method. The *whole formula* method always performed poorer for all topics.

Best retrieved results are the most relevant formulas to the query formula when ranked by maximum similarity score. For instance, the joint embedding method produced the best formula retrieval results for topic B.14 as shown in Table 7.2. It retrieved the most similar matched formulas to query formula that are mostly syntactically identical formulas. This shows the positive impact of the formula embedding method in the retrieval of mathematical information.

Partial retrieved results are the results where some parts of the formulas were matched to the query formula. In this case, the methods using tokens (*Thesis-task2-tokens-auto-math-A*) performed better compared to other two methods. For instance, the results for topic B.24 using the tokens method are shown in Table 7.3. This indicates that the token based approach has good potential to retrieve the partial match formula. It was found that the joint embedding method also retrieved less number of partial match formulas which indicates the low scores compared to token based approach for same topics.

Topic Id	Query formula	Retrieved Formulas

B.24	\sqrt{2i-1}	b=\sqrt{4}+2i
		\sqrt{2}+\sqrt{5}
		\sqrt{5}
		a^2=b^{-2}
		2i-1 = 2xy+x

Table 7.3: Partially relevant retrieved search results

Irrevalant results: The frailty of the *whole formula embedding* approach (*Thesis-task2-formulas-auto-math-A*) for topic B.30 is shown in table 7.4, which indicates that it sometimes retrieves the totally irrelevant results. The low scores of the metrics for this method is explained by the retrieved irrelevant formulas compared to other two methods.

Topic Id	Query formula	Retrieved Formulas
B.30	a^3+b^3+c^3-3abc	(a)\1 a,b 3+2i \frac{3i} b = (c \sin(18) - a)^{3}

Table 7.4: Irrelevant retrieved search results

7.3 ANALYSIS

Three evaluation measures were considered for the formula retrieval task: nDCG′, mAP, and p@10. Our results for the task were not promising compared to the baseline, and other participants results. The *joint word embeddings* method was implemented by training a word2vec model on text and math in the LaTeX representations separately and combined using the translational vector model as described previously. The main aim is to produce a highly sparse similarity matrix between text and math terms to align in similar vector space to improve retrieval accuracy and speed.

Here, we have comprehensively analyzed proposed methods' strengths and possible limitations from different perspectives. The following are the observations based on system outcomes of the proposed methods:

1. The first observation is that the primary method *joint embedding method*, does not outperform the other runs. Instead, the alternative run *token based* where the formulas are split as LaTeX tokens has performed comparatively well in all evaluation measures.
2. It was observed that, given a query formula, the proposed models retrieve all search results, wherein the query term is either present in its exact form or present as a sub-formula. This happens because the similarity score (matching embeddings of query vector and the search result) for such results will be maximum.
3. It is also observed that the search results may neither contain exact query formula for few queries. Still, they are retrieved by the models, because the results semantically resemble the query formula.
4. Generally, the word2vec model are used in words and textual data extraction. In our case, word2vec model was trained from scratch with formulas. Formulas are naturally ambiguous. Formulas are context-dependent; they can come in various notations depending on the context. The goal is to retrieve all forms irrespective of the notations. Even though they may be semantically the same but syntactically different, word2vec model would have failed to detect this. For instance the formula $x^2 + y^2 = z^2$ and the formula $z = \sqrt{(x^2 + y^2)}$ are semantically same and are considered as relevant formulas. But the word2vec model did not detect the semantics behind the formula and failed to return as relevant formulas. For

instance, word2vec may not distinguish the variants $\frac{n}{d}$, $n : d$ and n/d of a fraction, even though they are applications of the division function of n and d.

5. In MSE, Pythagoras theorem is written as $a^2 + b^2 = c^2$, some may choose to write the same theorem using different variables like $x^2 + y^2 = z^2$. These are semantically same, but model failed to detect this relevance.

6. The poor performance of the models could also be due to the small number of epochs (ten) used during training. Due to hardware limitation, small values of epochs were used during the experiment.

7. In some cases, a search result with a low similarity score might prove to be more relevant than the one with a better score. For example, the similarity score for topic B.24 in Table 7.3 was lower compared to topic B.30 in Table 7.4. This shows that even if the scores are low, better results was retrieved. In this case, the embedding vectors are not generated as intended.

8. Translational vector models are mainly used in many language translation tasks. Interestingly, it showed promising results in fetching sub-formulas to query formulas. This is a novel method that attempted to see the performance of existing models used for joint embedding formulas. This opens the door to explore more on the formula embedding domain.

