Math-word embedding in math search and semantic extraction

André Greiner-Petter[1] · Abdou Youssef[2,3] · Terry Ruas[1] · Bruce R. Miller[3] ·
Moritz Schubotz[1,4] · Akiko Aizawa[5] · Bela Gipp[1]

Abstract

Word embedding, which represents individual words with semantically fixed-length vectors, has made it possible to successfully apply deep learning to natural language processing tasks such as semantic role-modeling, question answering, and machine translation. As math text consists of natural text, as well as math expressions that similarly exhibit linear correlation and contextual characteristics, word embedding techniques can also be applied to math documents. However, while mathematics is a precise and accurate science, it is usually expressed through imprecise and less accurate descriptions, contributing to the relative dearth of machine learning applications for information retrieval in this domain. Generally, mathematical documents communicate their knowledge with an ambiguous, context-dependent, and non-formal language. Given recent advances in word embedding, it is worthwhile to explore their use and effectiveness in math information retrieval tasks, such as math language processing and semantic knowledge extraction. In this paper, we explore math embedding by testing it on several different scenarios, namely, (1) math-term similarity, (2) analogy, (3) numerical concept-modeling based on the centroid of the keywords that characterize a concept, (4) math search using query expansions, and (5) semantic extraction, i.e., extracting descriptive phrases for math expressions. Due to the lack of benchmarks, our investigations were performed using the arXiv collection of STEM documents and carefully selected illustrations on the Digital Library of Mathematical Functions (DLMF: NIST digital library of mathematical functions. Release 1.0.20 of 2018-09-1, 2018). Our results show that math embedding holds much promise for similarity, analogy, and search tasks. However, we also observed the need for more robust math embedding approaches. Moreover, we explore and discuss fundamental issues that we believe thwart the progress in mathematical information retrieval in the direction of machine learning.

✉ André Greiner-Petter
  greinerpetter@uni-wuppertal.de

Extended author information available on the last page of the article

Introduction

Mathematics is capable of explaining complicated concepts and relations in a compact, precise, and accurate way. Learning this idiom takes time and is often difficult, even to humans. The general applicability of mathematics allows a certain level of ambiguity in its expressions. Short explanations or mathematical expressions are often used to mitigate the ambiguity problem, that serve as a context to the reader. Along with context-dependency, inherent issues of linguistics (e.g., ambiguity, non-formality) make it even more challenging for computers to understand mathematical expressions. Nevertheless, a system capable of automatically capturing the semantics of mathematical expressions would be suitable for improving several applications, from search engines to recommendation systems.

Word embedding (Bengio et al. 2003; Mikolov et al. 2013a; Pennington et al. 2014) has made it possible to apply deep learning in natural language processing (NLP) with great effect. That is because embedding represents individual words with numerical vectors that capture contextual and relational semantics of the words. Such representation enables inputting words and sentences to a neural network (NN) in numerical form. This allows the training of NNs and using them as predictive models for various NLP tasks and applications, such as semantic role modeling (He et al. 2017; Zhou and Xu 2015), word-sense disambiguation (Iacobacci et al. 2016; Raganato et al. 2017), sentence classification (Kim 2014), sentiment analysis (Socher et al. 2013), coreference resolution (Lee et al. 2017; Wiseman et al. 2016), named entity recognition (Chiu and Nichols 2016), reading comprehension (Clark and Gardner 2018), question answering (Liu et al. 2018), natural language inference (Chen et al. 2017; Gong et al. 2018), document classification (Ruas et al. 2020), and machine translation (Devlin et al. 2014). The performance of word embedding in NLP tasks has been measured and shown to deliver fairly high accuracy (Mikolov et al. 2013b; Pennington et al. 2014; Peters et al. 2018).

As math text consists of natural text as well as math expressions that exhibit linear and contextual correlation characteristics that are very similar to those of natural sentences, word embedding applies to math text much as it does to natural text. Accordingly, it is worthwhile to explore the use and effectiveness of word embedding in math language processing (MLP), math knowledge management (MKM), and math information retrieval (MathIR). Still, math expressions and math writing styles are different from natural text to the point that NLP techniques have to undergo significant adaptations and modifications to work well in math contexts.

While some efforts have started to apply word embedding to MLP, such as equation embedding (Gao et al. 2017; Krstovski and Blei 2018; Yasunaga and Lafferty 2019; Greiner-Petter et al. 2019; Youssef and Miller 2019), there is a healthy skepticism about the use of machine learning (ML) and deep learning (DL) in MLP and MKM, on the basis that much work is still required to prove the effectiveness of DL in MLP. To learn how to adapt and apply DL in the MLP/MKM/MathIR context is not an easy task. Most applications of DL in MLP/MKM/MathIR rest on the effectiveness of word/math-term embedding (henceforth *math embedding*) because the latter is the most basic foundation in language DL. Therefore, it behooves us to start to look at the effectiveness of math embedding in basic tasks, such as term similarity, analogy, information retrieval, and basic math search, to learn more about their extension and limitations. More importantly, we need to learn how to refine and evolve math embedding to become accurate enough for more severe applications, such as knowledge extraction. That is the primary objective of this paper.

Working with MathMLBen (Schubotz et al. 2018), a benchmark for converting mathematical LaTeX expressions into MathML, we discovered several fundamental problems that generally affect MLP/MKM/MathIR towards ML/DL solutions to learn semantics of mathematical expressions. For instance, the first entry of the benchmark,

$$W(2, k) > 2^k/k^\varepsilon \tag{1}$$

is extracted from the English Wikipedia page about Van der Waerden's theorem.[1] Without further explanation, the symbols W, k, and ε might have several possible meanings. Depending on which one is considered, even the structure of the formula may be different. If we consider W as a variable, instead of a function, it changes the interpretation of $W(2, k)$ to a multiplication operation. Learning connections, such as between W and the entity 'Van der Waerden's number', requires a large specifically labeled scientific database that contains these mathematical objects.

To that effect, there is a fundamental need for datasets and benchmarks, preferably standard ones, to allow researchers to measure the performance of various math embedding techniques, and applications based on them, in an objective and statistically significant way, and to measure improvements and comparative progress. Such resources are abundant in the natural language domain but scarce in the MLP domain. Developing some of such datasets and benchmarks will hopefully form the nucleus for further development by the community to facilitate research and speed up progress in this vital area of research.

While the task of creating such resources for DL applications in MLP can be long and demanding, the examination of math embedding should not wait but should proceed right away, even if in an exploratory manner. Early evaluations of math embedding should ascertain its value for MLP/MKM/MathIR and inform the process and trajectory of creating the corpora and benchmarks. Admittedly, until adequate datasets and benchmarks become available for MLP, we have to resort to less systematic performance evaluation and rely on performing preliminary tests on the limited resources available. The DLMF (DLMF 2018) and arXiv.org preprint archive[2] are good resources to start our exploratory embedding efforts. The DLMF offers high quality, and the authors are familiar with its structure and content (which aids in crafting some of the tests). As for the arXiv collection, its large volume of mostly math articles makes it an option worth to investigate as well.

In this paper, we provide an exploratory investigation of the effectiveness and use of word embedding in MLP and MKM through different perspectives. First, we train word-2vec models on the DLMF and arXiv with slightly different approaches for embedding math. Since the DLMF is primarily a handbook of mathematical equations, it does not provide extensive textual content. We will show that the DLMF trained model is appropriate to discover mathematical term similarities and term analogies, and to generate query expansions. We hypothesize that the arXiv trained models are beneficial to extract definiens, i.e., textual descriptive phrases for math terms. We examine the possible reasons why the word embedding models, trained on the arXiv dataset, does not present valuable results for this task. Besides, we discuss some of the reasons that we believe thwart the progress in MathIR in the direction of machine learning. In summary, we focus on five tasks (i) term similarity, (ii) math analogies, (iii) concept modeling, (iv) query expansion, and (v) knowledge extraction.

---

[1] https://en.wikipedia.org/wiki/Van_der_Waerden's_theorem [Accessed Sep. 2019].

[2] https://arxiv.org/ [Accessed Sep. 2019].

The paper is organized as follows. Next section offers a survey of the foundations and prior work related to word and math embedding. The "Mathematical Information Retrieval" section presents our experiments on the DLMF trained model: term similarity, math analogies, concept modeling, and query expansion. In the section "Semantic Knowledge Extraction" we explore the arXiv trained model for knowledge extraction of mathematical terms. The "Overcoming Issues of Knowledge Extraction" section discusses potential concepts we believe are necessary to overcome the limitations of ML and DL algorithms in MLP/MKM/MathIR tasks. Finally, the last section presents our conclusions and outlines future directions.

Foundations and related work

Understanding mathematical expressions essentially mean comprehending the semantic value of its internal components, which can be accomplished by linking its elements with their corresponding mathematical definitions. Current MathIR approaches (Kristianto et al. 2014; Schubotz et al. 2016, 2017) try to extract textual descriptors of the parts that compose mathematical equations. Intuitively, there are questions that arise from this scenario, such as (i) how to determine the parts which have their own descriptors, and (ii) how to identify correct descriptors over others.

Answers to (i) are more concerned in choosing the correct definitions for which parts of a mathematical expression are considered as one mathematical object (Kohlhase 2017; Youssef 2017; Schubotz et al. 2018). Current definition-languages, such as the content MathML 3.0[3] specification, are often imprecise.[4] For example, content MathML 3.0 uses 'csymbol' elements for functions and specifies them as expressions that *refer to a specific, mathematically-defined concept with an external definition.*[5] However, it is not clear whether W or the sequence $W(2, k)$ (from (1)) should be declared as a 'csymbol'. Another example involves content identifiers, which MathML specifies as *mathematical variables that have properties, but no fixed value.*[6] While content identifiers are allowed to have complex rendered structures (e.g., β_i^2), it is not permitted to enclose identifiers within other identifiers. Let us consider α_i, where α is a vector and α_i its ith element. In this case, α_i should be considered as a composition of three content identifiers, each one carrying its own individualized semantic information, namely the vector α, the element α_i of the vector, and the index i. However, with the current specification, the definition of these identifiers would not be canonical. One possible workaround to represent such expressions with content MathML is to use a structure of four nodes, interpreting α_i as a function via a 'csymbol' (one parent 'apply' node with the three children *vector-selector*, α, and i). However, ML algorithms and MathIR approaches would benefit from more precise definitions and a unified answer for (i). Most of the related work relies on these relatively vague definitions and in the analysis of content identifiers, focusing their efforts on (ii).

