You must strive to find your own voice because the longer you wait to begin, the less likely you are going to find it at all.

John Keating - *Dead Poet Society*

CHAPTER 3

Semantification of Mathematical LaTeX

Contents

In this chapter, we will focus on the research task **II**, i.e., we develop a new semantification process that addresses the issues of existing approaches outlined in the previous chapter. We identified two main issues with existing MathIR approaches for disambiguation and semantification of LaTeX expressions. First, many semantification approaches solely focus on single tokens, such as identifiers, or the entire mathematical expression but miss to enrich the essential subexpressions between both extremes semantically. Second, existing translation approaches lack context sensitivity and disambiguate expressions by following an internal (often hidden)

A. Greiner-Petter, *Making Presentation Math Computable*,
https://doi.org/10.1007/978-3-658-40473-4_3

context-agnostic decision process. This chapter addresses these issues within three parts. First, we elaborate on the capabilities of word embedding techniques to semantically enrich mathematical expressions. Second, we study the frequency distribution of mathematical subexpressions in scientific corpora to understand the variety and complexity of subexpressions better. Third, we briefly outline a context-sensitive translation pipeline based on the gained knowledge from the first two parts.

The primary goal of this chapter is to develop a context-sensitive LaTeX to CAS translation pipeline. Unfortunately, it is not clear where we can find sufficient semantic information in the context to perform reliable translations. We can expect a certain amount of inclusive information in the given expression itself [54, 71, 394]. Additionally, related work has proven that noun phrases in the nearby textual context (such as the leading or following sentences of a formula) can successfully disambiguate math formulae [139, 209, 213, 329]. However, many functions are not necessarily declared in the surrounding context because the author presumes the interpretation is unambiguous. Wolska and Grigore [394] have shown that only around 70% of mathematical identifiers are explicitly declared in the surrounding context. In this case, the location of the information that disambiguates the expression may vary greatly depending on many factors, such as the expected education level of the target audience of the article, the given references in the document, or even the author's preferred notation style. One possible solution for exploiting this source of semantic information is to build a common knowledge database for mathematical expressions.

As a first attempt to automatically build such a common knowledge database that stores the standard, i.e., most common, meanings of mathematical symbols, we explore the capabilities of machine learning algorithms in the first part of this chapter. Specifically, we use word embeddings to train common co-occurrences of mathematical and natural language tokens. We will show that this approach is not as successful as we hoped for our knowledge extraction task but enables new approaches for mathematical search engines. Further, the results will once again underline the issues with the interpretation of nested mathematical objects. Word embeddings for mathematical tokens are mainly unable to properly train the connections with defining expressions in the context because they still ignore the function layer of mathematical expressions. In the following, we focused our studies on mathematical subexpressions.

As a thought experiment, consider mathematical expressions are like entire sentences in natural languages rather than single words. Following this analogy, entire math terms are analog to words, and the notation of mathematical expressions certainly follow a specific grammar [54]. However, our *mathematical sentences* have one distinct difference compared to natural language sentences. The grammar of mathematical expressions is built around a nested structure in contrast to the sequential order of words. For example, a math term representing a variable is a placeholder and can be replaced with arbitrarily complex and deeply nested subexpressions without violating any grammatical rules. This nested structure makes the semantic tokenization of mathematical expressions to a complex and eventually context-dependent task [71, 402]. In order to review our analogy, we perform the most extensive notation analysis of mathematical subexpressions (since those are the potential words) on two real-world scientific datasets. We discovered that the frequency distributions of mathematical objects obey Zipf's law, similar to words in natural language corpora. In turn, we can use frequency-based retrieval functions to distinguish important or informative mathematical objects from stop-word-like structures. We coin these essential and informative objects Mathematical Objects of Interest (MOI). The

success of this new interpretation finally motivated us to move away from the established MathIR techniques that focus on single identifiers or entire math expressions to meaningful subexpressions. Hence, we conclude this chapter with an abstract context-sensitive translation approach that finally attributes to the nested grammar of mathematical formulae and is based on the new concept of MOI.

In summary, this chapter is organized as follows. In Section 3.1, we explore the capabilities of word embeddings to discover common co-occurrences of natural language tokens and math tokens in large scientific datasets. In Section 3.2, we introduce the new concept of MOI and perform the first extensive frequency distribution study of mathematical notations in two large scientific corpora. Section 3.3 concludes the findings of the previous sections by introducing a novel context-sensitive translation approach from LaTeX to CAS expressions. Section 3.1 was published as an article in the Scientometrics journal [15]. Section 3.2 was published as full paper at the WWW conference [14]. Excerpts of Section 3.3 have been published at the ICMS conference in a full paper [10].

3.1 Semantification via Math-Word Embeddings

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 [33, 34, 43, 65, 73, 217, 222, 239, 250, 255, 272, 293, 295] has made it possible to apply deep learning in 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 [149, 412], word-sense disambiguation [160, 305], sentence classification [186], sentiment analysis [344], coreference resolution [223, 388], named entity recognition [72], reading comprehension [75], question answering [234], natural language inference [69, 137], and machine translation [97]. The performance of word embedding in NLP tasks has been measured and shown to deliver fairly high accuracy [256, 293, 295].

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 Mathematical Language Processing (MLP), Mathematical Knowledge Management (MKM), and 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 [121, 9, 215, 400, 404], there is a healthy skepticism about the use of 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 section.

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 [98] and arXiv.org preprint archive[1] 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 section, we provide an exploratory investigation of the effectiveness and use of word embedding in MLP and MKM through different perspectives. First, we train word2vec 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. In the context of this thesis, we are mostly interested in the latter, i.e., knowledge extractions, and will solely focus on these experiments and results. For the tasks (i-iv), see [15].

[1] https://arxiv.org/ [accessed 2019-09-01]

3.1.1 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 [213, 329, 330] 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 [197, 18, 402]. Current definition-languages, such as the content MathML 3.0[2] specification, are often imprecise[3]. 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*[4]. However, in case of the Van der Waerden number, for instance, it is not clear whether W or the sequence $W(r, k)$ should be declared as a 'csymbol'. Another example involves content identifiers, which MathML specifies as *mathematical variables that have properties, but no fixed value*[5]. 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 i-th 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).

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 Mathematical Library (DML)[6], 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* [43, 65, 73, 256, 293, 295, 313]. 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

[2] https://www.w3.org/TR/MathML3/ [accessed 2019-09-01]

[3] 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 2019-09-01]).

[4] https://www.w3.org/TR/\gls{mathml}3/chapter4.html#contm.csymbol [accessed 2019-09-01]

[5] https://www.w3.org/TR/\gls{mathml}3/chapter4.html#contm.ci [accessed 2019-09-01]

[6] https://www.nap.edu/read/18619 [accessed 2019-09-01]

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. [121], they embed single symbols and train a model that can discover similarities between mathematical symbols. Similarly to this approach, Krstovski and Blei [215] uses a variation of word embedding to represent complex mathematical expressions as single unit tokens for IR. In 2019, Yusanaga and Lafferty [400] 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[7] baseline. Equation embedding, as in [121, 215, 400], 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.

3.1.1.1 Word Embedding

In this section, we apply *word2vec* [256] on the DLMF [98] and on the collection of arXiv 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 have close values if the words have similar context, often illustrated by the *king-queen relationship*.

King-Queen Relationship of Word-Embedding Vectors

The king-queen relationship describes the similarity (in terms of the cosine distance between the vectors) of:

$$\vec{v}_{\text{king}} - \vec{v}_{\text{man}} \approx \vec{v}_{\text{queen}} - \vec{v}_{\text{woman}}, \tag{3.1}$$

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

Extending word2vec's approaches, Le and Mikolov [222] propose *Paragraph Vectors*, 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* and *Distributed Bag-of-Words*, 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 [43], ELMo [295], and GloVe [293]. The choice for word2vec for our experiments is justified because of its implementation ease, training speed using modest computing resources,

[7] Latent Dirichlet Allocation

general applicability, and robustness in several NLP tasks [160, 161, 229, 238, 302, 312]. Additionally, in fastText 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[8] 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[9]. Closer to the word2vec technique, GloVe [293] 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 [65] 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 [96] 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* and *Next Sentence Prediction*. 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 [96], XLNet [399], RoBERTa [235], and Transformers-XL [87], in future work.

The overall performance of word embedding algorithms has shown superior results in many different NLP tasks, such as machine translation [256], relation similarity [161], word sense disambiguation [55], word similarity [268, 312], and topic categorization [301]. In the same direction, we also 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.

To investigate the situations described in Sections 3.1.1.1 and 2.2.5 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, we embed all inner tokens, while for the arXiv, we only embed the identifiers. In this thesis, we are more interested in applying math embeddings to semantic extraction task. The MathIR task is described in [15, Section 3].

3.1.2 Semantic Knowledge Extraction

Extracting definiens of mathematical objects from a textual context is a common task in MathIR [214, 279, 329, 330, 405] 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 the approach by Schubotz et al. [330]. We apply varia-

[8] Noise means, the data consists of many uninteresting tokens that affect the trained model negatively.
[9] https://github.com/allenai/bilm-tf [accessed 2019-09-01]

tions of a word2vec [256] and paragraph vectors [222] implementation to extract mathematical relations from the arXMLiv 2018 [132] dataset (i.e., an HTML collection of the arXiv.org preprint archive[10]), 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[11].

3.1.2.1 Evaluation of Math-Embedding-Based Knowledge Extraction

As a pre-processing step, we represent mathematical expressions using the MathML[12] 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 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[13]: 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 [63, 221, 222, 255, 312].

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.

Previously, we used the semantic relations 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 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 $\vec{v}$ in

$$\vec{v} - \vec{v}_{\text{complex}} \approx \vec{v}_x - \vec{v}_{\text{real}}.$$

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 3.1 shows the top 10 semantically closest results to f with respect to the relations between $\vec{v}_{\text{variable}}$ and $\vec{v}_x$, $\vec{v}_{\text{variable}}$ and $\vec{v}_y$, and $\vec{v}_{\text{variable}}$ and $\vec{v}_a$.

From Table 3.1, we can observe a similar behaviour. Later, we will explore that mathematical vectors build a cluster in the trained model, i.e., that the vectors of $\vec{v}_f$, $\vec{v}_x$, and $\vec{v}_y$ 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 $\vec{v}_f$ and the vectors for the extracted words directly. Table 3.2 shows the cosine distances between $\vec{v}_f$ and the extracted words from the query: *Term* is to f what 'variable' is to a.