8 CONCLUSION

This thesis report described an unconventional formula embedding approach called joint embedding, which can ease retrieval of math formulas present in technical or scientific documents. This formula embedding approach opens the door to new horizons and helps combat underlying challenges of scientific document retrieval. The project aimed to retrieve a ranked list of relevant formulas for Task 2 of the ARQMath 2020 Challenge. A formula retrieval task is mainly to retrieve individual formulas from question posts that are located in posts of 2010 to 2018. The relevance is determined by considering the context of the post from which a query formula is taken, and the posts in which retrieved formulas appear. In this project, a total of three methods were proposed. Word2vec model from genism is used as the primary model in all three methods. Hyperparameters like context window, sub sampling, negative sampling, etc were considered in the implementation. The joint embedding method is used for combining the pre-processed formula vectors and the context word vectors using translational vector model. This embedding is used to retrieve the relevant formulas as that of the query formula. There were two other alternative to this primary method. One method was to use the whole formula is used for training the model. The other method was splitting the formula into valid LaTeX tokens, then adding the individual vectors to form the combined formula embedding. All these methods uses cosine similarity measure to retrieve formulas. The retrieved formulas were ranked based on the relevance score. The results were evaluated against the gold dataset (qrel dataset) using the metrics of nDCG′, mAP and p@10. Comprehensive analysis of the model results affirms the underlying strengths and weaknesses of it. The proposed methods showed the competence in retrieving exact query formulas, parent formulas, sub-formulas and similar formulas. This project was a novel approach of using translational vector model for formula embedding. This work gives a lot of scope for improvement further in the field of math formula embedding domain.

8.1 FUTURE SCOPE

The future scope will be mainly targeted to the joint embedding method to improve the ranking mechanism by assigning priorities to entities while calculating the similarities. To reduce the searching and retrieval time, index optimization can be applied. To enrich the efficiency of the formula search process and retrieval of semantically similar formulas, textual data can be incorporated with the mathematical data. Further optimizing hyperparameters like context window, embedding size, sub sampling, etc can be done to improve the performance of the model.

Choosing the right formula size from the dataset might be a better approach. However, this poses a new question on determining the best formula size. We believe that research related to Automatic Term Extraction (ATE) [SK18] in technical or mathematical domains might provide valuable insights to this problem. The fuzzy string search used in the proposed method is slower but has the advantage that indexing is not needed. The kNN method [Guo+03] can be employed as a fast search engine, provided the formulas are indexed. Instead of choosing all formulas for training, longest formulas or random formulas could be used to improve the performance of the model.

Other possible improvement is labeling the questions into different categories like trigonometry, integral, binomial, logarithmic, etc., and trained using supervised learning. In the future, variations of the existing task can provide greater diversity of goals, such as systems optimized for specific types of formulas, text queries to find math, etc. We could also perform equation searches using words as queries with the embedding representation of words and equations. Further research is required in semantic extraction of mathematical formulas for better performance.

REFERENCES

[RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning Representations by Back-propagating Errors". In: *Nature* 323.6088 (1986), pp. 533–536. DOI: 10.1038/323533a0. URL: http://www.nature.com/articles/323533a0.

[HS97] Sepp Hochreiter and Jurgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[Ben+03] Yoshua Bengio et al. "A Neural Probabilistic Language Model". In: *J. Mach. Learn. Res.* 3 (2003), pp. 1137–1155. URL: http://jmlr.org/papers/v3/bengio03a.html.

[Guo+03] Gongde Guo et al. "KNN Model-Based Approach in Classification". In: *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE - OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003*. Ed. by Robert Meersman, Zahir Tari, and Douglas C. Schmidt. Vol. 2888. Lecture Notes in Computer Science. Springer, 2003, pp. 986–996. DOI: 10.1007/978-3-540-39964-3_62. URL: https://doi.org/10.1007/978-3-540-39964-3%5C_62.

[Teh+04] Yee Whye Teh et al. "Sharing Clusters among Related Groups: Hierarchical Dirichlet Processes". In: *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*. 2004, pp. 1385–1392. URL: https://proceedings.neurips.cc/paper/2004/hash/fb4ab556bc42d6f0ee0f9e24ec4d1af0-Abstract.html.

[Rob07] Herbert E. Robbins. "A Stochastic Approximation Method". In: *Annals of Mathematical Statistics* 22 (2007), pp. 400–407.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008. ISBN: 978-0-521-86571-5. URL: http://nlp.stanford.edu/IR-book/information-retrieval-book.html.