---

[3] https://www.w3.org/TR/MathML3/ [Accessed Sep. 2019].

[4] Note that OpenMath is another specification designed to encode semantics of mathematics. However, content MathML is an encoding of OpenMath and inherent problems of content MathML also apply to OpenMath (see https://www.openmath.org/om-mml/) [Accessed Sep. 2019].

[5] https://www.w3.org/TR/MathML3/chapter4.html#contm.csymbol [Accessed Sep. 2019].

[6] https://www.w3.org/TR/MathML3/chapter4.html#contm.ci [Accessed Sep. 2019].

Questions (i), (ii), and other pragmatic issues are already in discussion in a bigger context, as data production continues to rise and digital repositories seem to be the future for any archive structure. A prominent example is the National Research Council's effort to establish what they call the *Digital Mathematics Library* (DML),[7] a project under the International Mathematical Union. The goal of this project is to take advantage of new technologies and help to solve the inability to search, relate, and aggregate information about mathematical expressions in documents over the web.

The advances most relevant to our work are the recent developments in *word embedding* (Mikolov et al. 2013b; Cho et al. 2014; Pennington et al. 2014; Bojanowski et al. 2017; Rudolph et al. 2017; Cer et al. 2018; Peters et al. 2018). Word embedding takes as input a text collection and generates a numerical feature vector (typically with 100 or 300 dimensions) for each word in the collection. This vector captures latent semantics of a word from the contexts of its occurrences in the collection; in particular, words that often co-occur nearby tend to have similar feature vectors (where similarity is measured by the cosine similarity, the Euclidean distance, etc.).

Recently, more and more projects try to adapt these word embedding techniques to learn patterns of the correlations between context and mathematics. In the work of Gao et al. (2017), they embed single symbols and train a model that can discover similarities between mathematical symbols. Similarly to this approach, Krstovski and Blei (2018) use a variation of word embedding (briefly discussed in the "Word Embedding" section) to represent complex mathematical expressions as single unit tokens for IR. In 2019, Yasunaga and Lafferty (2019) explore an embedding technique based on recurrent neural networks to improve topic models by considering mathematical expressions. They state their approach outperforms topic models that do not consider mathematics in text and report a topic coherence improvement of 0.012 over the LDA[8] baseline. Equation embedding, as in Gao et al. (2017); Krstovski and Blei (2018); Yasunaga and Lafferty (2019), present promising results for identifying similar equations and contextual descriptive keywords.

In the following, we will explore in more detail different techniques of word embedding ("Word Embedding" section). Likewise, we will examine different styles of adapting the process for math embedding ("Math Embedding" section).

Word embedding

In this paper, we apply *word2vec* (Mikolov et al. 2013b) on the DLMF (DLMF 2018) and on the collection of arXiv.org pre-print archive[9] documents for generating embedding vectors for various math symbols and terms. The word2vec technique computes real-valued vectors for words in a document using two main approaches: skip-gram and continuous bag-of-words (CBOW). Both produce a fixed-length n-dimensional vector representation for each word in a corpus. In the skip-gram training model, one tries to predict the context of a given the word, while CBOW predicts a target word given its context. In word2vec, context is defined as the adjacent neighboring words in a defined range, called a sliding window. The main idea is that the numerical vectors representing similar words should

---

[7] https://www.nap.edu/read/18619 [Accessed Sep. 2019].

[8] Latent Dirichlet Allocation.

[9] https://arxiv.org/ [Accessed Sep. 2019].

have close values if the words have similar context, often illustrated by the *king–queen relationship*

$$\mathbf{v}_{\text{king}} - \mathbf{v}_{\text{man}} \approx \mathbf{v}_{\text{queen}} - \mathbf{v}_{\text{woman}} \tag{2}$$

where $\mathbf{v}_t$ represents the vector for the token t.

Extending word2vec's approaches, Le and Mikolov (2014) propose *Paragraph Vectors* (PV), a framework that learns continuous distributed vector representations for any size of text segments (e.g., sentences, paragraphs, documents). This technique alleviates the inability of word2vec to embed documents as one single entity. This technique also comes in two distinct variations: Distributed Memory (DM) and Distributed Bag-of-Words (DBOW), which are analogous to the skip-gram and CBOW training models, respectively.

Other approaches also produce word embedding given a training corpus as input, such as fastText (Bojanowski et al. 2017), ELMo (Peters et al. 2018), and GloVe (Pennington et al. 2014). The choice for word2vec for our experiments is justified because of its implementation ease, training speed using modest computing resources, general applicability, and robustness in several NLP tasks (Iacobacci et al. 2015, 2016; Li and Jurafsky 2015; Mancini et al. 2017; Pilehvar and Collier 2016; Ruas et al. 2019). Additionally, in fast-Text they propose to learn word representations as a sum of the *n*-grams of its constituent characters (sub-words). The sub-word structure would incorporate a certain noise[10] to our experiments. In ELMo, they compute their word vectors as the average of their characters representations, which are obtained through a two-layer bidirectional language model (biLM). This would bring even more granularity than fastText, as they consider each character in a word as having their own *n*-dimensional vector representation. Another factor that prevents us from using ELMo, for now, is its expensive training process.[11] Closer to the word2vec technique, GloVe (Pennington et al. 2014) is also considered, but its co-occurrence matrix would escalate the memory usage, making its training for arXiv not possible at the moment. We also examine the recently published Universal Sentence Encoder (USE) (Cer et al. 2018) from Google, but their implementation does not allow one to use a new training corpus, only to access its pre-calculated vectors based on words. We also considered BERT (Devlin et al. 2019) with its recent advances of Transformer-based architectures in NLP as an alternative to *word2vec*. However, incorporating BERT and other Transformer-based architectures would require a significant restructuring of the core idea of our work. BERT is pre-trained in two general tasks that are not directly transferable to mathematics embeddings: Masked Language Modelling (MLM) and Next Sentence Prediction (NSP). Since this work is an exploratory investigation of the potential of word embedding techniques in MLP and MKM, we gave preference to tools that could be applied directly. Nonetheless, since some of our results are promising, we plan to include Transformer-based systems, such as BERT (Devlin et al. 2019), XLNet (Yang et al. 2019), RoBERTa (Liu et al. 2019), and Transformers-XL (Dai et al. 2019), in future work.

The overall performance of word embedding algorithms has shown superior results in many different NLP tasks, such as machine translation (Mikolov et al. 2013b), relation similarity (Iacobacci et al. 2015), word sense disambiguation (Camacho-Collados et al. 2015), word similarity (Neelakantan et al. 2014; Ruas et al. 2019), document classification (Ruas et al. 2020), and topic categorization (Pilehvar et al. 2017). In the same direction, we also

---

[10] Noise means, the data consists of many uninteresting tokens that affect the trained model negatively.

[11] https://github.com/allenai/bilm-tf [Accessed Feb. 2020].

explore how well mathematical tokens can be embedded according to their semantic information. However, mathematical formulae are highly ambiguous and, if not properly processed, their representation is jeopardized.

Math embedding

Recently, Krstovski and Blei (2018) proposed a variation of word embedding for mathematical expressions. Their main idea relies on the construction of a distributed representation of equations, considering the word context vector of an observed word and its word-equation context window. They treat equations as single-unit words (EqEmb), which eventually appears in the context of different words. They also try to explore the effects of considering the elements of mathematical expressions separately (EqEmb-U). In this scenario, mathematical equations are represented using a Syntax Layout Tree (SLT) (Zanibbi et al. 2016b), which contains the spatial relationship between its symbols. While they present some interesting findings for retrieving entire equations, there is little discussion about the vectors representing equation units, i.e., EqEmb-U embedding, and how they are described in their model. The word embedding techniques seem to have the potential for semantic distance measures between complex mathematical expressions. However, they are not appropriate for extracting the semantics of identifiers separately, indicating that the problems of representing mathematical identifiers are tied to more fundamental issues, which we address in the "Overcoming Issues of Knowledge Extraction" section.

Considering the equation embedding techniques in Krstovski and Blei (2018), we devise three main types of mathematical embedding, nameley *Mathematical Expressions as Single Tokens*, *Stream of Tokens*, and *Semantic Groups of Tokens*.

Mathematical Expressions as Single Tokens: EqEmb (Krstovski and Blei 2018) uses entire mathematical expressions as one token. In a one-token representation, the inner structure of the mathematical expression is not considered. For example, Eq. (1) is represented as one single token t_1. Any other expression, such as $W(2, k)$ in the surrounding text of (1), is an entirely independent token t_2. Therefore, this approach does not learn any connections between $W(2, k)$ and (1). However, Krstovski and Blei (2018) has shown promising results for comparing mathematical expressions with this approach.

Stream of Tokens: As an alternative to embedding mathematical expressions as a single token, one can also represent an expression through a sequence of its inner elements. For example, considering only the identifiers in Eq. (1), it would generate W, k, and ε as a sequence/stream of tokens. This approach has the advantage of learning all mathematical tokens. However, this method also has some drawbacks. Complex mathematical expressions may lead to long chains of elements, which can be especially problematic when the window size of the training model is too small. Naturally, there are approaches to reduce the length of chains. Gao et al. (2017) use a CBOW and embed all mathematical symbols, including identifiers and operands, such as $+$, $-$ or variations of equalities $=$. Yasunaga and Lafferty (2019) do not cut out symbols, but train their model on the entire sequence of tokens that the LaTeX tokenizer generates. Considering Eq. (1), it would result in a stream of 13 tokens. They use a long short-term memory (LSTM) architecture to overcome this issue and further limit chains length to 20–150 tokens. Usually, in word embedding, such behaviour is not preferred since it increases the noise in the data.

In the "Mathematical Information Retrieval" section, we use this stream of tokens approach to train our model on the DLMF without any filters. Thus, Eq. (1) generates

all 13 tokens. In the "Overcoming Issues of Knowledge Extraction" section, we show another model trained on the arXiv collection, which uses a stream of mathematical identifiers and cut out all other expressions, i.e., in case of (1), we embed W, k, and ε. We presume this approach is more appropriate to learn connections between identifiers and their definiens. We will see later in the paper that both of our models trained on math embedding is able to detect similarities between mathematical objects, but does not perform well detecting connections to word descriptors. In the scenario of identifying definiens, for mathematical objects, we consider close relations between mathematical symbols as noise. To mitigate this issue, we only work with mathematical identifiers and not any other symbols or structures for our experiments on the arXiv collections. Note that, since we focused on similarities for the experiments on the DLMF dataset, we preferred to not filter out any tokens for the DLMF model.