[10] https://arxiv.org/ [accessed 2019-09-01]
[11] https://github.com/ag-gipp/math2vec [accessed 2019-09-01]
[12] The source TEX file has to use mathematical environments for its expressions.
[13] Non mentioned hyperparameters are used with their default values as described in the Gensim API [307]

Table 3.1: 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

Asking for the meaning of f is a very generic question. Thus, we performed a detailed evaluation on the first 100 entries[14] of the MathMLben benchmark [18]. We evaluated the average of the *semantic distances* with respect to the relations between $\vec{v}_{\text{variable}}$ and $\vec{v}_{\text{x}}$, $\vec{v}_{\text{variable}}$ and $\vec{v}_{\text{a}}$, and $\vec{v}_{\text{function}}$ and $\vec{v}_{\text{f}}$. 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 3.2. 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.

3.1.2.2 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) [222] 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. Schubotz et al. [330] 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 by Schubotz et al. [330] only in the input paragraph and

[14] Same entries used in [330]

Table 3.2: The cosine distances of f regarding to the hits in Table 3.1.

Terms from Table 3.1 to f	Cosine distances between the
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

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 3.3 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 3.3 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.

3.1.2.3 Visualizing Our Model

Figure 3.1 illustrates four t-SNE[154] plots of our word2vec model. Since t-SNE plots may produce misleading structures [382], we plot four t-SNE plots with different perplexity values.

Table 3.3: 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

Other parameters were set to their default values according to the t-SNE 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 in [121], 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 [121], 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 [104, 105]) should also be investigated. This presumably solves some minor problems, such as removing the inaccurate first hit in Table 3.1. Nevertheless, the overall results would not surpass the ones in [330], 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 3.1.

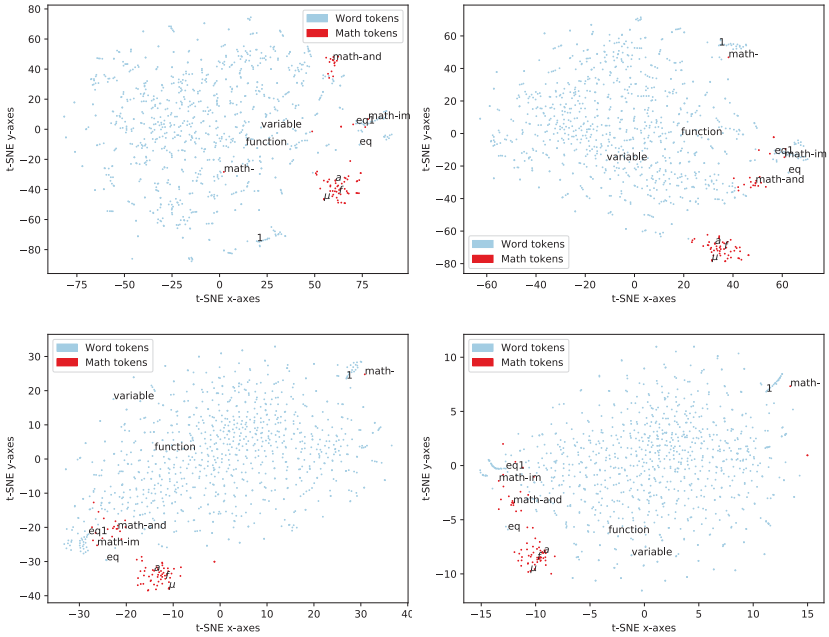


Figure 3.1: t-SNE plot of top-1000 closest vectors of the identifier f with perplexity values 5 (top left), 10 (top right), 40 (bottom left), and 100 (bottom right) and the default values of the t-SNE python package for other settings.

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 [330] 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 [18].

3.1.3 On Overcoming the Issues of Knowledge Extraction Approaches

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 [394]. 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. [247] 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[15] 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 [116, 261]) 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. Youssef [402] proposes 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 [104, 105] aims for generating a comprehensive ontology of mathematical knowledge and, therefore, also contain

[15] canonical form, dictionary form, or citation form of a set of words

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 [215], and Yasunaga and Lafferty [400] provide an interesting perspective on the problem of mathematic embeddings. Their experiments are focused on math-analogies. Our findings on Section 3.2 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 [21, 22, 405] or the upcoming ARQMath[16] 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 Section 2) would be beneficial for several tasks in MathIR, such as semantic knowledge extraction.

3.1.4 The Future of Math Embeddings

As we explored through this section, our preliminary results stress the urgent need for creating extensive math-specific benchmarks for testing math embedding techniques on math-specific tasks. 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 [47], one for machine comprehension [306], one for semantic role modeling [281], and one for language modeling [68], 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. Especially its applicability for our desired knowledge extraction process is highly questionable. One of the main issues we encountered for embedding mathematics is the inability to model the nested semantic structure of mathematical expressions. In the following, we will further explore properties of mathematical subexpressions by analyzing their frequency distributions in large datasets.

3.2 Semantification with Mathematical Objects of Interest

As discussed before, math expressions often contain meaningful and important subexpressions. MathIR [141] applications could benefit from an approach that lies between the extremes of

[16] https://www.cs.rit.edu/~dprl/ARQMath/ [accessed 2020-02-01]

examining only individual symbols or considering an entire equation as one entity. Consider for example, the explicit definition for Jacobi polynomials [98, (18.5.7)]

The Explicit Definition of Jacobi Polynomials

$$P_n^{(\alpha,\beta)}(x) = \frac{\Gamma(\alpha+n+1)}{n!\,\Gamma(\alpha+\beta+n+1)} \sum_{m=0}^{n} \binom{n}{m} \frac{\Gamma(\alpha+\beta+n+m+1)}{\Gamma(\alpha+m+1)} \left(\frac{x-1}{2}\right)^m \tag{3.2}$$

The *interesting* components in this equation are $P_n^{(\alpha,\beta)}(x)$ on the left-hand side, and the appearance of the gamma function $\Gamma(s)$ on the right-hand side, implying a direct relationship between Jacobi polynomials and the gamma function. Considering the entire expression as a single object misses this important relationship. On the other hand, focusing on single symbols can result in the misleading interpretation of Γ as a variable and $\Gamma(\alpha+n+1)$ as a multiplication between Γ and $(\alpha+n+1)$. A system capable of identifying the important components, such as $P_n^{(\alpha,\beta)}(x)$ or $\Gamma(\alpha+n+1)$, is therefore desirable. Hereafter, we define these components as Mathematical Objects of Interest (MOI) [9].

The *importance* of math objects is a somewhat imprecise description and thus difficult to measure. Currently, not much effort has been made in identifying meaningful subexpressions. Kristianto et al. [214] introduced dependency graphs between formulae. With this approach, they were able to build dependency graphs of mathematical expressions, but only if the expressions appeared as single expressions in the context. For example, if $\Gamma(\alpha+n+1)$ appears as a stand-alone expression in the context, the algorithm will declare a dependency with Equation (3.2). However, it is more likely that different forms, such as $\Gamma(s)$, appear in the context. Since this expression does not match any subexpression in Equation (3.2), the approach cannot establish a connection with $\Gamma(s)$. Kohlhase et al. studied in [191, 193, 196] another approach to identify essential components in formulae. They performed eye-tracking studies to identify important areas in rendered mathematical formulae. While this is an interesting approach that allows one to learn more about the insights of human behaviors of reading and understanding math, it is inaccessible for extensive studies.

This section presents the first extensive frequency distribution study of mathematical equations in two large scientific corpora, the e-Print archive arXiv.org (hereafter referred to as arXiv[17]) and the international reviewing service for pure and applied mathematics zbMATH[18]. We will show that math expressions, similar to words in natural language corpora, also obey Zipf's law [297], and therefore follows a *Zipfian* distribution. Related research projects observed a relation to Zipf's law for single math symbols [71, 329]. In the context of quantitative linguistics, Zipf's law states that given a text corpus, the frequency of any word is inversely proportional to its rank in the frequency table. Motivated by the similarity to linguistic properties, we will present a novel approach for ranking formulae by their relevance via a customized version of the ranking function BM25 [310]. We will present results that can be easily embedded in other systems in order to distinguish between common and uncommon notations within formulae. Our results lay a foundation for future research projects in MathIR.

[17] https://arxiv.org/ [accessed 2019-09-01]
[18] https://zbmath.org [accessed 2019-09-01]

Fundamental knowledge on frequency distributions of math formulae is beneficial for numerous applications in MathIR, ranging from educational purposes [341] to math recommendation systems [50], search engines [92, 274], and even automatic plagiarism detection systems [253, 254, 334]. For example, students can search for the conventions to write certain quantities in formulae; document preparation systems can integrate an auto-completion or auto-correction service for math inputs; search or recommendation engines can adjust their ranking scores with respect to standard notations; and plagiarism detection systems can estimate whether two identical formulae indicate potential plagiarism or are just using the conventional notations in a particular subject area. To exemplify the applicability of our findings, we present a textual search approach to retrieve mathematical formulae. Further, we will extend zbMATH's faceted search by providing facets of mathematical formulae according to a given textual search query. Lastly, we present a simple auto-completion system for math inputs as a contribution towards advancing mathematical recommendation systems. Further, we show that the results provide useful insights for plagiarism detection algorithms. We provide access to the source code, the results, and extended versions of all of the figures appearing in this paper at https://github.com/ag-gipp/FormulaCloudData.

3.2.1 Related Work

Today, mathematical search engines index formulae in a database. Much effort has been undertaken to make this process as efficient as possible in terms of precision and runtime performance [92, 181, 231, 236, 407]. The generated databases naturally contain the information required to examine the distributions of the indexed mathematical formulae. Yet, no in-depth studies of these distributions have been undertaken. Instead, math search engines focus on other aspects, such as devising novel similarity measures and improving runtime efficiency. This is because the goal of math search engines is to retrieve relevant (i.e., similar) formulae which correspond to a given search query that partially [211, 231, 274] or exclusively [92, 181, 182] contains formulae. However, for a fundamental study of distributions of mathematical expressions, no similarity measures nor efficient lookup or indexing is required. Thus, we use the general-purpose query language XQuery and employ the BaseX[19] implementation. BaseX is a free open-source XML database engine, which is fully compatible with the latest XQuery standard [140, 396]. Since our implementations rely on XQuery, we are able to switch to any other database which allows for processing via XQuery.