[SK08] Tetsuya Sakai and Noriko Kando. "On information retrieval metrics designed for evaluation with incomplete relevance assessments". In: *Inf. Retr.* 11.5 (2008), pp. 447–470. DOI: 10.1007/s10791-008-9059-7. URL: https://doi.org/10.1007/s10791-008-9059-7.

[08] "Spearman Rank Correlation Coefficient". In: *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, pp. 502–505. ISBN: 978-0-387-32833-1. DOI: 10.1007/978-0-387-32833-1_379. URL: https://doi.org/10.1007/978-0-387-32833-1_379.

[HER+10] N HERNANDEZ et al. "New challenges for nlp frameworks". In: *Proceedings of New Challenges for NLP Frameworks workshop in LREC*. Vol. 10. 2010.

[KAB12] P. Pavan Kumar, Arun Agarwal, and Chakravarthy Bhagvati. "A Structure Based Approach for Mathematical Expression Retrieval". In: *Multi-disciplinary Trends in Artificial Intelligence, 6th International Workshop, MIWAI 2012, Ho Chi Minh City, Vietnam, December 26-28, 2012. Proceedings*. Ed. by Chattrakul Sombattheera et al. Vol. 7694. Springer, 2012, pp. 23–34. DOI: 10.1007/978-3-642-35455-7_3. URL: https://doi.org/10.1007/978-3-642-35455-7%5C_3.

[NHC12] Tam T. Nguyen, Siu Cheung Hui, and Kuiyu Chang. "A lattice-based approach for mathematical search using Formal Concept Analysis". In: *Expert Syst. Appl.* 39.5 (2012), pp. 5820–5828. DOI: 10.1016/j.eswa.2011.11.085. URL: https://doi.org/10.1016/j.eswa.2011.11.085.

[MLS13] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. "Exploiting Similarities among Languages for Machine Translation". In: *CoRR* abs/1309.4168 (2013). arXiv: 1309.4168. URL: http://arxiv.org/abs/1309.4168.

[Mik+13a] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. Ed. by Christopher J. C. Burges et al. 2013, pp. 3111–3119. URL: https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.

[Mik+13b] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2013. URL: http://arxiv.org/abs/1301.3781.

[VC13] Karin Verspoor and Kevin Cohen. "Natural Language Processing". In: Jan. 2013, pp. 1495–1498. ISBN: 978-1-4419-9862-0. DOI: 10.1007/978-1-4419-9863-7_158.

[ZW13] Daxin Zhu and Xiaodong Wang. "A Simple Algorithm for Solving for the Generalized Longest Common Subsequence (LCS) Problem with a Substring Exclusion Constraint". In: *Algorithms* 6.3 (2013), pp. 485–493. DOI: 10.3390/a6030485. URL: https://doi.org/10.3390/a6030485.

[CLN14] Victor Codocedo, Ioanna Lykourentzou, and Amedeo Napoli. "A semantic approach to concept lattice-based information retrieval". In: *Ann. Math. Artif. Intell.* 72.1-2 (2014), pp. 169–195. DOI: 10.1007/s10472-014-9403-0. URL: https://doi.org/10.1007/s10472-014-9403-0.

[KTA14] Giovanni Yoko Kristianto, Goran Topic, and Akiko Aizawa. "Extracting Textual Descriptions of Mathematical Expressions in Scientific Papers". In: *D Lib Mag.* 20.11/12 (2014). DOI: 10.1045/november14-kristianto. URL: https://doi.org/10.1045/november14-kristianto.

[LM14] Quoc V. Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents". In: *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*. Vol. 32. JMLR Workshop and Conference Proceedings. JMLR.org, 2014, pp. 1188–1196. URL: http://proceedings.mlr.press/v32/le14.html.

[PS14] Partha Pakray and Petr Sojka. "An Architecture for Scientific Document Retrieval: Using Textual and Math Entailment Modules". In: *The 8th Workshop on Recent Advances in Slavonic Natural Languages Processing, RASLAN 2014, Karlova Studanka, Czech Republic, December 5-7, 2014*. Ed. by Ales Horak and Pavel Rychly. Tribun EU, 2014, pp. 107–117. URL: http://nlp.fi.muni.cz/raslan/2014/10.pdf.

[Cao+15] Ziqiang Cao et al. "A Novel Neural Topic Model and Its Supervised Extension". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. Ed. by Blai Bonet and Sven Koenig. AAAI Press, 2015, pp. 2210–2216. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9303.