Semantic Groups of Tokens: The third approach of embedding mathematics is only theoretical and concerns the problem mentioned above related to the vague definitions of identifiers and functions in a standardized format (e.g., MathML). As previously discussed, current MathIR and ML approaches would benefit from a basic structural knowledge of mathematical expressions, such that variations of function calls (e.g., $W(r,\ k)$ and $W(2,\ k)$) can be recognized as the same function. Instead of defining a unified standard, current techniques use their ad-hoc interpretations of structural connections, e.g., α_i is one identifier rather than three (Schubotz et al. 2017, 2018). We assume that an embedding technique would benefit from a system that can detect the parts of interest in mathematical expressions before any training processes. However, such a system still does not exist.

To investigate the situations described in the sections "Word Embedding" and "Math Embedding", we applied word2vec on two different scenarios, one focusing on MathIR (DLMF) and the other on semantic knowledge extraction (arXiv), i.e., identifying definiens for math objects. To summarize our decisions, for the DLMF and arXiv, we choose the stream of token embedding technique, i.e., each inner token is represented as a single n-dimensional vector in the embedding model. For the DLMF (section "Mathematical Information Retrieval"), we embed all inner tokens, while for the arXiv (section "Semantic Knowledge Extraction"), we only embed the identifiers.

Mathematical information retrieval

To perform the MathIR experiments on the DLMF using word and math-term embedding, we trained word2vec on the DLMF. Considerable preprocessing of the corpus had to performed: new algorithms and software for sentence-segmentation (in math documents) had to be developed, and the Part-of-Math tagger[12] (PoM tagger) (Youssef 2017) was adapted and used for math tokenization. For preprocessing the DLMF, we separated the content from annotations and metadata, and segmented all the contents into individual sentences (using our sentence-segmentation algorithm) since in the version of word2vec embedding that we used, the tokenization worked on a sentence by sentence basis. Afterwords, word-2vec was applied using the skip-gram model, a window size of 5, dimension of 100, and

---

[12] A tokenizer for LaTeX expressions that tags the tokens with additional information similar to Part-of-Speech taggers in NLP.

Table 1 Keywords and their top-20 most similar words, by the Euclidean distance

Transform	Fourier	Bessel	hypergeometric
transform	Fourier	Bessel	hypergeometric
Mellin	power	Airy	generalized
Transform	Hilbert	Hankel	confluent
Transforms	Heun	modified	multivariate
extend	Maclaurin	Struve	generating
By	Stieltjes	Modified	Olver's
defining	radii	Generalized	Lauricella's
Stieltjes	joining	Spherical	Heun
Hilbert	summable	Coulomb	Appell
allows	noninteger	Many	gamma
convolution	Transform	Inverse	bilateral
standard	Laurent	products	basic
rise	Every	Kelvin	elementary
group	trapezoidal	Mathieu	Gauss
us	geometric	spheroidal	Kummer
summable	rules	Weber	Inverse
transformation	iterative	spherical	Many
ellipsoids	Lagrange	gamma	plays
solve	vacuum	Lamé	Coulomb
Since	construction	Contiguous	beta

minimum word count of 3. These hyperparameter values were obtained after considerable experimentation with different values of the window size, vector dimension, and minimum word count. We selected the values that resulted in best performance (as evaluated by direct observations of the outcome of the example similarity/analogy queries that are reported on in this section).

Term similarity

Table 1 presents some initial findings when searching for the 20 most similar words for each of the following four math terms in the DLMF database: 'transform', 'Fourier', 'Bessel', and 'hypergeometric'. Identifying similar words (to a given keyword) can:

1. Serve as an indicator of the semantics-capturing capabilities of the underlying word embedding technique;
2. Enrich search queries (by combining the keyword with its semantic/related neighbors into an expanded query);
3. Find related concepts that could not be found as efficiently and conveniently as through embedding-based word similarity.

The context of the DLMF is very specific. Hence, the name *Fourier* (as well as other names, such as *Bessel*) only appears in a narrow context, primarily as descriptive linguistic modifiers of mathematical constructs such as 'transforms', 'series' and 'operators'.

The results in Table 1 show that many of the returned *hits* are quite what a mathematician would expect (especially hits for the words 'Bessel' and 'hypergeometric' in columns 3 and 4), but at the same time, certain similar/related terms failed to be returned. For example, considering the hits for 'transform' (1st column of the table), we observe:

– The top-20 hits showed some synonyms (e.g., 'transformation') and related terms like 'convolution', 'Mellin', and 'Hilbert' (the latter two are due probably to 'Mellin transform' and 'Hilbert transform'), which are all good;
– The top-20 hit list failed to include 'Fourier', despite the fact that it is arguably the most famous transform;
– As expected, certain irrelevant words (e.g., 'by', 'allows') matched high because many of them are frequent stopwords. However, not all general stopwords should be dismissed, e.g., 'almost everywhere' has major significance in math, but what constitutes stopwords in math should be considered carefully.

Also, looking at the 2nd column, the hits of words similar to 'Fourier' include many other terms that are truly related to the keyword 'Fourier', where in several instances (e.g., 'Stieltjes' and 'Hilbert'), the similarity could perhaps be attributed to the fact there are transforms associated with those terms. Unlike in the previous column, the word 'Transform' rightly appears in the top-20 similar words of 'Fourier'. This lack of symmetry, though understandable, shows that similarity, or rather *dissimilarity*, (based on word2vec embedding), is not a measure in the mathematical sense, which can be a serious shortcoming. To be sure, the lack of symmetry in term similarity is not an issue in information retrieval carried out by human users, if what is represented allows users to express queries whose correspondence with language usage statistics in a corpus is strong. Solid textual retrieval systems themselves often do not represent word/conceptual semantics in any direct sense. However, in certain situations, such as in end-to-end systems with no humans in the loop, where the non-symmetry (X is similar to Y but Y may not be similar to X) makes the behavior of such systems unpredictable (under the mercy of the right choice of the query term) and thus less than desirable. That being said, the cautionary concern over the lack of symmetry is mostly speculative and hypothetical at this time, and only future experimentation and applications will tell whether or not the non-symmetry is a serious problem.

These four similarity exercises are too few to draw any solid conclusions up to this point, but they illustrate the drawbacks and the promises of embedding for MLP, and press the need for benchmarks to achieve generalizable, statistically significant results.

Remarks about the distance measures used: In the similarity and analogy search experiments that we ran and present in this section, we considered different similarity measures and distance measures, such as cosine similarity and the Euclidean distance. Since the optimization of the best measure/distance is not of primary concern in this paper, and to save on space, we present only the results corresponding to the best measure. For example, in Table 1, we present the results where the Euclidean distance was used because it gave the best results, whereas in Tables 2, 3, 4, 5 and 6, we use cosine similarity because it yielded the best outcome.

Term analogy

Finding mathematical analogies is a powerful tool for crafting queries for analogy search, which cannot be performed by mere keyword search. Here are some examples: 'x' is to

Table 2 Analogies of the form: Find Term where Term is to X what Y is to Z. The similarity measure is cosine similarity

Top-10 best *Terms* where		
Term is to '*complex*' what x is to '*real*':	*Term* is to x what '*complex*' is to '*real*':	*Term* is to z what '*real*' is to '*complex*':
x	x	z
z	z	x
\left(	\left(	2
\right)	\right)	t
,	,	−
1	1	1
=	=	\right)
−	−	n
\pi	\pi	\pi
−	−	+
Term is to 'sin' what 'cosh' is to 'cos':	*Term* is to 'sin' what 'arccos' is to 'cos':	*Term* is to 'exp' what 'arccos' is to 'cos':
cosh	arccos	arccos
sinh	arcsinh	arccosh
sin	arctan	arctan
tanh	arctanh	arctanh
csch	arccosh	arcsinh
cot	arcsin	exp
coth	arccsc	arcsin
mt	arcsec	erfc
zs	arccoth	sign
sech	arccsch	xyz

Table 3 The words/lemmas most similar to the centroid of 'Fourier' and 'Mellin', by cosine similarity

Top-20 Most Similar *Words* for Centroid of {'Fourier', 'Mellin'}		Top-10 Most Similar *Lemmas* for Centroid of {'Fourier', 'Mellin'}
Mellin	Stieltjes	Mellin
Fourier	Leading	Fourier
Hilbert	convolution	Hilbert
Laplace	many-body	Laplace
Transform	summable	Transform
Kontorovich–Lebedev	ease	Kontorovich–Lebedev
transform	collections	products
products	FFT	Stieltjes
Transforms	us	Leading
transforms	Convergence	convolution

Table 4 Similarities to the centroids of 4 subsets of words, by cosine similarity

Top-20 Most Similar Words for the Centroid of			
{'se', 'ce'}	{'Si', 'Ci'}	{'Ai', 'Bi'}	{'sin', 'cos', 'tan'}
ce	Ci	Bi	sin
se	Si	Ai	cos
fe	Ei	envAi	tan
ge	nt	envBi	cot
Se	ez	Hi	cosh
Fe	sec	Gi	sinh
Ce	Arctan	envelope	tanh
Ge	xe	Airy	uv
Io	xM	xe	Arctanh
Gey	Shi	1535	csch
Ko	Gi	Chi	pm
me	Chi	'	si
Ke	ie	54703	Ein
Fey	Ein	Shi	ir
Ds	zn	implicitly	Arccsc
Dc	6144	derivative	sec
mz	Cin	Ein	Arccosh
Ie	xJ	1797	coth
Me	arccot	ie	csc
inh	1797	Ei	rh

