BERT Is Not The Count: Learning to Match Mathematical Statements with Proofs

Weixian (Waylon) Li

Master of Science by Research
Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh
2023

Abstract

We introduce a task consisting in matching a proof to a given mathematical statement. The task fits well within current research on Mathematical Information Retrieval and, more generally, mathematical article analysis (Mathematical Sciences, 2014). We release a dataset for the task (the MATcH dataset), consisting of over 180k statement-proof pairs extracted from mathematical research articles. We propose a bilinear similarity model and two decoding methods to match statements to proofs effectively. While the first decoding method matches a proof to a statement without being aware of other statements or proofs, the second method treats the task as a global matching problem. Through a symbol replacement procedure, we analyze the "insights" that pre-trained language models have in such mathematical article analysis and show that while these models perform well on this task with the best performing mean reciprocal rank of 73.7, they follow a relatively shallow symbolic analysis and matching to achieve that performance.

Acknowledgements

Completing my thesis has been an exciting journey, and I have greatly enjoyed the past year of research on this topic. Many individuals offered their assistance during this process, and I would like to take this opportunity to express my gratitude.

My sincere appreciation goes to my supervisor, Shay Cohen, for welcoming me into his research group as an MScR student. His guidance, instruction, and encouragement have been invaluable in helping me develop effective research habits. I also wish to thank Yftah Ziser, the postdoc in our group, for his valuable advice and assistance with project planning. The knowledge I have gained from them will undoubtedly benefit my future research career.

Finally, I want to thank my parents for their unwavering support over the past 24 years. Their unconditional love has allowed me to pursue my passions.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Weixian (Waylon) Li*)

Table of Contents

List of Tables

Chapter 1

Introduction

Research-level mathematical discourse is a challenging domain for Natural Language Processing (NLP). Mathematical articles switch frequently between natural language and mathematical formulae, and a semantic analysis of mathematical text needs to solve relationships (e.g. coreference) between mathematical symbols and concepts. Moreover, mathematical writing follows many conventions, such as variable naming or typography, that are implicit, and may differ between subfields. Researchers have proposed methods to solve some related tasks, most of which have not yet reach to human-level performance (Meadows and Freitas, 2022).

However, mathematical research can benefit from NLP (Mathematical Sciences, 2014), in particular as concerns bibliographical research: researchers need tools to find work relevant for their research. Indeed, prior NLP work on mathematical research articles focused on Mathematical Information Retrieval (MIR) and related tools or data (Zanibbi et al., 2016; Stathopoulos and Teufel, 2016, 2015). Figure 1.1 illustrates a simple example of MIR. By entering a mathematical formula as the query, the system should help search for the concept, object or result (Sakai et al., 2021).

Mathematical Information Retrieval has attracted more attention in recent years. The National Center for Science Information Systems (NTCIR), the first large-scale conference for information retrieval (IR), noticed the importance of math in technical documents, and the weakness of most search engines which do not support user to search for mathematical formulae in the target documents. They designed a series of NTCIR Math tasks (Zanibbi et al., 2016) and constructed the test collections, which proposed an evaluation framework of mathematical information retrieval.

This thesis introduces a more specific task of MIR aimed at improving the processing of research-level mathematical articles and make a step towards the modelling of

mathematical reasoning. We construct a dataset for the task (MATcH) and an encoder-decoder (Cho et al., 2014) based method to match statements to proofs effectively. We also focus on the insight of the statement-proof matching models to explore how decisions are made when judging if a proof match to the statement. Our analysis shows that pre-trained language models do not obtain significant "mathematical insight" for performing this matching, but rather rely on shallow matching. However, this does not prevent them from performing the matching relatively well in several carefully crafted scenarios.

The aim of this project is to publish a more professional MIR dataset and task for future research and also let the models consider a more realistic scenario of how people use MIR technique to look up information they need. This work is based on our ongoing submission of Empirical Methods in Natural Language Processing (EMNLP) 2022 conference[1]. It is related to the previous work of Coavoux and Cohen (2021). They constructed the first version of dataset and proposed the model architecture. We use the same technique to construct our dataset and replace their encoder by more powerful pretrained language models for our experiments.

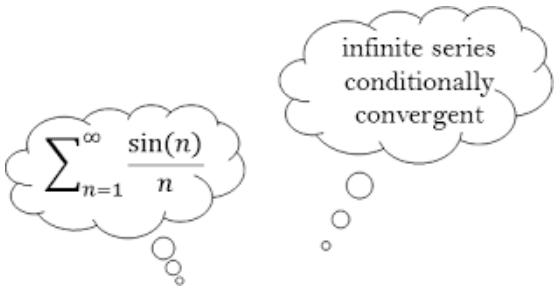

Figure 1.1: A simple example of MIR. The left-hand-side formula is the query and the right-hand-side text is the result retrieved from the system which explains the query formula (Dadure et al., 2021a).

1.1 Motivation

There are multiple motivations for the design of the task and our dataset. We believe it may help MIR by serving as a proxy for the search for the existence of a mathematical result, or for statements and proofs related to one another (e.g. using the same proof technique), an important search tool for any digital mathematical library (Mathematical Sciences, 2014). Learning to match statements and proofs would also benefit

[1] https://2022.emnlp.org/

computer-assisted theorem proving, as it is akin to tasks such as premise selection, also recently addressed with NLP methods (Piotrowski and Urban, 2019).