[MZ15] Long Ma and Yanqing Zhang. "Using Word2Vec to process big text data". In: *2015 IEEE International Conference on Big Data, Big Data 2015, Santa Clara, CA, USA, October 29 - November 1, 2015*. IEEE Computer Society, 2015, pp. 2895–2897. DOI: 10.1109/BigData.2015.7364114. URL: https://doi.org/10.1109/BigData.2015.7364114.

[SW15] Adriaan M. J. Schakel and Benjamin J. Wilson. "Measuring Word Significance using Distributed Representations of Words". In: *CoRR* abs/1508.02297 (2015). arXiv: 1508.02297. URL: http://arxiv.org/abs/1508.02297.

[SZ15] David Stalnaker and Richard Zanibbi. "Math expression retrieval using an inverted index over symbol pairs". In: *Document Recognition and Retrieval XXII, San Francisco, California, USA, February 11-12, 2015*. Ed. by Eric K. Ringger and Bart Lamiroy. Vol. 9402. SPIE Proceedings. SPIE, 2015, p. 940207. DOI: 10.1117/12.2074084. URL: https://doi.org/10.1117/12.2074084.

[BK16] Oren Barkan and Noam Koenigstein. "ITEM2VEC: Neural item embedding for collaborative filtering". In: *26th IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2016, Vietri sul Mare, Salerno, Italy, September 13-16, 2016*. Ed. by Francesco A. N. Palmieri et al. IEEE, 2016, pp. 1–6. DOI: 10.1109/MLSP.2016.7738886. URL: https://doi.org/10.1109/MLSP.2016.7738886.

[PA16] Mateusz Pawlik and Nikolaus Augsten. "Tree edit distance: Robust and memory-efficient". In: *Information Systems* 56 (2016), pp. 157–173. ISSN: 0306-4379. DOI: https://doi.org/10.1016/j.is.2015.08.004. URL: https://www.sciencedirect.com/science/article/pii/S0306437915001611.

[Rud+16] Maja R. Rudolph et al. "Exponential Family Embeddings". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee et al. 2016, pp. 478–486.

[Zan+16a] Richard Zanibbi et al. “Multi-Stage Math Formula Search: Using Appearance-Based Similarity Metrics at Scale”. In: *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*. Ed. by Raffaele Perego et al. ACM, 2016, pp. 145–154. DOI: `10.1145/2911451.2911512`. URL: `https://doi.org/10.1145/2911451.2911512`.

[Zan+16b] Richard Zanibbi et al. “NTCIR-12 MathIR Task Overview”. In: *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies, National Center of Sciences, Tokyo, Japan, June 7-10, 2016*. Ed. by Noriko Kando, Tetsuya Sakai, and Mark Sanderson. National Institute of Informatics (NII), 2016. URL: `http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings12/pdf/ntcir/OVERVIEW/01-NTCIR12-OV-MathIR-ZanibbiR.pdf`.

[Zho+16] Guangyou Zhou et al. “Learning the Multilingual Translation Representations for Question Retrieval in Community Question Answering via Non-Negative Matrix Factorization”. In: *IEEE ACM Trans. Audio Speech Lang. Process.* 24.7 (2016), pp. 1305–1314. DOI: `10.1109/TASLP.2016.2544661`. URL: `https://doi.org/10.1109/TASLP.2016.2544661`.

[Boj+17] Piotr Bojanowski et al. “Enriching Word Vectors with Subword Information”. In: *Trans. Assoc. Comput. Linguistics* 5 (2017), pp. 135–146. URL: `https://transacl.org/ojs/index.php/tacl/article/view/999`.

[DZ17] Kenny Davila and Richard Zanibbi. “Layout and Semantics: Combining Representations for Mathematical Formula Search”. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. Ed. by Noriko Kando et al. ACM, 2017, pp. 1165–1168. DOI: `10.1145/3077136.3080748`. URL: `https://doi.org/10.1145/3077136.3080748`.

[Deh+17] Mostafa Dehghani et al. “Neural Ranking Models with Weak Supervision”. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. Ed. by Noriko Kando et al. ACM, 2017, pp. 65–74. DOI: `10.1145/3077136.3080832`. URL: `https://doi.org/10.1145/3077136.3080832`.

[Gao+17] Liangcai Gao et al. “Preliminary Exploration of Formula Embedding for Mathematical Information Retrieval: can mathematical formulae be embedded like a natural language?” In: *CoRR* abs/1707.05154 (2017). arXiv: `1707.05154`.