'real' what 'z' is to 'complex', 'cos' is to 'cosh' what 'sin' is to 'sinh', 'cos' to 'arccos' is what 'log' is to 'exp', 'arcsin' is to 'sin' what 'integral' is to 'derivative', and so on. To illustrate the use of analogies, consider this simple example of a math student who has taken courses on real analysis and is just starting to learn complex analysis. That student is likely curious to know the common notation for a complex variable, as the counterpart of 'x' being the common notation for a real variable. In plain English, the student can formulate that information need as a query/question of the form: What is to 'complex' as 'x' is to 'real'? This query is essentially the aforementioned king–queen example (2) from the "Word Embedding" section with an unknown variable $\mathbf{v}_{unknown}$. Thus, with powerful word embedding, the unknown term being searched for satisfies the following relation

$$\mathbf{v}_{unknown} - \mathbf{v}_{complex} \approx \mathbf{v}_x - \mathbf{v}_{real} \tag{3}$$

or equivalently

$$\mathbf{v}_{unknown} \approx \boxed{\mathbf{v}_x} + \boxed{\mathbf{v}_{complex} - \mathbf{v}_{real}} \tag{4}$$

where $\mathbf{v}_t$ is the embedding vector of term t. Accordingly, to find the *unknown term*, one has to find the closest vectors to the vector $\mathbf{v}_x + \mathbf{v}_{complex} - \mathbf{v}_{real}$, and retrieve the corresponding words. Ideally, with good embedding, the unknown term should be the top match or at least among the top few matches. Note that in the equation above, the vector on the right hand side of the approximation is the sum of the vector for the *known term* (i.e., first box)

Table 5 Similarities to the centroids of three subsets of words, by cosine similarity

	Top-20 Most Similar Words for the Centroid of		
	{'Legendre', 'Hypergeometric'}	{'Bessel', 'Struve'}	{'Legendre', 'Hypergeometric', 'Bessel', 'Struve'}
	Legendre	Bessel	Struve
	Hypergeometric	Struve	Bessel
	Generalized	Kelvin	Generalized
	Struve	Hankel	Hypergeometric
	Gamma	Modified	Weber
	Generating	Weber	Contiguous
	dilated	Noninteger	Modified
	Associated	Airy	Gamma
	Arguments	Spherical	Legendre
	Confluent	Contiguous	Kelvin
	Products	Anger–Weber	Hankel
	Ferrers	Many	Associated
	Contiguous	Generalized	Anger–Weber
	Number-Theoretic	Inverse	Many
	Incomplete	Half-Integer	Noninteger
	Parabolic	Functions-Real	Generating
	Kind	Functions-Complex	Confluent
	Mittag–Leffler	Gamma	Spherical
	Function	Incomplete	Incomplete
	Elementary	gamma	Arguments

and the vector (in the second box) that captures the relation between the known and the unknown terms. We will call the second-box vector the *relation vector*.

We tested this analogy capability using several analogy queries, as shown in Table 2 where *Term* stands for the unknown term being searched for. The query illustrated above about the complex variable notation was formulated in two different flavors (top half of the first two columns), and, for extra measure, the question was modified to search for the real variable given the complex variable (top half of the third column); the purpose of varying the question is to test for robustness. Examining the results in the table, one can observe the following:

– In all three queries, the desired answer was the second topmost match, which, though not ideal, is quite impressive.
– The topmost match for the unknown term is, interestingly, the known term, in all three queries. That indicates that the relation vector is, at least in these queries, of very small magnitude. This could result from one of two factors: (1) the vectors of 'complex' and 'real' are quite similar due to the strong inter-relatedness, making their difference quite small, or (2) those two vectors are of small magnitude, causing their difference to be small as well. The 1st factor is mitigated when the relation is between two disparate (i.e., not so correlated) concepts, while the 2nd factor can be remedied by taking *normalized* embedding, resulting in unit-length vectors.

Table 6 The words most similar to the weighted centroids of {'Euler', 'Gamma'}, by cosine similarity. Note that certain matches like ':sec:LA.F2.DC' are obviously wrong. Those are tokenizer artifacts generated by the used tokenizer and will be fixed in future versions

Top-20 Most Similar Words to the Centroid of {'Euler', 'Gamma'} using different weights w_1 and w_2

$(w_1, w_2) = (1, 0)$	$(w_1, w_2) = (1, 1)$	$(w_1, w_2) = (1, 2)$
Euler	Euler	Euler
Bernoulli	LA4	Gamma
belonging	Bernoulli	Beta
cyclotomic	Numbers	Generating
polynomials	exponentials	Periodic
splines	Periodic	Parabolic
Numbers	splines	Cylinder
LA4	Beta	LA4
generating	belonging	exponentials
Stirling	Gamma	Number-Theoretic
:sec:LA.F2.DC	15-point	Generalized
quotients	:sec:LA.F2.DC	Unmodified
Splines	5-point	Numbers
1851	Curve	Modular
Genocchi	cyclotomic	Exponential
Hermite	Cylinder	Toroidal
Factorization	generating	Trigonometric
Computer	Factorization	Elementary
exponentials	Gauss	Contiguous
15-point	Generating	Associated

The bottom half of Table 2 shows the results for three other analogy queries. The first query tests for analogy between trigonometric $(\sin, \cos)$ and hyperbolic functions $(\sinh, \cosh)$, the second query tests for inverse function relation in trigonometry, i.e., analogy between $(\sin, \arcsin)$ and $(\cos, \arccos)$, and the third query also tests for inverse function relation but this time between functions from different areas (one pair from trigonometry, namely, $(\cos, \arccos)$, and the other pair being $(\exp, \log)$). One can observe the following about the results of those queries:

– For the first 2 queries, we observe the same as above: the desired match ranks second, and the topmost match is (wrongly) identical to the known term;
– For the third query, the desired match is not even in the top-10 list.

All mentioned observations, but especially the last one, point to the urgent need for mass testing on large benchmarks, to assess the power and limitations of embedding in MLP applications accurately, and for better diagnosis of shortcomings so that more targeted remedies or adjustments can be made. For example, is it the case that, when analogies are being drawn from two rather different areas, the analogy search is not as accurate? The preliminary tests performed in this paper do not give us a strong basis for answering such a question, but further tests on large benchmarks will almost certainly provide an answer.

Concept modeling and query expansion

Many concepts can be *described* by a (sub)set of keywords that capture different aspects of the concept, and serve as 'axes' that characterize the concept. For example, the concept of 'dog' can be described (at least partially) by keywords like 'animal', 'domestic', 'friendly', 'loyal', etc. Note that when a concept has no name, the characterizing keywords form the fundamental pieces of an implicit definition of the unnamed concept. In any case, numerical modeling of a concept can be achieved by taking *the centroid of the keywords* that characterize the concept, i.e., by taking the average of the embedding vectors of the characterizing keywords. The centroid can also be viewed as capturing the commonalities between the concepts represented by the component keywords. We tested the centroid-based concept modeling technique using several examples. The results are shown in Tables 3, 4, and 5.

In Table 3, we show the most similar words to the centroid of {'Fourier', 'Mellin'}. Observe that although 'transform' did not appear in the top-20 similarity matches of 'transform' in Table 1, it appears in the centroid matches, probably because 'transform' is a central commonality between 'Fourier' and 'Mellin'. This illustrates the conceived power of this simple centroid-based technique. By the same token, and indirectly through 'transform', other transforms appear in the top list of centroid matches, such as 'Laplace', 'Hilbert', 'Kontorovich–Lebedev', and so on.

Tables 4 and 5 show more examples, where the top-20 matches are populated by mostly relevant terms. One striking yet unexplainable observation is that the matches tend to be of lengths comparable to those of the component keywords of the centroid. For example, nearly all the matches for the centroid of {'se', 'ce'} in Table 4 have lenght two characters and the remaining matches of three characters. Likewise, in Table 5, the top-20 matches for the centroid of {'Bessel' , 'Struve'} are the names of various similar functions, where the names are of length comparable to (or larger than) that of the two component keywords, even though the one-letter names of specific Bessel functions (e.g., I, J, and K) should be on/near the top of the list. In future work, this phenomenon will be investigated further, and, if length-bias is confirmed, length-agnostic embedding techniques will be sought.

Weighted centroids

The notion of centroids is flexible enough to allow the user to put more, or less, emphasis on certain aspects/keywords in the component list, so as to represent a different concept or a different gradation of a concept. This is achieved through weighted centroids, where more emphasis is put by giving a larger weight to the corresponding keyword, and vice versa. One can even *de-emphasize* certain aspects/dimensions/keywords by giving them *negative* numerical weights.

Table 6 illustrates the power of weighted centroids. In the 1st column, by giving weight 0 to 'Gamma', effectively finding the words similar to 'Euler', the search failed to uncover the relevant Beta (function) in the top 20 hits. By including 'Gamma' with equal weight as 'Euler' (2nd column), 'Beta' was returned (as the 7th hit). Disappointingly, 'Gamma' itself was not returned, and that is because the vector of 'Euler' has a relatively much larger magnitude than the vector of 'Gamma'. This points to the need to use *normalized* embedding vectors instead. In the third column, more weight is given to 'Gamma' than to 'Euler'; as a result, two things were observed: (1) 'Gamma' is in the hit list and near the top, and (2) 'Beta' now ranks higher than when both query words are of equal weight.

Weighted centroids show much promise and potential, and call for further studies to determine techniques for systematically selecting weights to meet certain objects. One of the applications of centroids, whether weighted or unweighted, is query expansion for a more effective search. This is addressed in the next subsection.

It should be noted that weighted averaging is a commonly used technique, and was applied in search fairly early on, as in the Rocchio algorithm in IR used as a tf-idf vector-based feedback approach to revise the user's search query to improve the search outcome (Rocchio 1971). Although both the Rocchio algorithm and our weighted centroid are vector based, they differ in several respects. Our approach is embedding based, and thus carries more contextual/relational semantics than tf-idf based vectors tend to do, and it applies to searching for individual (similar) words (and phrases), rather than just coarser search (for documents), leading to more applications than conventional IR.

Query expansion

One limitation of keyword/keyphrase search is that it is based on the literal occurrence of the terms used in the query, regardless of the underlying semantics. Incorporating a synonym search does not eliminate this deficiency. The *integrated* semantic notions represented by conjoining several characterizing keywords cannot be identified by the mere occurrence of those keywords in a document. Rather, it is more effective first to determine new keywords that pertain very closely to the concept represented/shared by certain characterizing keywords, and then add those new keywords to the original keywords to form an expanded query. In other terms, it is significantly more effective to *explicitize* the concept that is underlying a set of keywords into new keywords to add to the search query.