3.2.2 Data Preparation

LaTeX is the de facto standard for the preparation of academic manuscripts in the fields of mathematics and physics [129]. Since LaTeX allows for advanced customizations and even computations, it is challenging to process. For this reason, LaTeX expressions are unsuitable for an extensive distribution analysis of mathematical notations. For mathematical expressions on the web, the XML formatted MathML[20] is the current standard, as specified by the World Wide Web Consortium (W3C). The tree structure and the fixed standard, i.e., MathML tags, cannot be changed, thus making this data format reliable. Several available tools are able to convert from LaTeX to MathML [18] and various databases are able to index XML data. Thus, for this study,

[19] http://basex.org/ [accessed 2019-09-01]; We used BaseX 9.2 for our experiments.
[20] https://www.w3.org/TR/MathML3/ [accessed 2019-09-01]

we have chosen to focus on MathML. In the following, we investigate the databases arXMLiv (08/2018) [132] and zbMATH[21] [333].

The arXMLiv dataset (≈1.2 million documents) contains HTML5 versions of the documents from the e-Print archive arXiv.org. The HTML5 documents were generated from the TEX sources via LATEXML [257]. LATEXML converted all mathematical expressions into MathML with parallel markup, i.e., presentation and content MathML. In this study we only consider the subsets *no-problem* and *warning*, which generated no errors during the conversion process. Nonetheless, the MathML data generated still contains some errors or falsely annotated math. For example, we discovered several instances of affiliation and footnotes, SVG[22] and other unknown tags, encoded in MathML. Regarding the footnotes, we presumed that authors falsely used mathematical environments for generating footnote or affiliation marks. We used the TEX string, provided as an attribute in the MathML data, to filter out expressions that match the string '`{}^{*}`', where '`*`' indicates any possible expression. In addition, we filtered out SVG and other unknown tags. We assume that these expressions were generated by mistake due to limitations of LATEXML. The final arXiv dataset consisted of 841,008 documents which contained at least one mathematical formula. The dataset contained a total of 294,151,288 mathematical expressions.

In addition to arXiv, we investigated zbMATH, an international reviewing service for pure and applied mathematics which contains abstracts and reviews of articles, hereafter uniformly called abstracts, mainly from the domains of pure and applied mathematics. The abstracts in zbMATH are formatted in TEX [333]. To be able to compare arXiv and zbMATH, we manually generated MathML via LATEXML for each mathematical formula in zbMATH and performed the same filters as used for the arXiv documents. The zbMATH dataset contained 2,813,451 abstracts, of which 1,349,297 contained at least one formula. In total, the dataset contained 11,747,860 formulae. Even though the total number of formulae is smaller compared to arXiv, we hypothesize that math formulae in abstracts are particularly meaningful.

3.2.2.1 Data Wrangling

Since we focused on the frequency distributions of visual expressions, we only considered pMML. Rather than normalizing the pMML data, e.g., via MathMLCan [117], which would also change the tree structure and visual core elements in pMML, we only eliminated the attributes. These attributes are used for minor visual changes, e.g., stretched parentheses or inline limits of sums and integrals. Thus, for this first study, we preserved the core structure of the pMML data, which might provide insightful statistics for the MathML community to further cultivate the standard. After extracting all MathML expressions, filtering out falsely annotated math and SVG tags, and eliminating unnecessary attributes and annotations, the datasets required 83GB of disk space for arXiv and 6GB for zbMATH, respectively.

In the following, we indexed the data via BaseX. The indexed datasets required a disk space of 143.9GB in total (140GB for arXiv and 3.9GB for zbMATH). Due to the limitations[23] of databases in BaseX, it was necessary to split our datasets into smaller subsets. We split the datasets

[21] https://zbmath.org/ [accessed 2019-09-01]

[22] Scalable Vector Graphics

[23] A detailed overview of the limitations of BaseX databases can be found at http://docs.basex.org/wiki/Statistics [accessed 2019-09-01].

according to the 20 major article categories of arXiv[24] and classifications of zbMATH. To increase performance, we use BaseX in a server-client environment. We experienced performance issues in BaseX when multiple clients repeatedly requested data from the same server in short intervals. We determined that the best workaround for this issue was to launch BaseX servers for each database, i.e., each category/classification.

Mathematical expressions often consist of multiple meaningful subexpressions, which we defined as MOIs. However, without further investigation of the context, it is impossible to determine meaningful subexpressions. As a consequence, every equation is a potential MOI on its own and potentially consists of multiple other MOIs. For an extensive frequency distributional analysis, we aim to discover all possible mathematical objects. Hence, we split every formula into its components. Since MathML is an XML data format (essentially a tree-structured format), we define subexpressions of equations as subtrees of its MathML format.

$P_n^{(\alpha,\beta)}(x)$

```
1  <math><mrow>
2    <msubsup>
3      <mi>P</mi>
4      <mi>n</mi>
5      <mrow>
6        <mo>(</mo>
7        <mi>α</mi>
8        <mo>,</mo>
9        <mi>β</mi>
10       <mo>)</mo>
11       <mo></mo>
12     </mrow>
13   </msubsup>
14   <mo></mo>
15   <mrow>
16     <mo>(</mo>
17     <mi>x</mi>
18     <mo>)</mo>
19   </mrow>
20 </mrow></math>
```

Listing 3.1: MathML representation of $P_n^{(\alpha,\beta)}(x)$.

Listing 3.1 illustrates a Jacobi polynomial $P_n^{(\alpha,\beta)}(x)$ in pMML. The `<mo>` element on line 14 contains the *invisible times* UTF-8 character. By definition, the `<math>` element is the root element of MathML expressions. Since we cut off all other elements besides pMML nodes, each `<math>` element has one and only one child element[25]. Thus, we define the child element of the `<math>` element as the root of the expression. Starting from this root element, we explore all subexpressions. For this study, we presume that every meaningful mathematical object (i.e., MOI) must contain at least one identifier.

Hence, we only study subtrees which contain at least one `<mi>` node. Identifiers, in the sense of MathML, are '*symbolic names or arbitrary text*' [26], e.g., single Latin or Greek letters. Identifiers do not contain special characters (other than Greek letters) or numbers. As a consequence, arithmetic expressions, such as $(1+2)^2$, or sequences of special characters and numbers, such as $\{1,2,...\} \cap \{-1\}$, will not appear in our distributional analysis. However, if a sequence or arithmetic expression consists of an identifier somewhere in the pMML tree (such as in $\{1,2,...\} \cap A$), the entire expression will be recognized. The Jacobi polynomial $P_n^{(\alpha,\beta)}(x)$, therefore consists of the following subexpressions: $P_n^{(\alpha,\beta)}$, (α,β), (x), and the single identifiers P, n, α, β, and x. The entire expression is also a mathematical object. Hence, we take entire expressions with an identifier into account for our analysis. In the following, the set of subexpressions will be understood to include the expression itself.

For our experiments, we also generated a string representation of the MathML data. The string is generated recursively by applying one of two rules for each node: (i) if the current node is a leaf, the node-tag and the content will be merged by a colon, e.g., `<mi>x</mi>` will be converted

[24] The arXiv categories *astro-ph* (astro physics), *cond-mat* (condensed matter), and *math* (mathematics) were still too large for a single database. Thus, we split those categories into two equally sized parts.

[25] Sequences are always nested in an `<mrow>` element.

[26] https://www.w3.org/TR/MathML3/chapter3.html [accessed 2019-09-01]

to `mi:x`; (ii) otherwise the node-tag wraps parentheses around its content and separates the children by a comma, e.g.,

```
<mrow><mo>(</mo><mi>x</mi><mo>)</mo></mrow>
```
(3.3)

will be converted to

```
mrow(mo:(,mi:x,mo:)).
```
(3.4)

Furthermore, the special UTF-8 characters for invisible times (U+2062) and function application (U+2061) are replaced by `ivt` and `fa`, respectively. For example, the gamma function with argument $x+1$, $\Gamma(x+1)$ would be represented by

```
mrow(mi:Γ,mo:ivt,mrow(mo:(,mrow(mi:x,mo:+,mn:1),mo:))).
```
(3.5)

Between Γ and $(x+1)$, there would most likely be the special character for *invisible times* rather than for *function application*, because LaTeXML is not able to parse Γ as a function. Note that this string conversion is a bijective mapping. The string representation reduces the verbose XML format to a more concise presentation. Thus, an equivalence check between two expressions is more efficient.

3.2.2.2 Complexity of Math

Mathematical expressions can become complex and lengthy. The tree structure of MathML allows us to introduce a measure that reflects the complexity of mathematical expressions. More complex expressions usually consist of more extensively nested subtrees in the MathML data. Thus, we define the complexity of a mathematical expression by the maximum depth of the MathML tree. In XML the content of a node and its attributes are commonly interpreted as children of the node. Thus, we define the depth of a single node as 1 rather than 0, i.e., single identifiers, such as `<mi>P</mi>`, have a complexity of 1. The Jacobi polynomial from Listing 3.1 has a complexity of 4.

We perform the extraction of subexpressions from MathML in BaseX. The algorithm for the extraction process is written in XQuery. The algorithm traverses recursively downwards from the root to the leaves. In each iteration, it checks whether there is an identifier, i.e., `<mi>` element, among the descendants of the current node. If there is no such element, the subtree will be ignored. It seems counterintuitive to start from the root and check if an identifier is among the descendants rather than starting at each identifier and traversing upwards to the root. If an XQuery requests a node in BaseX, BaseX loads the entire subtree of the requested node into the cache (up to a specified size). If the algorithm traverses upwards through the MathML tree, the XQuery will trigger database requests in every iteration. Hence, the downwards implementation performs better, since there is only one database request for every expression rather than for every subexpression.

Since we only minimize the pMML data rather than normalizing it, two identically rendered expressions may have different complexities. For instance,

```
<mrow><mi>x</mi></mrow>
```
(3.6)

consists of two distinct subexpressions, but both of them are displayed the same. Another problem often appears for arrays or similar visually complicated structures. The extracted expressions are not necessarily logical subexpressions. We will consider applying more advanced embedding techniques such as special tokenizers [231], symbol layout trees [92, 407], and a MathML normalization via MathMLCan [117] in future research to overcome these issues.