[Jan17] Stefan Jansen. “Word and Phrase Translation with word2vec”. In: *CoRR* abs/1705.03127 (2017). arXiv: `1705.03127`. URL: `http://arxiv.org/abs/1705.03127`.

[NCG17] Marwa Naili, Anja Habacha Chaibi, and Henda Hajjami Ben Ghezala. “Comparative study of word embedding methods in topic segmentation”. In: *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 21st International Conference KES-2017, Marseille, France, 6-8 September 2017*. Ed. by Cecilia Zanni-Merk et al. Vol. 112. Procedia Computer Science. Elsevier, 2017, pp. 340–349. DOI: `10.1016/j.procs.2017.08.009`. URL: `https://doi.org/10.1016/j.procs.2017.08.009`.

[TGH17] Julien Tissier, Christophe Gravier, and Amaury Habrard. “Dict2vec : Learning Word Embeddings using Lexical Dictionaries”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. Ed. by Martha Palmer, Rebecca Hwa, and Sebastian Riedel. Association for Computational Linguistics, 2017, pp. 254–263. DOI: 10.18653/v1/d17-1024. URL: https://doi.org/10.18653/v1/d17-1024.

[Yin+17] Wenpeng Yin et al. "Comparative Study of CNN and RNN for Natural Language Processing". In: *CoRR* abs/1702.01923 (2017). arXiv: 1702.01923. URL: http://arxiv.org/abs/1702.01923.

[Zha+17] Zhe Zhao et al. "Ngram2vec: Learning Improved Word Representations from Ngram Co-occurrence Statistics". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. Ed. by Martha Palmer, Rebecca Hwa, and Sebastian Riedel. Association for Computational Linguistics, 2017, pp. 244–253. DOI: 10.18653/v1/d17-1023. URL: https://doi.org/10.18653/v1/d17-1023.

[Che+18] Kehai Chen et al. "A Neural Approach to Source Dependence Based Context Model for Statistical Machine Translation". In: *IEEE ACM Trans. Audio Speech Lang. Process.* 26.2 (2018), pp. 266–280. DOI: 10.1109/TASLP.2017.2772846. URL: https://doi.org/10.1109/TASLP.2017.2772846.

[KB18] Kriste Krstovski and David M. Blei. "Equation Embeddings". In: *CoRR* abs/1803.09123 (2018). arXiv: 1803.09123. URL: http://arxiv.org/abs/1803.09123.

[PPG18] Amarnath Pathak, Partha Pakray, and Alexander F. Gelbukh. "A Formula Embedding Approach to Math Information Retrieval". In: *Computacion y Sistemas* 22.3 (2018). URL: http://www.cys.cic.ipn.mx/ojs/index.php/CyS/article/view/3015.

[SK18] Nisha Ingrid Simon and Vlado Keselj. "Automatic Term Extraction in Technical Domain using Part-of-Speech and Common-Word Features". In: *Proceedings of the ACM Symposium on Document Engineering 2018, DocEng 2018, Halifax, NS, Canada,August28-31,2018*. ACM, 2018, 51:1–51:4. DOI: 10.1145/3209280.3229100. URL: https://doi.org/10.1145/3209280.3229100.

[Yu+18] Liang-Chih Yu et al. "Refining Word Embeddings Using Intensity Scores for Sentiment Analysis". In: *IEEE ACM Trans. Audio Speech Lang. Process.* 26.3 (2018), pp. 671–681. DOI: 10.1109/TASLP.2017.2788182. URL: https://doi.org/10.1109/TASLP.2017.2788182.

[Man+19] Behrooz Mansouri et al. "Tangent-CFT: An Embedding Model for Mathematical Formulas". In: *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR 2019, Santa Clara, CA, USA, October 2-5, 2019*. Ed. by Yi Fang et al. ACM, 2019, pp. 11–18. DOI: 10.1145/3341981.3344235. URL: https://doi.org/10.1145/3341981.3344235.

[Mof19] Alistair Moffat. "Huffman Coding". In: *ACM Comput. Surv.* 52.4 (2019), 85:1–85:35. DOI: 10.1145/3342555. URL: https://doi.org/10.1145/3342555.

[PPD19] Amarnath Pathak, Partha Pakray, and Ranjita Das. "LSTM Neural Network Based Math Information Retrieval". In: *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*. 2019, pp. 1–6. DOI: 10.1109/ICACCP.2019.8882887.