This query expansion process is quite straightforward thanks to the embedding-based centroid concept introduced earlier in this section. Specifically, the query expansion is done through the following steps:

1. **Embedding**: Retrieve the embedding vectors of the keywords of the query;
2. **Centroid**: Compute the (weighted) centroid C of those vectors;
3. **Similarity Search**: Find the top N most similar vectors to C (N is preset);
4. **Expansion Keywords**: Retrieve the words corresponding to those vectors;
5. **Expansion-keywords Selection** [optional]: In human-in-the-middle situations, where the user has domain expertise, the user selects from the list of the previous step the words that truly pertain to the keywords of the query;
6. **Expanded Query**: Add the words of the previous step to the keywords of the original query, conjoined by the OR or the AND boolean operator.

We performed several preliminary tests of query expansion on 4–5 queries using DLMF and its math search capability and following the steps outlined above. We carefully selected the expansion keywords using our familiarity with DLMF and Special Functions. Table 7 shows the original queries and the expanded queries, and reports the recall as *P-Recall* and *E-Recall*, corresponding to page search and equation search, respectively. The recall of the OR-expanded queries increases and this increase indicates that this expansion is not superfluous, i.e., the new keywords are not subsumed by the original query, but it is a judicious expansion that helps the user uncover new, relevant hits that the original query could not match.

Table 7 Query Expansion Using Centroids and Similarity

	Query	P-Recall	E-Recall
Original Query:	Bessel	195	858
1st Expansion:	Bessel OR Airy	234	1061
2nd Expansion:	Bessel OR Spherical	220	909
3rd Expansion:	Bessel OR Airy OR Spherical	259	1112
Original Query:	Fourier OR Mellin	84	54
1st Expansion:	Fourier OR Mellin OR transform	189	90
2nd Expansion:	Fourier OR Mellin OR transform OR Convolution	192	92
Original Query:	Euler	381	1513
Original Query:	Gamma	220	879
"Inspired" Query:	Beta	179	543
1st Expansion:	Euler OR Gamma	381	1515
2nd Expansion:	Euler OR Beta	420	1817
3rd Expansion:	Euler OR Gamma OR Beta	420	1819
4th Expansion:	Euler AND Gamma	220	877
5th Expansion:	Euler AND Beta	79	543
6th Expansion:	Euler AND Gamma AND Beta	62	38

The reported recalls are derived from running the queries on the DLMF (DLMF 2018) in two modes: (1) page search (for P-Recall) and (2) equation search (for E-Recall)

Note that there are applications where users have domain expertise for a judicious selection of expansion keywords. For example, frequently, users of math search are domain experts in their disciplines. Another example is patent search, where patent examiners are domain experts who search for *prior art* through a sequence of queries that they evolve by expansion. Our proposed query expansion method suggests valuable keywords to add, and the patent examiners have the expertise to wisely choose from the suggested keywords to form better queries for more effective and more efficient search of prior arts. This method has already been tested by the US Patent & Trademark Office (US PTO) and has shown good results (Krishna et al. 2019).

The expansion is not limited to an OR-expansion for an increased recall, but it can suggest valuable new keywords to intersect with the old keywords using AND-expansion, leading to better precision.

Another use of weighted centroids is relevance ranking, especially in search systems that do not allow the user to weight the query keywords. For example, to give more, or less, emphasis to some keywords in the query, one can compute an appropriately weighted centroid of the query keywords. The new keywords, resulting from Step 3 of the algorithm, capture the inherent emphasis. By expanding the original query with those properly biased new keywords, the hits will be ranked in a way that reflects that bias, leading to better relevance ranking and higher user satisfaction. Of course, further, more extensive, and more systematic study of this novel form of query expansion is needed and will be included in future work.

Note that query expansion has received considerable attention (Vechtomova 2009) over the years, and some of the authors of this paper have addressed that topic in the context of math search (Youssef 2005; Al-Tamimi and Youssef 2007) and incorporated their findings and techniques into the deployed math search of the Digital Library of Mathematical

Functions (DLMF 2018). For example, in math search, we make use of not only thesaurus-based query expansions, but we also turn expression queries into phrase queries with stretchable proximity operators to allow for intervening symbols in the matches even if those symbols are missing from the query expression. That way, a query "$\sin^2 + \cos^2$" will match an expression like "$\sin^2 x + \cos^2 x$".

More recently, embedding-based query expansion started to receive some attention (Kuzi et al. 2016; Almasri et al. 2016). However, *weighted*-centroid based query expansion is novel, and no study of it has been done, either in general search or in the context of math search. Of course, query expansion can take other forms beyond synonyms and keywords-implied latent concepts, using language models and topic models (Croft et al. 2009). Our weighted-centroid based query expansion should not be viewed as an alternative to those other query expansion techniques or math-query expansion methods. Instead, it can be synergistically combined with all those query expansion methods.

Accordingly, in our future work, we will elaborate, study and evaluate weighted-centroid based query expansion, on its own as well as in combination with other query expansions methods.

Semantic knowledge extraction

In the "Term Analogy" section, we analyzed the effectiveness of term similarity, term analogy, and query expansion when searching for entire mathematical expressions instead of descriptive keywords. To further investigate the MathIR task, we also explored how descriptive phrases for mathematical objects can be used to improve our results. Extracting definiens of mathematical objects from a textual context is a common task in MathIR (Pagel and Schubotz 2014; Schubotz et al. 2016; Zanibbi et al. 2016a; Schubotz et al. 2017; Kristianto et al. 2017) that often provides a gold dataset for its evaluation. Since the DLMF does not provide extensive textual information for its mathematical expressions, we considered an alternative scenario in our analysis, one in which we trained a second word2vec model on a much larger corpus composed of articles/papers from the arXiv collection. In this section, we compare our findings against Schubotz et al. (2017)'s approach. We apply variations of a word2vec (Mikolov et al. 2013b) and paragraph vectors (Le and Mikolov 2014) implementation to extract mathematical relations from the arXMLiv dataset (Ginev 2018) (i.e., an HTML collection of the arXiv.org preprint archive,[13]), which is used as our training corpus. We also consider the subsets that do not report errors during the document conversion (i.e., *no_problem* and *warning*) which represent 70% of archive.org. We make the code, regarding our experiments, publicly available.[14]

Evaluation of math-embedding-based knowledge extraction

As a pre-processing step, we represent mathematical expressions using the MathML[15] notation. First, we replace all mathematical expressions with the identifiers sequence it contains, i.e., $W(2, k)$ is replaced by '$W\ k$'. We also add the prefix 'math-' to all identifier

---

[13] https://arxiv.org/ [Accessed Feb. 2020].

[14] https://github.com/ag-gipp/math2vec [Accessed Feb. 2020].

[15] The source TeX file has to use mathematical environments for its expressions.

Table 8 Analogies of the form: Find the *Term* where *Term* is a word that is to X what Y is to Z

Top-10 best *Terms* and their cosine similarities where					
Term is to *f* what 'variable' is to *x*		*Term* is to *f* what 'variable' is to *y*		*Term* is to *f* what 'variable' is to *a*	
variables	0.7655	variables	0.7481	variables	0.7600
independent	0.7411	**function**	0.7249	**function**	0.7154
appropriate	0.7279	given	0.7103	appropriate	0.6925
means	0.7250	means	0.7083	independent	0.6789
ie	0.7234	ie	0.7067	instead	0.6784
instead	0.7233	independent	0.7030	defined	0.6729
namely	0.7139	thus	0.6925	namely	0.6719
function	0.7131	instead	0.6922	continuous	0.6707
following	0.7117	appropriate	0.6891	depends	0.6629
depends	0.7095	defined	0.6889	represents	0.6623

tokens to distinguish between textual and mathematical terms later. Second, we remove all common English stopwords from the training corpus. Finally, we train a word2vec model (skip-gram) using the following hyperparameters:[16] vector size of 300 dimensions, a window size of 15, minimum word count of 10, and a negative sampling of $1E − 5$. We justify the hyperparameter used in our experiments based on previous publications using similar models (Mikolov et al. 2013a; Le and Mikolov 2014; Lau and Baldwin 2016; Caselles-Dupré et al. 2018; Ruas et al. 2019).

In the following, distances between vectors are calculated via the cosine distance. The trained model was able to partially incorporate semantics of mathematical identifiers. For instance, the closest 27 vectors to the mathematical identifier *f* are mathematical identifiers themselves and the fourth closest noun vector to *f* is '*function*'. We observe that the results of the model trained on arXiv are comparable with our previous experiments on the DLMF.

In the "Term Analogy" section, we used the semantic relations, see Eq. 4, between embedding vectors to search for relevant terms in the model. Hereafter, we will refer to this algebraic property as *semantic distance* to a given term with respect to a given relation, i.e., two related vectors. For example, to answer the previously mentioned query/question: What is to 'complex' as *x* is to 'real', one has to find the closest *semantic vectors* to 'complex' with respect to the relation between *x* and 'real', i.e., finding **v** in

$$\mathbf{v} − \mathbf{v}_{\text{complex}} \approx \mathbf{v}_x − \mathbf{v}_{\text{real}}.$$

Instead of asking for mathematical expressions, we will now reword the query to ask for specific words. For example, to retrieve the meaning of *f* from the model, we can ask for: What is to *f* as 'variable' is to *x*? Or in other words, what is semantically close to *f* with respect to the relation between 'variable' and *x*? Table 8 shows the top 10 semantically closest results to *f* with respect to the relations between $\mathbf{v}_{\text{variable}}$ and $\mathbf{v}_x$, $\mathbf{v}_{\text{variable}}$ and $\mathbf{v}_y$, and $\mathbf{v}_{\text{variable}}$ and $\mathbf{v}_a$.

Table 9 The cosine distances of f regarding to the hits in Table 8

Cosine distances between the Terms from Table 8 to f.	
function	0.8138
defined	0.7932
independent	0.7323
namely	0.7214
depends	0.7022
represents	0.6983
instead	0.6837
appropriate	0.6698
continuous	0.6203
variables	0.5638

From Table 8, we can observe a similar behaviour as seen in the "Term Analogy" section. Later, we will explore that mathematical vectors build a cluster in the trained model, i.e., that the vectors of $\mathbf{v}_f$, $\mathbf{v}_x$, and $\mathbf{v}_y$ are close to each other with respect to the cosine similarity. This cluster, and the fact that we did not use stemming and lemmatization for preprocessing, explains that the top hit to the queries is always 'variables'. To refine the order of the extracted answers, we calculated the cosine similarity between $\mathbf{v}_f$ and the vectors for the extracted words directly. Table 9 shows the cosine distances between $\mathbf{v}_f$ and the extracted words from the query: *Term* is to f what 'variable' is to a.