3.2.3 Frequency Distributions of Mathematical Formulae

By splitting each formula into subexpressions, we generated longer documents and a bias towards low complexities. Note that, hereafter, we only refer to the mathematical content of documents. Thus, the length of a document refers to the number of math formulae - here the number of subexpressions - in the document. After splitting expressions into subexpressions, arXiv consists of 2.5B and zbMATH of 61M expressions, which raised the average document length to 2,982.87 for arXiv and 45.47 for zbMATH, respectively.

For calculating frequency distributions, we merged two subexpressions if their string representations were identical. Remember, the string representation is unique for each MathML tree. After merging, arXiv consisted of 350,206,974 unique mathematical subexpressions with a maximum complexity of 218 and an average complexity of 5.01. For high complexities over 70, the formulae show some erroneous structures that might be generated from LaTeXML by mistake. For example, the expression with the highest complexity is a long sequence of a polynomial starting with '$P_4(t_1, t_3, t_7, t_{11}) =$' followed by 690 summands. The complexity is caused by a high number of unnecessarily deeply nested `<mrow>` nodes. The highest complexity with a minimum document frequency of two is 39, which is a continued fraction. Since continued fractions are nested fractions, they naturally have a large complexity. One of the most complex expressions (complexity 20) with a minimum document frequency of three was the formula

$$\left(\sum_{j_1=1}^{n}\left(\sum_{j_2=1}^{n}\left(\cdots\left(\sum_{j_m=1}^{n}\left|T\left(e_{j_1},\ldots,e_{j_m}\right)\right|^{q_m}\right)^{\frac{q_{m-1}}{q_m}}\cdots\right)^{\frac{q_2}{q_3}}\right)^{\frac{q_1}{q_2}}\right)^{\frac{1}{q_1}} \leq C^{\mathbb{K}}_{m,p,\mathbf{q}}\,\|T\|\,. \quad (3.7)$$

In contrast, zbMATH only consisted of 8,450,496 unique expressions with a maximum complexity of 26 and an average complexity of 3.89. One of the most complex expressions in zbMATH with a minimum document frequency of three was

$$M_p(r,f) = \left(\frac{1}{2\pi}\int_0^{2\pi}\left|f\left(re^{i\theta}\right)\right|^p d\theta\right)^{1/p}. \quad (3.8)$$

As we expected, reviews and abstracts in zbMATH were generally shorter and consisted of less complex mathematical formulae. The dataset also appeared to contain fewer erroneous expressions, since expressions of complexity 25 are still readable and meaningful.

Figure 3.2 shows the ratio of unique subexpressions for each complexity in both datasets. The figure illustrates that both datasets share a peak at complexity four. Compared to zbMATH, the arXiv expressions are slightly more evenly distributed over the different levels of complexities. Interestingly, complexities one and two are not dominant in either of the two datasets. Single identifiers only make up 0.03% in arXiv and 0.12% in zbMATH, which is comparable to expressions of complexity 19 and 14, respectively. This finding illustrates the problem of capturing semantic meanings for single identifiers rather than for more complex expressions [330]. It also substantiates that entire expressions, if too complex, are not suitable either for capturing the semantic meanings [214]. Instead, a middle ground is desirable, since the most unique expressions in both datasets have a complexity between 3 and 5. Table 3.4 summarizes the statistics of the examined datasets.

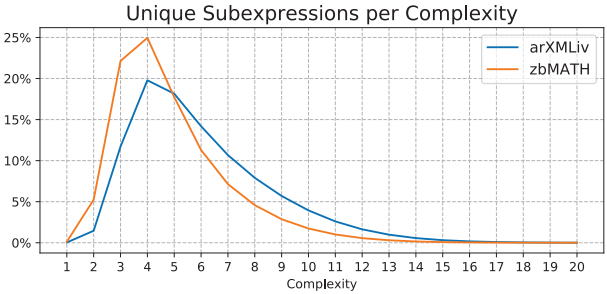


Figure 3.2: Unique subexpressions for each complexity in arXiv and zbMATH.

Table 3.4: Dataset overview. Average Document Length is defined as the average number of subexpressions per document.

Category	arXiv	zbMATH
Documents	841,008	1,349,297
Formulae	294,151,288	11,747,860
Subexpressions	2,508,620,512	61,355,307
Unique Subexpressions	350,206,974	8,450,496
Average Document Length	2,982.87	45.47
Average Complexity	5.01	3.89
Maximum Complexity	218	26

3.2.3.1 Zipf's Law

In linguistics, it is well known that word distributions follow Zipf's Law [297], i.e., the r-th most frequent word has a frequency that scales to

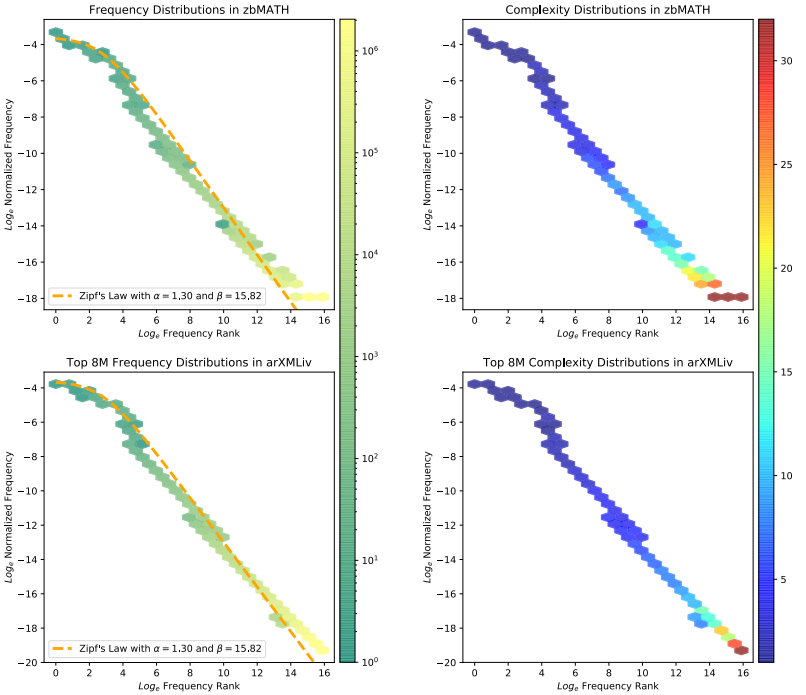
$$f(r) \propto \frac{1}{r^{\alpha}} \tag{3.9}$$

with $\alpha \approx 1$. A better approximation can be applied by a shifted distribution

$$f(r) \propto \frac{1}{(r+\beta)^{\alpha}}, \tag{3.10}$$

where $\alpha \approx 1$ and $\beta \approx 2.7$. In a study on Zipf's law, Piantadosi [297] illustrated that not only words in natural language corpora follow this law surprisingly accurately, but also many other human-created sets. For instance, in programming languages, in biological systems, and even in music. Since mathematical communication has derived as the result of centuries of research, it would not be surprising if mathematical notations would also follow Zipf's law. The primary conclusion of the law illustrates that there are some very common tokens against a large number of symbols which are not used frequently. Based on this assumption, we can postulate that a score based on frequencies might be able to measure the peculiarity of a token. The infamous TF-IDF ranking functions and their derivatives [23, 310] have performed well in linguistics for

many years and are still widely used in retrieval systems [30]. However, since we split every expression into its subexpressions, we generated an anomalous bias towards shorter, i.e., less complex, formulae. Hence, distributions of subexpressions may not obey Zipf's law.



(a) Frequency Distributions

(b) Complexity Distributions

Figure 3.3: Each figure illustrates the relationship between the frequency ranks (x-axis) and the normalized frequency (y-axis) in zbMATH (top) and arXiv (bottom). For arXiv, only the first 8 million entries are plotted to be comparable with zbMATH ($\approx$ 8.5 million entries). Subfigure (a) shades the hexagonal bins from green to yellow using a logarithmic scale according to the number of math expressions that fall into a bin. The dashed orange line represents Zipf's distribution (3.10). The values for α and β are provided in the plots. Subfigure (b) shades the bins from blue to red according to the maximum complexity in each bin.

Figure 3.3 visualizes a comparison between Zipf's law and the frequency distributions of mathematical subexpressions in arXiv and zbMATH. The dashed orange line visualizes the power law (3.10). The plots demonstrate that the distributions in both datasets obey this power law. Interestingly, there is not much difference in the distributions between both datasets. Both distributions seem to follow the same power law, with $\alpha = 1.3$ and $\beta = 15.82$. Moreover, we can observe that the developed complexity measure seems to be appropriate, since the complexity distributions for formulae are similar to the distributions for the length of words [297]. In other

words, more complex formulae, as well as long words in natural languages, are generally more specialized and thus appear less frequent throughout the corpus. Note that colors of the bins for complexities fluctuate for rare expressions because the color represents the maximum rather than the average complexity in each bin.

3.2.3.2 Analyzing and Comparing Frequencies

Figure 3.4 shows in detail the most frequently used mathematical expressions in arXiv for the complexities 1 to 7. The orange dashed line visible in all graphs represents the normal Zipf's law distribution from Equation (3.9). We explore the total frequency values without any normalization. Thus, Equation (3.9) was multiplied by the highest frequency for each complexity level to fit the distribution. The plots in Figure 3.4 demonstrate that even though the parameter α varies between 0.35 and 0.62, the distributions in each complexity class also obey Zipf's law.

The plots for each complexity class contain some interesting fluctuations. We can spot a set of five single identifiers that are most frequently used throughout arXiv: n, i, x, t, and k. Even though the distributions follow Zipf's law accurately, we can explore that these five identifiers are proportionally more frequently used than other identifiers and clearly separate themselves above the rest (notice the large gap from k to a). All of the five identifiers are known to be used in a large variety of scenarios. Surprisingly, one might expect that common pairs of identifiers would share comparable frequencies in the plots. However, typical pairs, such as x and y, or α and β, possess a large discrepancy.

The plot of complexity two also reveals that two expressions are proportionally more often used than others: (x) and (t). These two expressions appear more than three times as often in the corpus than any other expression of the same complexity. On the other hand, the quantitative difference between (x) and (t) is negligible. We may assume that arXiv's primary domain, physics, causes the quantitative disparity between (x), (t), and the other tokens. The primary domain of the dataset becomes more clearly visible for higher complexities, such as $SU(2)$ (C3[27]) or kms^{-1} (C4).