[Wan+19] Bin Wang et al. "Evaluating word embedding models: methods and experimental results". In: *APSIPA Transactions on Signal and Information Processing* 8 (2019), e19. DOI: 10.1017/ATSIP.2019.12.

[YL19] Michihiro Yasunaga and John D. Lafferty. "TopicEq: A Joint Topic and Mathematical Equation Model for Scientific Texts". In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 7394–7401. DOI: 10.1609/aaai.v33i01.33017394. URL: https://doi.org/10.1609/aaai.v33i01.33017394.

[AK20] Akiko Aizawa and Michael Kohlhase. "Mathematical Information Retrieval". In: 2020.

[Gre+20] Andre Greiner-Petter et al. "Math-word embedding in math search and semantic extraction". In: *Scientometrics* 125.3 (2020), pp. 3017–3046. DOI: 10.1007/s11192-020-03502-9. URL: https://doi.org/10.1007/s11192-020-03502-9.

[MOZ20a] Behrooz Mansouri, Douglas W. Oard, and Richard Zanibbi. "DPRL Systems in the CLEF 2020 ARQMath Lab". In: *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020*. Ed. by Linda Cappellato et al. Vol. 2696. CEUR Workshop Proceedings. CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696/paper%5C_223.pdf.

[MOZ20b] Behrooz Mansouri, Douglas W. Oard, and Richard Zanibbi. "DPRL Systems in the CLEF 2020 ARQMath Lab". In: *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020*. Ed. by Linda Cappellato et al. Vol. 2696. CEUR Workshop Proceedings. CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696/paper%5C_223.pdf.

[Nov+20] Vit Novotny et al. "Three is Better than One: Ensembling Math Information Retrieval Systems". In: *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020*. Ed. by Linda Cappellato et al. Vol. 2696. CEUR Workshop Proceedings. CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696/paper%5C_235.pdf.

[OAM20] Bas van Opheusden, Luigi Acerbi, and Wei Ji Ma. "Unbiased and Efficient Log-Likelihood Estimation with Inverse Binomial Sampling". In: *CoRR* abs/2001.03985 (2020). arXiv: 2001.03985. URL: https://arxiv.org/abs/2001.03985.

[Zan+20] Richard Zanibbi et al. "ARQMath: a new benchmark for math-aware CQA and math formula retrieval". In: *SIGIR Forum* 54.2 (2020), 4:1–4:9. DOI: 10.1145/3483382.3483388. URL: https://doi.org/10.1145/3483382.3483388.

[CWW21] Stijn Cambie, Stephan Wagner, and Hua Wang. "On the maximum mean subtree order of trees". In: *Eur. J. Comb.* 97 (2021), p. 103388. DOI: 10.1016/j.ejc.2021.103388. URL: https://doi.org/10.1016/j.ejc.2021.103388.

[DPB21] Pankaj Dadure, Partha Pakray, and Sivaji Bandyopadhyay. "BERT-Based Embedding Model for Formula Retrieval". In: *Proceedings of the Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21st - to - 24th, 2021*. Ed. by Guglielmo Faggioli et al. Vol. 2936. CEUR Workshop Proceedings. CEUR-WS.org, 2021, pp. 36–46. URL: http://ceur-ws.org/Vol-2936/paper-03.pdf.

[Man+21] Behrooz Mansouri et al. “Overview of ARQMath-2 (2021): Second CLEF Lab on Answer Retrieval for Questions on Math (Working Notes Version)”. In: *Proceedings of the Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21st - to - 24th, 2021*. Ed. by Guglielmo Faggioli et al. Vol. 2936. CEUR Workshop Proceedings. CEUR-WS.org, 2021, pp. 1–24. URL: `http://ceur-ws.org/Vol-2936/paper-01.pdf`.

[Tan21] Ren Jie Tan. *Breaking down Mean Average Precision (mAP)*. en. Jan. 2021. URL: `https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52` (visited on 11/28/2021).

[WLB21] Zichao Wang, Andrew S. Lan, and Richard G. Baraniuk. “Mathematical Formula Representation via Tree Embeddings”. In: *Proceedings of the Third International Workshop on Inteligent Textbooks 2021 Co-located with 22nd International Conference on Artificial Intelligence in Education (AIED 2021), Online, June 15, 2021*. Ed. by Sergey A. Sosnovsky et al. Vol. 2895. CEUR Workshop Proceedings. CEUR-WS.org, 2021, pp. 121–133. URL: `http://ceur-ws.org/Vol-2895/paper02.pdf`.