Asking for the meaning of f is a very generic question. Thus, we performed a detailed evaluation on the first 100 entries[17] of the *MathMLBen* benchmark (Schubotz et al. 2018). We evaluated the average of the *semantic distances* with respect to the relations between $\mathbf{v}_{variable}$ and $\mathbf{v}_x$, $\mathbf{v}_{variable}$ and $\mathbf{v}_a$, and $\mathbf{v}_{function}$ and $\mathbf{v}_f$. We have chosen to test on these relations because we believe that these relations are the most general and still applicable, e.g., seen in Table 9. In addition, we consider only results with a cosine similarity equal to or greater than 0.70 to maintain a minimum quality in our experiments. The overall results were poor, with a precision of $p = .0023$ and a recall of $r = .052$. Despite the weak results, an investigation of some specific examples showed interesting characteristics; for example, for the identifier W, the four semantically closest results were *functions*, *variables*, *form*, and the mathematical identifier q. The poor performance illustrates that there might be underlying issues with our approach. However, as mentioned before, mathematical notation is highly flexible and content-dependent. Hence, in the next section, we explore a technique that rearranges the hits according to the actual close context of the mathematical expression.

Improvement by considering the context

We also investigate how a different word embedding technique would affect our experiments. To do so, we trained a Distributed Bag-of-Words of Paragraph Vectors (DBOW-PV) (Le and Mikolov 2014) model. We trained this DBOW-PV in the same corpus as our word2vec model (with the same preprocessing steps) with the following configuration: 400 dimensions, a window size of 25, and minimum count of 10 words. In Schubotz et al.

---

[17] Same entries used in Schubotz et al. (2017).

Table 10 We are looking for descriptive terms for f in a given context '*Let f(x, y) be a continuous function where x and y are arbitrary values*'. To achieve this, we retrieved close vectors to f and computed their distances to the given context sentence. To bring variety to the context, we used our DBOW-PV model to retrieve related sentences to the given context and computed the average distance of the words to these related sentences

Top-10 closes words (no math symbols) to f and their cosine similarities.		After reordering the hits according to their distances to the context vector.	
given	0.8162	case	0.8568
case	0.7960	corresponding	0.8562
corresponding	0.7957	note	0.8451
function	0.7900	thus	0.8414
note	0.7803	obtain	0.8413
thus	0.7726	ie	0.8335
obtain	0.7712	since	0.8250
value	0.7682	**function**	0.8086
ie	0.7656	value	0.8015
since	0.7583	given	0.7096

(2017), they analyze all occurrences of mathematical identifiers and consider the entire article at once. We believe this prevents the algorithm from finding the right descriptor in the text, since later or prior occurrences of an identifier might appear in a different context, and potentially introduce different meanings. Instead of using the entire document, we consider the algorithm of Schubotz et al. (2017) only in the input paragraph and similar paragraphs given by our DBOW-PV model. Unfortunately, the obtained variance within the paragraphs brings a high number of false positives to the list of candidates, which affects the performance of the original approach negatively.

As a second approach for improving our system, we considered a given textual context to reorder extracted words according to their cosine similarities to the given context. For example, consider the sentence: 'Let $f(x, y)$ be a continuous function where x and y are arbitrary values.'. We ask for the meaning of f concerning this given context sentence. The top-k closest words to f in the word2vec model only represent the distance over the entire corpus, in this case, arXiv, but not regarding a given context. To address this issue, we retrieved similar paragraphs to our context example via the DBOW-PV model and computed the weighted average distance between all top-k words, that are similar to f and the retrieved sentences. We expected that the word describing f in our example sentence would also present a higher cosine similarity to the context itself. Table 10 shows the top-10 closest words (i.e., we filtered out other math tokens) and their cosine similarity to f in the left column. The right column shows the average cosine similarities of the extracted words to the context example sentence we used and its retrieved similar sentences.

As Table 10 illustrates, this context-sensitive approach was not beneficial; in fact it undermined our model. According to the fact that the identifier should be closer to the given context sentence rather than to the related sentences retrieved from the DBOW-PV model, we also explored the use of weighted average. However, the weighted average did not improve the results of the normal average. Other hyperparameters for the word embedding models were also tested in an attempt to tune our system. However, we could not determine any drastic changes regarding the measured performances.

Fig. 1 t-SNE plot of top-1000 closest vectors of the identifier *f* with perplexity values 5 (top left), 10 (top right), 40 (bottom left), and 100 (bottom right) and the default values of the TSNE python package for other settings

Visualizing our model

Figure 1 illustrates four t-SNE plots of our word2vec model. Since t-SNE plots may produce misleading structures (Wattenberg et al. 2016), we plot four t-SNE plots with different perplexity values. Other parameters were set to their default values according to the TSNE python package. We colored word tokens in blue and math tokens in red. The plots illustrate, though not surprisingly, that math tokens are clustered together. However, a certain subset of math tokens appear isolated from other math tokens. By attaching the content to some of the vectors, we can see that math tokens, such as *and* (an *and* within math mode) and *im* (most likely referring to imaginary numbers) are part of a second cluster of math tokens. The plot is similar to the visualized model presented by Gao et al. (2017), even though they use a different word embedding technique. Hence, the general structure within math word2vec models seems to be insensitive to the embedding technique of formulae used. Compared to Gao et al. (2017), we provide a model with richer details that reveal some dense clusters, e.g., numbers (bottom right plot at (11, 8)) or equation labels (bottom right plot at (-14, 0)).

Based on the presented results, one can still argue that more settings should be explored (e.g., different parameters and embedding techniques) for the embedding phase, different pre-processing steps (e.g., stemming and lemmatization) should be adopted, and post-processing techniques (e.g., boosting terms of interest based on a knowledge database such as OntoMathPro (Elizarov et al. 2017)) should also be investigated. This presumably solves

some minor problems, such as removing the inaccurate first hit in Table 8. Nevertheless, the overall results would not surpass the ones in Schubotz et al. (2017), which reports a precision score of $p = 0.48$. On the grounds that mathematics is highly customizable, many of the defined relations between mathematical concepts and their descriptors are only valid in a local scope. Let us consider an author that notates his algorithm using the symbol π. The author's specific use of π does not change its general use, but it affects the meaning in the scope of the article. Current ML approaches only learn patterns of most frequently used combinations, e.g., between f and 'function', as seen in Table 8.

Even though math notations can change, such as π in the example above, one could assume the existence of a common ground for most notations. The low performance of our experiments compared to the results in Schubotz et al. (2017) seem to confirm that math notations change regularly in real-world documents, i.e., are tied to a specific context. If a common ground exists, for math notations, it must be marginally small, at least in the 100 test cases from Schubotz et al. (2018).

Overcoming issues of knowledge extraction

We assume the low performance regarding our knowledge extraction experiments are caused by fundamental issues that should be discussed before more efforts are made to train ML algorithms for extracting knowledge of math expressions. In the following, we discuss some reasons that we believe can help ML algorithms to understand mathematics better.

It is reported that 70% of mathematical symbols are explicitly declared in the context (Wolska and Grigore 2010). Only four reasons justify an explicit declaration in the context: (a) a new mathematical symbol is defined, (b) a known notation is changed, (c) used symbols are present in other contexts and require specifications to be correctly interpreted, or (d) authors' declarations are redundant (e.g., for improving readability). We assume (d) is a rare scenario compared to the other ones (a–c), except in educational literature. Current math-embedding techniques can learn semantic connections only in that 70%, where the definiens is available. Besides (d), the algorithm would learn either rare notations (in case of (a)) or ambiguous notations (in cases (b–c)). The flexibility that mathematical documents allow to (re)define used mathematical notations further corroborates the complexity of learning mathematics.

Learning algorithms would benefit from literature focused on (a) and (d), instead of (b) and (c). Similar to students who start to learn mathematics, ML algorithms have to consider the structure of the content they learn. It is hard to learn mathematics only considering arXiv documents without prior or complementary knowledge. Usually, these documents represent state-of-the-art findings containing new and unusual notations and lack of extensive explanations (e.g., due to page limitations). In contrast, educational books carefully and extensively explain new concepts. We assume better results can be obtained if ML algorithms are trained in multiple stages, first on educational literature, then on datasets of advanced math articles. A basic model trained in educational literature should capture standard relations between mathematical concepts and descriptors. This model should also be able to capture patterns independently of how new or unusual the notations are present in the literature. In 2014, Matsuzaki et al. (2014) presented some promising results to answer mathematical questions from Japanese university entrance exams automatically. While the approach involves many manual adjustments and analysis, the promising results

illustrate the different levels of knowledge that is still required for understanding arXiv documents vs. university entrance level exams. A well-structured digital mathematical library that distinguishes the different levels of sophistication in articles (e.g. introductions vs. state-of-the-art publications) would also benefit mathematical machine learning tasks.

The lack of references and applications that provide a solid semantic structure of natural language for mathematical identifiers make the disambiguation process of the latter even more challenging. In natural texts, one can try to infer the most suitable word sense for a word based on the lemma[18] itself, the adjacent words, dictionaries, and thesauri to name a few. However, in the mathematical arena, the scarcity of resources and the flexibility of redefining their identifiers make this issue much harder. The context text preceding or following the mathematical equation is essential for its understanding. This context can be considered in a long or short distance away from the equation, which aggravates the problem. Thus, a comprehensive annotated dataset that addresses these needs of structural knowledge would enable further progress in MathIR with the help of ML algorithms.

Another primary source of complexity is the inherent ambiguity present in any language, especially in mathematics. A typical workaround in linguistics for such ambiguous notations is to consider the use of lexical databases (e.g., WordNet Miller 1995; Fellbaum 1998) to identify the most suitable word senses for a given word. These databases allow embeddings algorithms to train a vector for each semantic meaning for every token. For example, *Java* could have multiple vectors in a single model according to its different meanings of the word, e.g., the island in the south of Indonesia, the programming language or the coffee beans. However, mathematics lacks such systems, which makes its adoption not feasible at the moment. In Youssef (2017), they propose the use of tags, similarly to the POS tags in linguistics, but for tagging mathematical TeX tokens, bringing more information to the tokens considered. As a result, a lexicon containing several meanings for a large set of mathematical symbols is developed. OntoMathPro (Elizarov et al. 2017) aims for generating a comprehensive ontology of mathematical knowledge and, therefore, also contain information about the different meanings of mathematical tokens. Such dictionaries might enable the disambiguation approaches in linguistics to be used in mathematical embedding in the near future.