Another surprising property of arXiv is that symmetry groups, such as $SU(2)$, appear to play an essential role in the majority of articles on arXiv, see $SU(2)$ (C3), $SU(2)_L$ (C4), and $SU(2) \times SU(2)$ (C5), among others. The plots of higher complexities[28], made this even more noticeable. Given a complexity of six, for example, the most frequently used expression was $SU(2)_L \times SU(2)_R$, and for a complexity of seven it was $SU(3) \times SU(2) \times U(1)$. Given a complexity of eight, ten out of the top-12 expressions were from symmetry group calculations.

It is also worthwhile to compare expressions among different levels of complexities. For instance, (x) and (t) appeared almost six million times in the corpus, but $f(x)$ (at position three in C3) was the only expression which contained one of these most common expressions. Note that subexpressions of variations, such as (x_0), (t_0), or $(t - t')$, do not match the expression of complexity two. This may imply that (x), and especially (t), appear in many different scenarios. Further, we can examine that even though (x) is a part of $f(x)$ in only approximately 3% of all cases, it is still the most likely combination. These results are especially useful for recommendation systems that make use of math as input. Moreover, plagiarism detection

[27] We refer to a given complexity n with Cn, i.e., C3 refers to complexity 3.

[28] More plots showing higher complexities are available at https://github.com/ag-gipp/FormulaCloudData [accessed 2021-10-01]

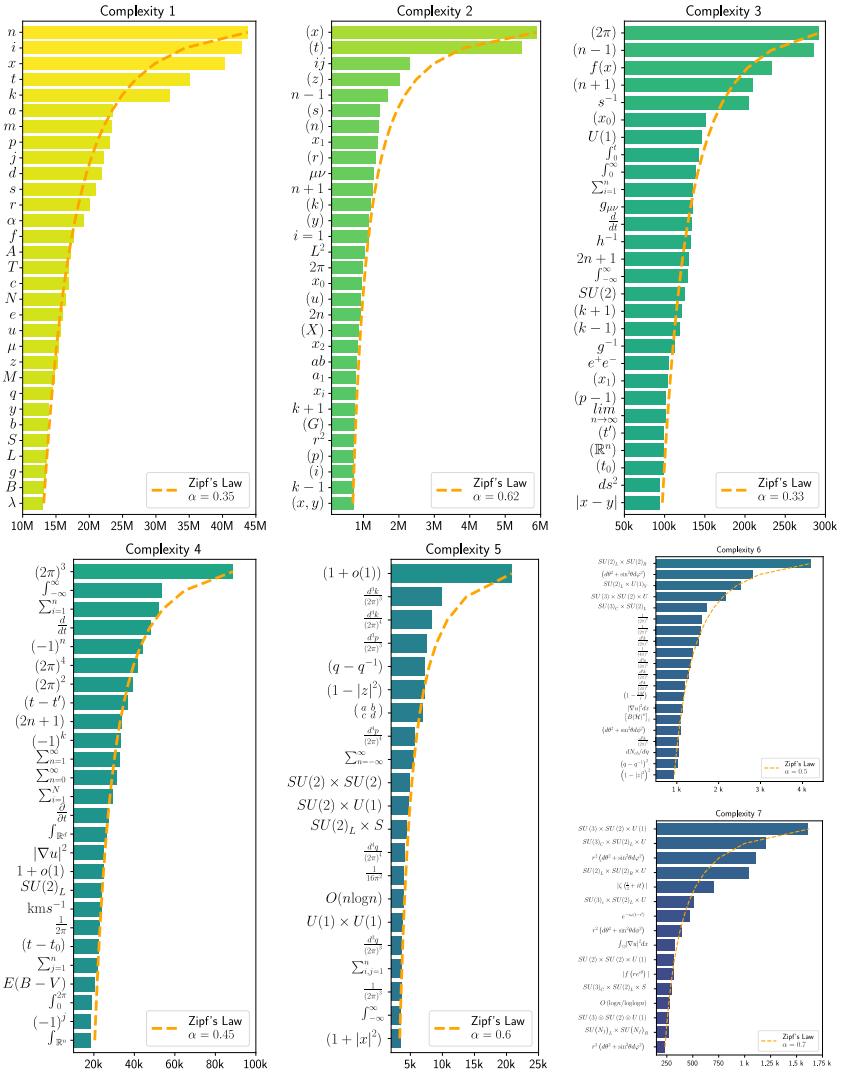


Figure 3.4: Overview of the most frequent mathematical expressions in arXiv for complexities 1-7. The color gradient from yellow to blue represents the frequency in the dataset. Zipf's law (3.9) is represented by a dashed orange line.

systems may also benefit from such a knowledge base. For instance, it might be evident that $f(x)$ is a very common expression, but for automatic systems that work on a large scale, it is not clear whether duplicate occurrences of $f(x)$ or $\Xi(x)$ should be scored differently, e.g., in the case of plagiarism detection.

Figure 3.4 shows only the most frequently occurring expressions in arXiv. Since we already explored a bias towards physics formulae in arXiv, it is worth comparing the expressions present within both datasets. Figure 3.5 compares the 25-top expressions for the complexities one to six. In zbMATH, we discovered that computer science and graph theory appeared as popular topics, see for example $G = (V, E)$ (in C3 at position 20) and the Bachmann-Landau notations in $O(\log n)$, $O(n^2)$, and $O(n^3)$ (C4 positions 2, 3, and 19).

From Figure 3.5, we can also deduce useful information for MathIR tasks which focus on semantic information. Current semantic extraction tools [330] or LaTeX parsers [18] still have difficulties distinguishing *multiplications* from *function calls*. For example as mentioned before, LaTeXML [257] adds an *invisible times* character between $f(x)$ rather than a *function application*. Investigating the most frequently used terms in zbMATH in Table 3.5 reveals that u is most likely considered to be a function in the dataset: $u(t)$ (rank 8), $u(x)$ (rank 13), u_{xx} (rank 16), $u(0)$ (rank 17), $|\nabla u|$ (rank 22). Manual investigations of extended lists reveal even more hits: $u_0(x)$ (rank 30), $-\Delta u$ (rank 32), and $u(x, t)$ (rank 33). Since all eight terms are among the most frequent 35 entries in zbMATH, it implies that u can most likely be considered to imply a function in zbMATH. Of course, this does not imply that u must always be a function in zbMATH (see $f(u)$ on rank 14 in C3), but this allows us to exploit probabilities for improving MathIR performance. For instance, if not stated otherwise, u could be interpreted as a function by default, which could help increase the precision of the aforementioned tools.

Figure 3.5 also demonstrates that our two datasets diverge for increasing complexities. Hence, we can assume that frequencies of less complex formulae are more topic-independent. Conversely, the more complex a math formula is, the more context-specific it is. In the following, we will further investigate this assumption by applying TF-IDF rankings on the distributions.

3.2.4 Relevance Ranking for Formulae

Zipf's law encourages the idea of scoring the relevance of words according to their number of occurrences in the corpus and in the documents. The family of BM25 ranking functions based on TF-IDF scores are still widely used in several retrieval systems [30, 310]. Since we demonstrated that mathematical formulae (and their subexpressions) obey Zipf's law in large scientific corpora, it appears intuitive to also use TF-IDF rankings, such as a variant of BM25, to calculate their relevance.

Okapi BM25

In its original form [310], *Okapi BM25* was calculated as follows

$$\mathrm{bm25}(t,d) := \frac{(k+1)\,\mathrm{IDF}(t)\,\mathrm{TF}(t,d)}{\mathrm{TF}(t,d) + k\left(1 - b + \frac{b|d|}{\mathrm{AVG}_{\mathrm{DL}}}\right)}. \tag{3.11}$$

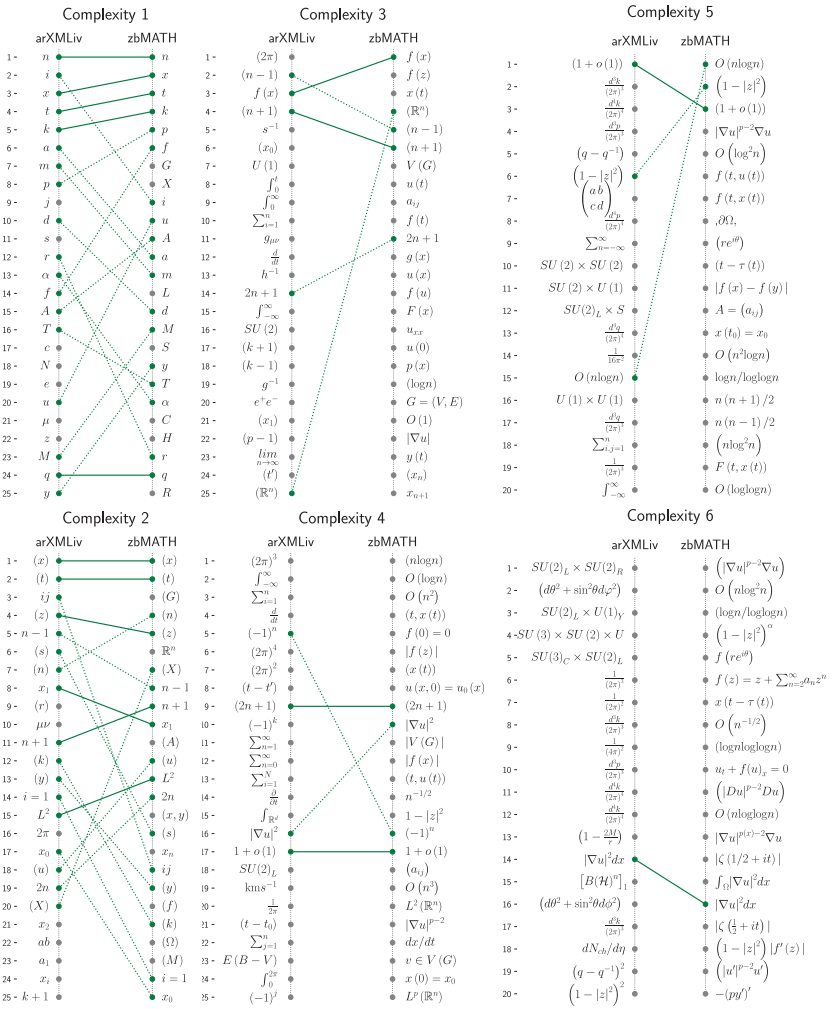


Figure 3.5: The top-20 and 25 most frequent expressions in arXiv (left) and zbMATH (right) for complexities 1-6. A line between both sets indicates a matching set. Bold lines indicate that the matches share a similar rank (distance of 0 or 1).