Another issue in recent publications is the lack of standards and the scarcity of benchmarks to properly evaluate MathIR algorithms. Krstovski and Blei (2018); Yasunaga and Lafferty (2019) provide an interesting perspective on the problem of mathematic embeddings. Their experiments are focused on math-analogies. Our findings on "Term Analogy" section corroborate with the math-analogies results, as our experiments have comparable results in a controlled environment. However, because of a missing well-established benchmark, we, as well the mentioned publications, are only able to provide incipient results. Existing datasets are often created for and, therefore, limited to specific tasks. For example, the NTCIR math tasks (Aizawa et al. 2013, 2014; Zanibbi et al. 2016a) or the upcoming ARQMath[19] task, provide datasets that are specifically designed to tackle problems of mathematical search engines. The secondary task of ARQMath actually search for math-analogies. In general, a proper, common standard for interpreting semantic structures of mathematics (see for example the mentioned problems with α_i in "Foundations and Related

---

[18] Canonical form, dictionary form, or citation form of a set of words.
[19] https://www.cs.rit.edu/~dprl/ARQMath/ [Accessed Feb. 2020].

Work" section) would be beneficial for several tasks in MathIR, such as semantic knowledge extraction.

Conclusions

In this paper, we explored the use and effectiveness of word embedding for MLP on five fundamental tasks, namely, (i) term similarity, (ii) math analogies, (iii) concept modeling, (iv) query expansion using a novel centroid-based technique, and (v) semantic knowledge extraction. While we were able to produce promising results regarding the tasks (i) to (iv), we evaluated that task (v), semantic knowledge extraction, is compromised even when considering robust techniques borrowed from the NLP domain. After experimenting with popular mathematical representations in MathIR, we exposed fundamental problems that prevent ML algorithms from learning mathematics. We also discovered the same problems in several related research projects. Many of these projects show promising examples without extensive evaluations, motivating more researchers to follow the same idea with similar fundamental issues. To address this problem, we discussed some of the reasons that we believe thwart the progress in MathIR in the direction of machine learning.

As we explored through this paper, our preliminary results stress the urgent need for creating extensive math-specific benchmarks for testing math embedding techniques on math-specific tasks. The authors plan to begin developing some of such benchmarks and to pursue further investigations of some of the lines of inquiry indicated in the paper. Developing such a benchmark will also enable us to extensively evaluate our exploratory results in this paper.

To appreciate more the magnitude and dimensions of creating such benchmarks, it is instructive to look at some of those developed for NLP whose tasks can beneficially inform and guide corresponding tasks in MLP. The NLP benchmarks include one for natural language inference (Bowman et al. 2015), one for machine comprehension (Rajpurkar et al. 2016), one for semantic role modeling (Palmer et al. 2005), and one for language modeling (Chelba et al. 2014), to name a few. With such benchmarks, which are often *de facto* standards for the corresponding NLP tasks, the NLP research community has been able to (1) measure the performance of new techniques up to statistical significance, and (2) track progress in various NLP techniques, including deep learning for NLP, by quickly comparing the performance of new techniques to others and to the state-of-the-art.

While our exploratory studies regarding our term similarities, analogies, and query expansions need extensive future experimentation for statistically significant validation on large datasets and benchmarks, they show some of the promise and limitations of word embedding in math (MLP) applications. Future investigations will also examine the effectiveness of other embedding techniques, as well as more MLP tasks and applications such as part-of-math tagging, math-term disambiguation, and representation-to-computation deep learning models, to name a few.

Compliance with ethical standards

References

Aizawa, A., Kohlhase, M., & Ounis, I. (2013). *NTCIR-10 math pilot task overview* (pp. 654–661).

Aizawa, A., Kohlhase, M., Ounis, I., & Schubotz, M. (2014). NTCIR-11 math-2 task overview. In *Proceedings of the 11th NTCIR conference on evaluation of information access technologies*, NTCIR-11, National Center of Sciences, Tokyo, Japan, December 9–12, 2014 (pp. 88–98). National Institute of Informatics (NII).

Al-Tamimi, M., & Youssef, A. (2007). An extensive math query language. In *The ISCA 16th international conference on software engineering and data engineering (SEDE-2007)*.

Almasri, M., Berrut, C., & Chevallet, J.-P. (2016). A comparison of deep learning based query expansion with pseudo-relevance feedback and mutual information. In *ECIR* (pp. 709–715).

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research, 3*, 1137–1155.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics, 5*, 135–146.

Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of 2015 conference on empirical methods in natural language processing (EMNLP)*, Lisbon, Portugal (pp. 632–642). ACL.

Camacho-Collados, J., Pilehvar, M. T., & Navigli, R. (2015). A unified multilingual semantic representation of concepts. In *Proceedings of 53rd annual meeting of the association for computational linguistics (ACL)*, Beijing, China (pp. 741–751). ACL.

Caselles-Dupré, H., Lesaint, F., & Royo-Letelier, J. (2018). Word2vec applied to recommendation: Hyperparameters matter. In S. Pera, M. D. Ekstrand, X. Amatriain, & J. O'Donovan (Eds.), *Proceedings of the 12th ACM conference on recommender systems*, RecSys 2018, Vancouver, BC, Canada, October 2–7, 2018 (pp. 352–356). ACM.

Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Strope, B., & Kurzweil, R. (2018). Universal sentence encoder for English. In *Proceedings of conference on empirical methods in natural language processing (EMNLP)*, Brussels, Belgium (pp. 169–174). ACL.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., & Robinson, T. (2014). One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH 2014, 15th annual conference of the international speech communication association*, Singapore (pp. 2635–2639). ISCA.

Chen, Q., Zhu, X., Ling, Z., Wei, S., Jiang, H., & Inkpen, D. (2017). Enhanced LSTM for natural language inference. In *Proceedings of the 55th annual meeting of the association for computational linguistics (ACL)*, Vancouver, Canada, Volume 1: Long Papers (pp. 1657–1668). ACL.

Chiu, J. P. C., & Nichols, E. (2016). Named entity recognition with bidirectional LSTM-CNNS. *TACL, 4*, 357–370.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, Doha, Qatar (pp. 1724–1734). ACL.

Clark, C., & Gardner, M. (2018). Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th annual meeting of the association for computational linguistics (ACL)*, Melbourne, Australia, Volume 1 (pp. 845–855). ACL.

Croft, W. B., Metzler, D., & Strohman, T. (2009). *Search engines—Information retrieval in practice*.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th conference of the association for computational linguistics (ACL)*, Florence, Italy, July 28–August 2, 2019, Volume 1 (pp. 2978–2988). ACL.

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Human language technologies (NAACL-HLT)*, Minneapolis, MN, USA, June 2–7, 2019, Volume 1 (pp. 4171–4186). ACL.

Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R. M., & Makhoul, J. (2014). Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd annual meeting of the association for computational linguistics*, ACL, Baltimore, MD, USA, Volume 1: Long Papers (pp. 1370–1380). ACL.

DLMF. (2018). *NIST Digital Library of Mathematical Functions*. http://dlmf.nist.gov/, Release 1.0.20 of 2018-09-1. F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, & M. A. McClain (Eds).

Elizarov, A., Kirillovich, A., Lipachev, E., & Nevzorova, O. (2017). Ontomath digital ecosystem: Ontologies, mathematical knowledge analytics and management. *CoRR*. arXiv:1702.05112.

Fellbaum, C. (1998). A semantic network of English: The mother of all wordnets. *Computers and the Humanities*, *32*(2), 209–220.

Gao, L., Jiang, Z., Yin, Y., Yuan, K., Yan, Z., & Tang, Z. (2017). Preliminary exploration of formula embedding for mathematical information retrieval: Can mathematical formulae be embedded like a natural language? *CoRR*. arXiv:1707.05154.

Ginev, D. (2018). arxmliv:08.2018 dataset, an html5 conversion of arxiv.org. SIGMathLing—Special Interest Group on Math Linguistics.

Gong, Y., Luo, H., & Zhang, J. (2018). Natural language inference over interaction space. In *6th international conference on learning representations*, ICLR 2018, Vancouver, BC, Canada, Conference Track Proceedings. OpenReview.net.

Greiner-Petter, A., Ruas, T., Schubotz, M., Aizawa, A., Grosky, W. I., & Gipp, B. (2019). Why machines cannot learn mathematics, yet. In *Proceedings of 4th BIRNDL workshop at 42nd SIGIR*, Paris, France (pp. 130–137). CEUR-WS.org.

He, L., Lee, K., Lewis, M., & Zettlemoyer, L. (2017). Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th annual meeting of the association for computational linguistics*, ACL 2017, Vancouver, Canada, July 30–August 4, Volume 1: Long Papers (pp. 473–483).

Iacobacci, I., Pilehvar, M. T., & Navigli, R. (2015). Sensembed: Learning sense embeddings for word and relational similarity. In *Proceedings of 53rd annual meeting of the association for computational linguistics (ACL)*, Beijing, China (Vol. 1, pp. 95–105). ACL.

Iacobacci, I., Pilehvar, M. T., & Navigli, R. (2016). Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of 54th annual meeting of the association for computational linguistics (ACL)*, Berlin, Germany (Vol. 1). ACL.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 conference on empirical methods in natural language processing*, EMNLP 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL (pp. 1746–1751). ACL.

Kohlhase, M. (2017). Math object identifiers—Towards research data in mathematics. In M. Leyer (Ed.), *Lernen, Wissen, Daten, Analysen (LWDA) conference proceedings*, Rostock, Germany, September 11–13, 2017, *CEUR Workshop Proceedings* (Vol. 1917, p. 241). CEUR-WS.org.

Krishna, A., Jin, Y., Foster, C., Youssef, A., & Gabel, G. (2019). Query expansion for patent searching using word embedding and professional crowdsourcing. In *2019 AAAI fall symposium series, FSS-19, Washington, DC*. AAAI Press.

Kristianto, G. Y., Topic, G., & Aizawa, A. (2014). Extracting textual descriptions of mathematical expressions in scientific papers. *D-Lib Magazine, 20*(11/12), 9.