Here, $\mathrm{TF}\,(t, d)$ is the term frequency of t in the document d, $|d|$ the length of the document d (in our case, the number of subexpressions), $\mathrm{AVG}_{\mathrm{DL}}$ the average length of the documents in the corpus (see Table 3.4), and $\mathrm{IDF}\,(t)$ is the inverse document frequency of t, defined as

$$\mathrm{IDF}(t) := \log \frac{N - n(t) + \frac{1}{2}}{n(t) + \frac{1}{2}}, \tag{3.12}$$

where N is the number of documents in the corpus and $n(t)$ the number of documents which contain the term t. By adding $\frac{1}{2}$, we avoid $\log 0$ and division by 0. The parameters k and b are free, with b controlling the influence of the normalized document length and k controlling the influence of the term frequency on the final score. For our experiments, we chose the standard value $k = 1.2$ and a high impact factor of the normalized document length via $b = 0.95$.

As a result of our subexpression extraction algorithm, we generated a bias towards low complexities. Moreover, longer documents generally consist of more complex expressions. As demonstrated in Section 3.2.2.1, a document that only consists of the single expression $P_n^{(\alpha,\beta)}(x)$, i.e., the document had a length of one, would generate eight subexpressions, i.e., it results in a document length of eight. Thus, we modify the BM25 score in Equation (3.11) to emphasize higher complexities and longer documents. First, the average document length is divided by the average complexity AVG_C in the corpus that is used (see Table 3.4), and we calculate the reciprocal of the document length normalization to emphasize longer documents.

Moreover, in the scope of a single document, we want to emphasize expressions that do not appear frequently in this document, but are the most frequent among their level of complexity. Thus, less complex expressions are ranked more highly if the document overall is not very complex. To achieve this weighting, we normalize the term frequency of an expression t according to its complexity $c(t)$ and introduce an inverse term frequency according to all expressions in the document. We define the inverse term frequency as

$$\mathrm{ITF}(t, d) := \log \frac{|d| - \mathrm{TF}(t, d) + \frac{1}{2}}{\mathrm{TF}(t, d) + \frac{1}{2}}. \tag{3.13}$$

Definition of the importance score of a formula in a document

Finally, we define the score $\mathrm{s}(t, d)$ of a term t in a document d as

$$\mathrm{s}(t, d) := \frac{(k+1)\,\mathrm{IDF}(t)\,\mathrm{ITF}(t, d)\,\mathrm{TF}(t, d)}{\max\limits_{t' \in d|_{c(t)}} \mathrm{TF}(t', d) + k\left(1 - b + \frac{b\,\mathrm{AVG}_{\mathrm{DL}}}{|d|\,\mathrm{AVG}_C}\right)}. \tag{3.14}$$

The TF-IDF ranking functions and the introduced $\mathrm{s}(t, d)$ are used to retrieve relevant documents for a given search query. However, we want to retrieve relevant subexpressions over a set of documents.

Definition of the Mathematical BM25

Thus, we define the score of a formula (mBM25) over a set of documents as the maximum score over all documents

$$\mathrm{mBM25}(t,d) := \max_{d\in D} s(t,d), \tag{3.15}$$

where D is a set of documents.

We used *Apache Flink* [157] to count the expressions and process the calculations. Thus, our implemented system scales well for large corpora.

Table 3.6 shows the top-7 scored expressions, where D is the entire zbMATH dataset. The retrieved expressions can be considered as meaningful and real-world examples of MOIs, since most expressions are known for specific mathematical concepts, such as $\mathrm{Gal}(\overline{\mathbb{Q}}/\mathbb{Q})$, which refers to the Galois group of $\overline{\mathbb{Q}}$ over $\mathbb{Q}$, or $L^2(\mathbb{R}^2)$, which refers to the L^2-space (also known as *Lebesgue space*) over $\mathbb{R}^2$. However, a more topic-specific retrieval algorithm is desirable. To achieve this goal, we (i) retrieved a topic-specific subset of documents $D_q \subset D$ for a given textual search query q, and (ii) calculated the scores of all expressions in the retrieved documents. To generate D_q, we indexed the text sources of the documents from arXiv and zbMATH via Elasticsearch (ES)[29] and performed the pre-processing steps: filtering stop words, stemming, and ASCII-folding[30]. Table 3.5 summarizes the settings we used to retrieve MOIs from a topic-specific subset of documents D_q. We also set a minimum hit frequency according to the number of retrieved documents an expression appears in. This requirement filters out uncommon notations.

Table 3.5: Settings for the retrieval experiments.

	arXiv	zbMATH
Retrieved Doc.	40	200
Min. Hit Freq.	7	7
Min. DF	50	10
Max. DF	10k	10k

Figure 3.6 shows the results for five search queries. We asked a domain expert from the NIST to annotate the results as related (shown as green dots in Figure 3.6) or non-related (red dots). We found that the results range from good performances (e.g., for the Riemann zeta function) to bad performances (e.g., beta function). For instance, the results for the Riemann zeta function are surprisingly accurate, since we could discover that parts of Riemann's hypothesis[31] were ranked highly throughout the results (e.g., $\zeta(\frac{1}{2}+it)$). On the other hand, for the beta function, we retrieved only a few related hits, of which only one had a strong connection to the beta function $B(x,y)$. We observed that the results were quite sensitive to the chosen settings (see Table 3.5). For instance, according to the beta function, the minimum hit frequency has a strong effect on the results, since many expressions are shared among multiple documents. For arXiv, the expressions $B(\alpha,\beta)$ and $B(x,y)$ only appear in one document of the retrieved 40. However, decreasing the minimum hit frequency would increase noise in the results.

[29] https://github.com/elastic/elasticsearch [accessed 2019-09-01]. We used version 7.0.0

[30] This means that non-ASCII characters are replaced by their ASCII counterparts or will be ignored if no such counterpart exists.

[31] Riemann proposed that the real part of every non-trivial zero of the Riemann zeta function is $1/2$. If this hypothesis is correct, all the non-trivial zeros lie on the critical line consisting of the complex numbers $1/2+it$.

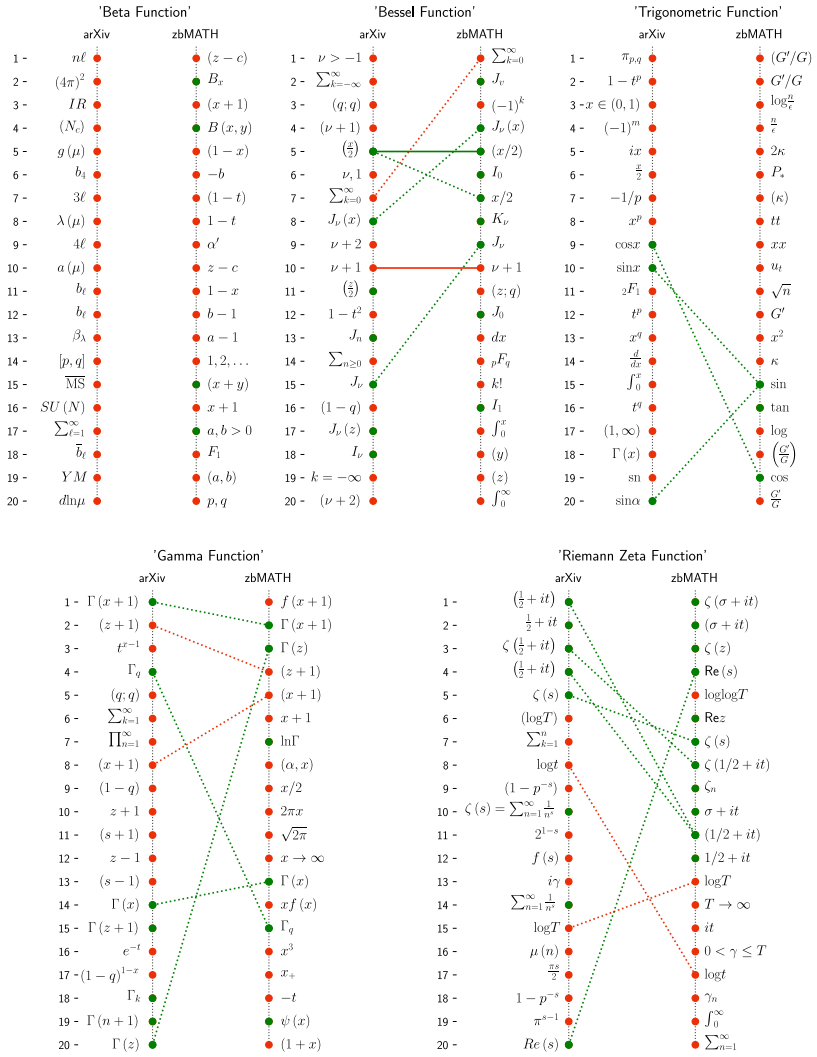


Figure 3.6: Top-20 ranked expressions retrieved from a topic-specific subset of documents D_q. The search query q is given above the plots. Retrieved formulae are annotated by a domain expert with green dots for relevant and red dots for non-relevant hits. A line is drawn if a hit appears in both result sets. The line is colored in green when the hit was marked as relevant.

Table 3.6: Top $\mathrm{s}(t, D)$ scores, where D is the set of all zbMATH documents with a minimum document frequency of 200, maximum document frequency of 500k, and a minimum complexity of 3.