Kristianto, G. Y., Topic, G., & Aizawa, A. (2017). Utilizing dependency relationships between math expressions in math IR. *Information Retrieval Journal, 20*(2), 132–167.

Krstovski, K., & Blei, D. M. (2018). Equation embeddings. *CoRR*. arXiv:1803.09123.

Kuzi, S., Shtok, A., & Kurland, O. (2016). Query expansion using word embeddings. In *the 25th ACM international on conference on information and knowledge management (CIKM '16)* (pp. 1929–1932).

Lau, J. H., & Baldwin, T. (2016). An empirical evaluation of doc2vec with practical insights into document embedding generation. In P. Blunsom, K. Cho, S. B. Cohen, E. Grefenstette, K. M. Hermann, L. Rimell, J. Weston, & S. W. Yih (Eds.), *Proceedings of the 1st workshop on representation learning for NLP*, Rep4NLP@ACL 2016, Berlin, Germany, August 11, 2016 (pp. 78–86). ACL.

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st international conference on machine learning (ICML)*, Beijing, China (pp. II–1188–II–1196). JMLR.org.

Lee, K., He, L., Lewis, M., & Zettlemoyer, L. (2017). End-to-end neural coreference resolution. In *Proceedings of the 2017 conference on empirical methods in natural language processing (EMNLP)*, Copenhagen, Denmark (pp. 188–197). ACL.

Li, J., & Jurafsky, D. (2015). Do multi-sense embeddings improve natural language understanding? In *Proceedings of conference on empirical methods in natural language processing (EMNLP)*, Lisbon, Portugal (pp. 1722–1732). ACL.

Liu, X., Shen, Y., Duh, K., & Gao, J. (2018). Stochastic answer networks for machine reading comprehension. In *Proceedings of the 56th annual meeting of the association for computational linguistics (ACL)*, Melbourne, Australia, July 15–20, 2018, Volume 1 (pp. 1694–1704). ACL.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., et al. (2019). Roberta: A robustly optimized BERT pretraining approach. *CoRR*. arXiv:1907.11692.

Mancini, M., Camacho-Collados, J., Iacobacci, I., & Navigli, R. (2017). Embedding words and senses together via joint know-ledge-enhanced training. In *Proceedings of 21st conference on computational natural language learning (CoNLL)*, Vancouver, Canada (pp. 100–111). ACL.

Matsuzaki, T., Iwane, H., Anai, H., & Arai, N. H. (2014). The most uncreative examinee: A first step toward wide coverage natural language math problem solving. In C. E. Brodley & P. Stone (Eds.), *Proceedings of twenty-eighth AAAI conference on artificial intelligence*, Québec City, Québec, Canada (pp. 1098–1104). AAAI Press.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. *CoRR*. arXiv:1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th international conference on neural information processing systems (NIPS)*, Volume 2 (pp. 3111–3119). USA: Curran Associates Inc.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the ACM, 38*(11), 39–41.

Neelakantan, A., Shankar, J., Passos, A., & McCallum, A. (2014). Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of conference on empirical methods in natural language processing (EMNLP)*, Doha, Qatar (pp. 1059–1069). ACL.

Pagel, R., & Schubotz, M. (2014). Mathematical language processing project. In *Joint proceedings of the MathUI, OpenMath and ThEdu workshops and work in progress track at CICM co-located with conferences on intelligent computer mathematics (CICM 2014)*, Coimbra, Portugal, *CEUR Workshop Proceedings* (Vol. 1186). CEUR-WS.org.

Palmer, M., Kingsbury, P., & Gildea, D. (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics, 31*(1), 71–106.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of conference on empirical methods in natural language processing (EMNLP)*, Doha, Qatar, Volume 14 (pp. 1532–1543). ACL.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: Human language technologies (NAACL-HLT)*, New Orleans, Louisiana, USA, June 1–6, 2018, Volume 1 (pp. 2227–2237). ACL.

Pilehvar, M. T., Camacho-Collados, J., Navigli, R., & Collier, N. (2017). Towards a seamless integration of word senses into downstream NLP applications. *CoRR*. arXiv:1710.06632.

Pilehvar, M. T., & Collier, N. (2016). De-conflated semantic representations. In *Proceedings of conference on empirical methods in natural language processing (EMNLP),* Austin, Texas, USA (pp. 1680–1690). ACL.

Raganato, A., Bovi, C. D., & Navigli, R. (2017). Neural sequence learning models for word sense disambiguation. In *Proceedings of the 2017 conference on empirical methods in natural language processing (EMNLP)*, Copenhagen, Denmark, September 9–11, 2017 (pp. 1156–1167). ACL.

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing (EMNLP)*, Austin, Texas, USA (pp. 2383–2392). ACL.

Řehůřek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks*, Valletta, Malta (pp. 45–50). ELRA. http://is.muni.cz/publication/884893/en.

Rocchio, J. (1971). Relevance feedback in information retrieval. In *The SMART retrieval system—Experiments in automatic document processing* (pp. 313–323). Upper Saddle River: Prentice-Hall Inc.

Ruas, T., Grosky, W., & Aizawa, A. (2019). Multi-sense embeddings through a word sense disambiguation process. *Expert Systems with Applications, 136*(1), 288–303.

Ruas T, Ferreira CHP, Grosky W, de França FO, de Medeiros DMR (2020) Enhanced word embeddings using multi-semantic representation through lexical chains. *Information Sciences, 532*, 16–32.

Rudolph, M. R., Ruiz, F. J. R., Athey, S., & Blei, D. M. (2017). Structured embedding models for grouped data. In *Advances in neural information processing systems 30: Annual conference on neural information processing systems 2017*, Long Beach, CA, USA (pp. 251–261). Curran Associates, Inc.

Schubotz, M., Greiner-Petter, A., Scharpf, P., Meuschke, N., Cohl, H. S., & Gipp, B. (2018). Improving the representation and conversion of mathematical formulae by considering their textual context. In *Proceedings of the 18th ACM/IEEE on joint conference on digital libraries (JCDL)*, Fort Worth, TX, USA, June 03–07, 2018 (pp. 233–242). ACM.

Schubotz, M., Grigorev, A., Leich, M., Cohl, H. S., Meuschke, N., Gipp, B., Youssef, A. S., & Markl, V. (2016). Semantification of identifiers in mathematics for better math information retrieval. In *Proceedings of the 39th international ACM SIGIR conference on research and development in information retrieval (SIGIR)*, Pisa, Tuscany, Italy (pp. 135–144). ACM.

Schubotz, M., Krämer, L., Meuschke, N., Hamborg, F., & Gipp, B. (2017). Evaluating and improving the extraction of mathematical identifier definitions. In *Experimental IR meets multilinguality, multimodality, and interaction—8th international conference of the CLEF association*, CLEF 2017, Dublin, Ireland, *Lecture notes in computer science* (Vol. 10456, pp. 82–94). Berlin: Springer.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, Seattle, Washington, USA (pp. 1631–1642). ACL.

Vechtomova, O. (2009). *Query expansion for information retrieval*. Boston, MA: Springer.

Wattenberg, M., Viégas, F., & Johnson, I. (2016). How to use t-SNE effectively. *Distill*.

Wiseman, S., Rush, A. M., & Shieber, S. M. (2016). Learning global features for coreference resolution. In *NAACL HLT 2016, The 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*, San Diego California, USA (pp. 994–1004). ACL.

Wolska, M., & Grigore, M. (2010). Symbol declarations in mathematical writing. In P. Sojka (Ed.), *Towards a digital mathematics library* (pp. 119–127). Brno: Masaryk University Press.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32: Annual conference on neural information processing systems 2019*, NeurIPS 2019, 8–14 December 2019, Vancouver, BC, Canada (pp. 5754–5764).

Yasunaga, M., & Lafferty, J. (2019). Topiceq: A joint topic and mathematical equation model for scientific texts. *CoRR*. arXiv:1902.06034.

Youssef, A. (2005). Search of mathematical contents: Issues and methods. In *The ISCA 14th international conference on intelligent and adaptive systems and software engineering (IASSE-2005)*.

Youssef, A. (2017). Part-of-math tagging and applications. In *Intelligent computer mathematics* (pp. 356–374). Cham: Springer.

Youssef, A., & Miller, B. R. (2019). Explorations into the use of word embedding in math search and math semantics. In *Intelligent computer mathematics—12th international conference*, CICM 2019, Prague, Czech Republic (pp. 291–305). Springer.

Zanibbi, R., Aizawa, A., Kohlhase, M., Ounis, I., Topic, G., & Davila, K. (2016a). NTCIR-12 MathIR task overview. In *Proceedings of the 12th NTCIR conference on evaluation of information access technologies*, National Center of Sciences, Tokyo, Japan. National Institute of Informatics (NII).

Zanibbi, R., Davila, K., Kane, A., & Tompa, F. W. (2016b). Multi-stage math formula search: Using appearance-based similarity metrics at scale. In R. Perego, F. Sebastiani, J. A. Aslam, I. Ruthven, & J. Zobel (Eds.), *Proceedings of 39th international ACM SIGIR conference on research and development in information retrieval*, Pisa, Italy (pp. 145–154). ACM.

Zhou, J., & Xu, W. (2015). End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing of the Asian federation of natural language processing (ACL)*, Beijing, China, Volume 1 (pp. 1127–1137). ACL.

Affiliations

André Greiner-Petter[1] (ORCID) **· Abdou Youssef[2,3] · Terry Ruas[1] · Bruce R. Miller[3] ·
Moritz Schubotz[1,4] · Akiko Aizawa[5] · Bela Gipp[1]**

Abdou Youssef
ayoussef@gwu.edu; youssef@nist.gov

Terry Ruas
ruas@uni-wuppertal.de

Bruce R. Miller
miller@nist.gov

Moritz Schubotz
moritz.schubotz@fiz-karlsruhe.de; schubotz@uni-wuppertal.de

Akiko Aizawa
aizawa@nii.ac.jp

Bela Gipp
gipp@uni-wuppertal.de

[1]   University of Wuppertal, Wuppertal, Germany

[2]   The George Washington University, Washington, USA

[3]   Applied and Computational Mathematics Division, NIST, Gaithersburg, MD, USA

[4]   FIZ Karlsruhe, Berlin, Germany

[5]   National Institute of Informatics, Tokyo, Japan