C3		C4		C5	
114.84	$(n!)$	129.44	$i, j = 1, \ldots, n$	119.21	$\mathrm{Gal}\left(\overline{\mathbb{Q}}/\mathbb{Q}\right)$
108.85	ϕ^{-1}	108.52	x_{ij}	112.55	$\lvert f(z)\rvert^p$
100.19	z^{n-1}	108.50	$\dot{x} = A(t)x$	110.52	$\left(1 + \lvert x\rvert^2\right)$
100.06	(c_n)	106.66	$\lvert x - x_0\rvert$	109.19	$\lvert f(x)\rvert^p$
100.05	$B(G)$	105.52	S^{2n+1}	106.22	$\lvert\nabla u\rvert^2 dx$
99.87	$\log_2 n$	104.91	$L^2\left(\mathbb{R}^2\right)$	102.86	$n(n-1)/2$
99.65	$\xi\,(x)$	103.70	$\dot{x} = Ax + Bu$	101.40	$O(n^{-1})$

C6		C7	
110.83	$(1 + \lvert z\rvert^2)^\alpha$	98.72	$\mathrm{div}\left(\lvert\nabla u\rvert^{p-2}\nabla u\right)$
105.69	$f\left(re^{i\theta}\right)$		–
94.14	$f(z) = z + \sum_{n=2}^{\infty} a_n z^n$		–
92.33	$\left(\lvert\nabla u\rvert^{p-2}\nabla u\right)$		–
87.27	$(\log n/\log\log n)$		–
78.54	$O\left(n\log^2 n\right)$		–
	–		–

Even though we asked a domain expert to annotate the results as relevant or not, there is still plenty of room for discussion. For instance, $(x + y)$ (rank 15 in zbMATH, 'Beta Function') is the argument of the gamma function $\Gamma(x + y)$ that appears in the definition of the beta function [98, (5.12.1)] $B(x, y) := \Gamma(x)\Gamma(y)/\Gamma(x + y)$. However, this relation is weak at best, and thus might be considered as not related. Other examples are $\mathrm{Re}z$ and $\mathrm{Re}(s)$, which play a crucial role in the scenario of the Riemann hypothesis (all non-trivial zeroes have $\mathrm{Re}(s) = \frac{1}{2}$). Again, this connection is not obvious, and these expressions are often used in multiple scenarios. Thus, the domain expert did not mark the expressions as being related.

Considering the differences in the documents, it is promising to have observed a relatively high number of shared hits in the results. Further, we were able to retrieve some surprisingly good insights from the results, such as extracting the full definition of the Riemann zeta function [98, (25.2.1)] $\zeta(s) := \sum_{n=1}^{\infty} \frac{1}{n^s}$. Even though a high number of shared hits seem to substantiate the reliability of the system, there were several aspects that affected the outcome negatively, from the exact definition of the search queries to retrieve documents via ES, to the number of retrieved documents, the minimum hit frequency, and the parameters in mBM25.

3.2.5 Applications

The presented results are beneficial for a variety of use-cases. In the following, we will demonstrate and discuss several of the applications that we propose.

Extension of zbMATH's Search Engine Formula search engines are often counterintuitive when compared to textual search, since the user must know how the system operates to enter a search query properly (e.g., does the system supports LaTeX inputs?). Additionally, mathematical concepts can be difficult to capture using only mathematical expressions. Consider, for example, someone who wants to search for mathematical expressions that are related to eigenvalues. A textual search query would only retrieve entire documents that require further investigation to find related expressions. A mathematical search engine, on the other hand, is impractical since it is not clear what would be a fitting search query (e.g., $Av = \lambda v$?). Moreover, formula and textual search systems for scientific corpora are separated from each other. Thus, a textual search engine capable of retrieving mathematical formulae can be beneficial. Also, many search engines allow for narrowing down relevant hits by suggesting filters based on the retrieved results. This technique is known as faceted search. The zbMATH search engine also provides faceted search, e.g., by authors, or year. Adding facets for mathematical expressions allows users to narrow down the results more precisely to arrive at specific documents.

Our proposed system for extracting relevant expressions from scientific corpora via mBM25 scores can be used to search for formulae even with textual search queries, and to add more filters for faceted search implementations. Table 3.7 shows two examples of such an extension for zbMATH's search engine. Searching for 'Riemann Zeta Function' and 'Eigenvalue' retrieved 4,739 and 25,248 documents from zbMATH, respectively. Table 3.7 shows the most frequently used mathematical expressions in the set of retrieved documents. It also shows the reordered formulae according to a default TF-IDF score (with normalized term frequencies) and our proposed mBM25 score. The results can be used to add filters for faceted search, e.g., show only the documents which contain $u \in W_0^{1,p}(\Omega)$. Additionally, the search system now provides more intuitive textual inputs even for retrieving mathematical formulae. The retrieved formulae are also interesting by themselves, since they provide insightful information on the retrieved publications. As already explored with our custom document search system in Figure 3.6, the Riemann hypothesis is also prominent in these retrieved documents.

The differences between TF-IDF and mBM25 ranking illustrates the problem of an extensive evaluation of our system. From a broader perspective, the hit $Ax = \lambda Bx$ is highly correlated with the input query 'Eigenvalue'. On the other hand, the raw frequencies revealed a prominent role of $\operatorname{div}(|\nabla u|^{p-2}\nabla u)$. Therefore, the top results of the mBM25 ranking can also be considered as relevant.

Math Notation Analysis A faceted search system allows us to analyze mathematical notations in more detail. For instance, we can retrieve documents from a specific time period. This allows one to study the evolution of mathematical notation over time [54], or for identifying trends in specific fields. Also, we can analyze standard notations for specific authors since it is often assumed that authors prefer a specific notation style which may vary from the standard notation in a field.

Table 3.7: The top-5 frequent mathematical expressions in the result set of zbMATH for the search queries 'Riemann Zeta Function' (top) and 'Eigenvalue' (bottom) grouped by their complexities (left) and the hits reordered according to their relevance scores (right). The TF-IDF score was calculated with normalized term frequencies.

Riemann Zeta Function

C1		C2		C3		C4	
15,051	n	4,663	(s)	1,456	$\zeta(s)$	349	$(\frac{1}{2}+it)$
11,709	s	2,460	(x)	340	$\sigma+it$	232	$(1/2+it)$
9,768	x	2,163	(n)	310	$\sum_{n=1}^{\infty}$	195	$(\sigma+it)$
8,913	k	1,485	(t)	275	$(\log T)$	136	$\frac{1}{2}+it$
8,634	T	1,415	it	264	$1/2+it$	97	$s=\sigma+it$

C5		C6		TF-IDF	mBM25
203	$\zeta(\frac{1}{2}+it)$	105	$\lvert\zeta(1/2+it)\rvert$	$\zeta(s)$	$\zeta\,(1/2+it)$
166	$\zeta(1/2+it)$	88	$\left\lvert\zeta(\frac{1}{2}+it)\right\rvert$	$\zeta(1/2+it)$	$(1/2+it)$
124	$\zeta(\sigma+it)$	81	$\lvert\zeta(\sigma+it)\rvert$	$(1/2+it)$	$(\frac{1}{2}+it)$
54	$\zeta(1+it)$	32	$\lvert\zeta(1+it)\rvert$	$\frac{1}{2}+it$	$\zeta\,(\frac{1}{2}+it)$
44	$\zeta(2n+1)$	22	$\lvert\zeta(+it)\rvert$	$(\frac{1}{2}+it)$	$(\sigma+it)$

Eigenvalue

C1		C2		C3		C4	
45,488	n	12,515	(x)	686	$-\Delta u$	218	$\lvert\nabla u\rvert^{p-2}$
43,090	x	6,598	(t)	555	$(n-1)$	218	$-\Delta_p u$
37,434	λ	4,377	λ_1	521	$\lvert\nabla u\rvert$	133	$W_0^{1,p}(\Omega)$
35,302	u	2,787	(Ω)	512	a_{ij}	127	$\lvert\nabla u\rvert^2$
22,460	t	2,725	$\mathbb{R}^n$	495	$u(x)$	97	(a_{ij})

C5		C6		TF-IDF	mBM25
139	$\lvert\nabla u\rvert^{p-2}\nabla u$	137	$\left(\lvert\nabla u\rvert^{p-2}\nabla u\right)$	$Ax=\lambda Bx$	$-\operatorname{div}\left(\lvert\nabla u\rvert^{p-2}\nabla u\right)$
68	$-d^2/dx^2$	35	$-(py')'$	$-\Delta p$	$\operatorname{div}\left(\lvert\nabla u\rvert^{p-2}\nabla u\right)$
51	$A=(a_{ij})$	26	$(\lvert u'\rvert^{p-2}u')$	$P(\lambda)$	$p=\frac{N+2}{N-2}$
46	$-\frac{d^2}{dx^2}$	18	$(\phi_p(u'))'$	λ_{k+1}	$\left(\phi_p\left(u'\right)\right)'$
45	$u\in W_0^{1,p}(\Omega)$	18	$\int_\Omega \lvert\nabla u\rvert^2\,dx$	$\lambda_1>0$	$\lambda\in(0,\lambda^*)$

Table 3.8: Suggestions to complete '$E = m$' and '$E = \{m, c\}$' (the right-hand side contains m and c) with term and document frequency based on the distributions of formulae in arXiv.

Auto-completion for '$E = m$'			Suggestions for '$E = \{m, c\}$'		
Sug. Expression	TF	DF	Sug. Expression	TF	DF
$E = mc^2$	558	376	$E = mc^2$	558	376
$E = m\cosh\theta$	23	23	$E = \gamma mc^2$	39	38
$E = mv_0$	7	7	$E = \gamma m_e c^2$	41	36
$E = m/\sqrt{1-\dot{q}^2}$	12	6	$E = m\cosh\theta$	23	23
$E = m/\sqrt{1-\beta^2}$	10	6	$E = -mc^2$	35	17
$E = mc^2\gamma$	6	6	$E = \sqrt{m^2c^4 + p^2c^2}$	10	8

Math Recommendation Systems The frequency distributions of formulae can be used to realize effective math recommendation tasks, such as type hinting or error-corrections. These approaches require long training on large datasets, but may still generate meaningless results, such as $G_i = \{(x, y) \in \mathbb{R}^n : x_i = x_i\}$ [400]. We propose a simpler system which takes advantage of our frequency distributions. We retrieve entries from our result database, which contain all unique expressions and their frequencies. We implemented a simple prototype that retrieves the entries via pattern matching. Table 3.8 shows two examples. The left side of the table shows suggested autocompleted expressions for the query '$E = m$'. The right side shows suggestions for '$E =$', where the right-hand side of the equation should contain m and c in any order. A combination using more advanced retrieval techniques, such as similarity measures based on symbol layout trees [92, 407], would enlarge the number of suggestions. This kind of autocomplete and error-correction type-hinting system would be beneficial for various use-cases, e.g., in educational software or for search engines as a pre-processing step of the input.

Plagiarism Detection Systems As previously mentioned, plagiarism detection systems would benefit from a system capable of distinguishing conventional from uncommon notations [253, 254, 334]. The approaches described by Meuschke et al. [254] outperform existing approaches by considering frequency distributions of single identifiers (expressions of complexity one). Considering that single identifiers make up only 0.03% of all unique expressions in arXiv, we presume that better performance can be achieved by considering more complex expressions. The conferred string representation also provides a simple format to embed complex expressions in existing learning algorithms.

Expressions with high complexities that are shared among multiple documents may provide further hints to investigate potential plagiarisms. For instance, the most complex expression that was shared among three documents in arXiv was Equation (3.7). A complex expression being identical in multiple documents could indicate a higher likelihood of plagiarism. Further investigation revealed that similar expressions, e.g., with infinite sums, are frequently used among a larger set of documents. Thus, the expression seems to be a part of a standard notation that is commonly shared, rather than a good candidate for plagiarism detection. Resulting from manual investigations, we could identify the equation as part of a concept called *generalized Hardy-Littlewood inequality* and Equation (3.7) appears in the three documents [24, 292, 304]. All

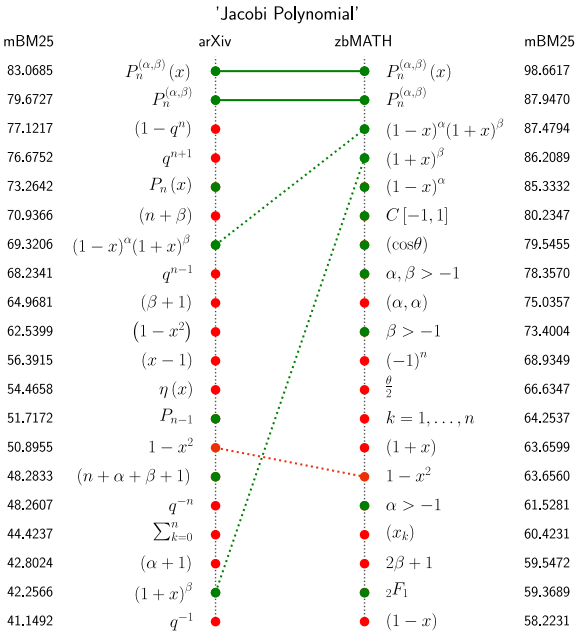


Figure 3.7: The top ranked expression for '*Jacobi polynomial*' in arXiv and zbMATH. For arXiv, 30 documents were retrieved with a minimum hit frequency of 7.

three documents shared one author in common. Thus, this case also demonstrates a correlation between complex mathematical notations and authorship.

Semantic Taggers and Extraction Systems We previously mentioned that semantic extraction systems [214, 329, 330] and semantic math taggers [71, 402] have difficulties in extracting the essential components (MOIs) from complex expressions. Considering the definition of the Jacobi polynomial in Equation (3.2), it would be beneficial to extract the groups of tokens that belong together, such as $P_n^{(\alpha,\beta)}(x)$ or $\Gamma(\alpha+m+1)$. With our proposed search engine for retrieving MOIs, we are able to facilitate semantic extraction systems and semantic math taggers. Imagine such a system being capable of identifying the term 'Jacobi polynomial' from the textual context. Figure 3.7 shows the top relevant hits for the search query 'Jacobi polynomial' retrieved from zbMATH and arXiv. The results contain several relevant and related expressions, such as the constraints $\alpha,\beta>-1$ and the weight function for the Jacobi polynomial $(1-x)^\alpha(1+x)^\beta$, which are essential properties of this orthogonal polynomial. Based on these retrieved MOIs, the extraction systems can adjust its retrieved math elements to improve precision, and semantic taggers or a tokenizer could re-organize parse trees to more closely resemble expression trees.

3.2.6 Outlook

In this first study, we preserved the core structure of the MathML data which provided insightful information for the MathML community. However, this makes it difficult to properly merge formulae. In future studies, we will normalize the MathML data via MathMLCan [117]. In addition to this normalization, we will include wildcards for investigating distributions of formula patterns rather than exact expressions. This will allow us to study connections between math objects, e.g., between $\Gamma(z)$ and $\Gamma(x+1)$. This would further improve our recommendation system and would allow for the identification of regions for parameters and variables in complex expressions.

3.3 Semantification with Textual Context Analysis

The results of our math embedding experiments and the introduction of MOI motivates us to develop a context-sensitive LaTeX to CAS translation approach around the MOI concept. In this section, we briefly discuss our novel approach to perform context-sensitive translations from LaTeX to CAS, which concludes research task **II**. We focus on three main sources of semantic information to disambiguate mathematical expressions sufficiently for such translations:

1. the inclusive structural information in the expression itself;
2. the textual context surrounding the expression; and
3. a common knowledge database.

The first source is what most existing translators rely on by concluding the semantics from a given structure. The second source is rather broad. The necessary information can be given in the sentences before and after an equation, somewhere in the same article, or even through references (e.g., hyperlinks in Wikipedia articles or citations in scientific publications). In this thesis, we will focus on the textual context in a single document, i.e., we do not analyze references or deep links to other articles yet. The last source can be considered a backup option. If we cannot retrieve information from the context of a formula, the semantic meaning of a formula might be considered common knowledge, such as π referring to the mathematical constant.

We extract knowledge from each of the three sources with different approaches. For the inclusive structural information, we rely on the semantic LaTeX macros developed by Miller [260] for the DLMF that define standard notation patterns for numerous OPSF. To analyze the textual context of a formula, we rely on the approach proposed by Schubotz et al. [330], who extracted noun phrases to enrich identifiers semantically. As a backup common knowledge database, we use the POM tagger developed by Youssef [402] that relies on manually crafted lexicon files with several common knowledge annotations for mathematical tokens.

3.3.1 Semantification, Translation & Evaluation Pipeline

Figure 3.8 illustrates the pipeline of the proposed system to convert generic LaTeX expressions to CAS. The figure contains numbered badges that represent the different steps in the system. Steps 1-4 represent the conversion pipeline, while steps 5-7 are different ways to evaluate the system.

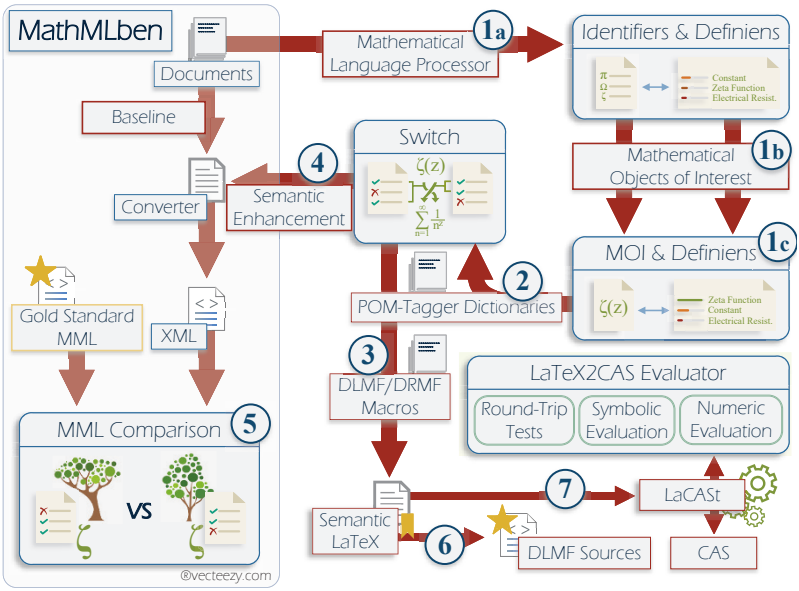


Figure 3.8: Pipeline of the proposed context-sensitive conversion process. The pipeline consists of four semantification steps (1-4) and three evaluation approaches (5-7).

The conversion pipeline starts with *mathosphere*[32] (step 1a). Mathosphere is the Java framework developed by Schubotz et al. [279, 329, 330] in a sequence of publications to semantically enrich mathematical identifier with defining phrases from the textual context. First, we will modify mathosphere so that it extracts MOI-definiens pairs rather than single identifiers (step 1b). For this purpose, we propose the following significant simplification: an isolated mathematical expression in a textual context is considered essential and informative. Hence, *isolated formulae* are defined as MOI. Moreover, mathosphere scores identifier-definiens pairs in regard of their first appearance in a document (since the first declaration of a symbol often remains valid throughout the rest of the document [394]). We adopt this scoring for MOI with a matching algorithm that allows us to identify MOI within other MOI in the same document (step 1c).

Step 2 is currently optional and combines the results from the MOI-definiens extraction process with the common knowledge database of the POM tagger. The information can then be used to feed existing LaTeX to MathML converters with additional semantic information. In Chapter 2, we created a MathML benchmark, called MathMLben, to evaluate such converters. We have also shown that, for example, LaTeXML can adopt additional semantic information via given semantic macros. Hence, via step 4 (and subsequently step 5) we can evaluate our semantification so far with the help of existing converters. The steps 2, 4, and 5 are not subject of this thesis but part of upcoming projects.

[32] https://github.com/ag-gipp/mathosphere [accessed 03-24-2020]

Besides this optional evaluation over MathMLben, we continue our main translation path. Once we extracted the MOI-definiens pairs, we replace the generic LaTeX expressions by their semantic counterparts (step (3)). We do so by indexing semantic LaTeX macros so that we can search for them by textual queries. Afterward, we are able to retrieve semantic LaTeX macros by the previously extracted definiens. Finally, we create replacement patterns so that the generic LaTeX expression can be replaced with the semantic enriched semantic macros from the DLMF. The result should be semantic LaTeX, which enables another evaluation method. Consider we perform this pipeline on the DLMF, we can compare the generated semantic LaTeX with the original, manually crafted semantic LaTeX source in the DLMF to validate its correctness (step (6)). Unfortunately, the entire pipeline focuses on the textual context. The DLMF does not provide sophisticated textual information because semantic information is available via special infoboxes, through hyperlinks, or in tables and graphs. A more comprehensive evaluation approach can be enabled by further translating the expressions to the syntax of CAS via LaCaST as we have shown in previous projects [2] (step (7)), namely symbolic and numeric evaluations. Moreover, this evaluation is most desired since it evaluates the entire proposed translation pipeline, from the semantification via mathosphere and the semantic LaTeX macros, and the final translation via LaCaST. The next chapter will aim to realize this proposed pipeline. The steps (1) and (3) are discussed in Chapter 4. The step (7) is subject of Chapter 5. Step (6) has not been realized due to the reduced amount of textual context within the DLMF. Steps (2), (4), and (5) are subject of future work.