Existing datasets such as LEANSTEP (Han et al., 2021) and the synthetic dataset of Polu and Sutskever (2020) do not include natural language. Figure 1.2 and Figure 1.3 respectively demonstrate the datapoint in the LEANSTEP and synthetic dataset. It is not reasonable to expect all the users to have the prior knowledge for reading and writing the statements and proofs in these two special formats. Figure 1.4 shows an example data from our dataset. It is written in natural language, which matches how people do web search via search engines. NaturalProofs (Welleck et al., 2021), another related dataset, only consists of 32k theorem-proof pairs from ProofWiki[2], some sub-topics in algebraic geometry and two textbooks. Our dataset is over five times larger and contains pairs extracted from professional mathematical papers.

```
lemma peirce_identity {P Q :Prop} : ((P → Q) → P) → P :=
begin
apply or.elim (em P),
intros h _,
exact h,
tauto!
end
```

Figure 1.2: Example data from LEANSTEP, namely the proof of the Peirce identity (Han et al., 2021). The data is written in the format which LEAN can understand. LEAN is an open-source theorem prover proposed by de Moura et al. (2015). The system requires fully specified, type-correct expressions as the input. Users have to manually translate the mathematical statements to this format. Otherwise, LEAN will not be able to process the input properly. Also, users need to learn the format in order to understand the proof given by the prover.

1.2 Contributions

The contributions of this thesis are summarized as follows:

[2]https://proofwiki.org/xmldump/latest.xml

```
{
  "proof_label": "unidmrn",
  "goal": "[[ ]] |- U. U. `' A = ( dom A u. ran A )",
  "proof_step": "[[ |- A = B |- C = B ]] |- A = C \\
     {{ A : U. U. `' A }} \\
     {{ B : ( ran `' A u. dom `' A ) }} \\
     {{ C : ( dom A u. ran A ) }}",
  "proof_step_hash": "37yZVNorgF8=",
  "parent_hash": ["n4Kl7judEN4="]
}
```


Figure 1.3: Example data from the synthetic dataset of Polu and Sutskever (2020). The dataset is in Metamath (Megill, 1969) environment, a tiny language that can express theorems in abstract mathematics, accompanied by proofs that can be verified by a computer program. The dataset is in tree structure. Every statement has a proof tree and each datapoint is a node on one of the proof trees, representing a step of proof. `proof_label` indicates which statement the proof step belongs to. `goal` is the target this specific proof step needs to achieve. `parent_hash` tells the parent goal if any. By collecting all the proof steps with the same `proof_label`, we can recover the tree structure of the proof.

- We introduce the MATcH task aimed at improving the MIR research. It is a practical scenario for many mathematical researchers and students for learning purpose.
- We construct and release a dataset for the task (MATcH) from over 439k professional articles in the MREC corpus[3] (Líška et al., 2011). The dataset consists of over 180k statement-proof pairs, which is one of the largest profession-level mathematical statement-proof datasets.
- We provide results on our proposed task with an array of neural models, aimed at scoring the likelihood of relationship between a statement and its proof. We also provide an analysis of our models through the **symbol replacement procedure**.
- We provide two methods for decoding, one is local decoding, matching a proof

[3] https://mir.fi.muni.cz/MREC/, version 2011.4.439.

Theorem 1.3. *Suppose that* $|\mathrm{Sing}(S)| < (2r-1)r$. *Then X is factorial.*
Proof. The subset $\mathrm{Sing}(S) \subset \mathbb{P}^3$ is a set-theoretic intersection of surfaces of degree $2r-1$, which implies that X is factorial by Theorem 1.1.

Figure 1.4: Example of a statement-proof pair.

to a statement in a greedy way, and one that provides a global bipartite matching based on a structured max-margin objective. Such an architecture may have applications to other NLP problems that can be cast as maximum bipartite matching problems, which is the case, for example, for some alignment problems Taskar et al. (2005); Padó and Lapata (2006).

- We provide the results of qualitative analysis on our model, which sheds light on the features that affect the model predictions.

1.3 Thesis Outline

Chapter 2 presents the prior background knowledge and related work of this thesis. We will provide an introduction to the history of MIR and some existing datasets and MIR models proposed by other researchers. We also introduce some basic concepts and development of pretrained language models (PLMs), some of which will be used as the encoder of our model. Then we briefly demonstrate the encoder-decoder architecture, the basal structure of our model. Lastly, we give details about how we evaluate the models for our MIR task (MATcH).

Chapter 3 provides the details of dataset construction, including the introduction to the source corpus and its format, the method we use to identify the statement-proof pairs from the articles and how we pre-process the data.

Chapter 4 discusses the symbol replacement setup. We start with the motivation of designing this procedure. Then we show examples for the four levels of symbol replacement which clearly illustrate the differences. Finally, we provide more information about how symbol replacement is done in practical.

Chapter 5 presents our encoder-decoder based model. Firstly we introduce two types of encoders and the trainable bilinear similarity function which used for measuring if the given proof matches the statement. We also discuss more detail regarding the two aforementioned decoding algorithms, local decoding and global decoding, as two options for the decoder.

Chapter 6 gives introduction to two training method with different objective functions, local training and global training.

Chapter 7 shows the details of the experiments setup and the results obtained. We provide the hyper-parameter setting and time consumption for all the experiments carried out. We also present the results of qualitative analysis to shed light on the insight of our model.

Chapter 8 concludes the thesis with a discussion of the limitations and suggests potential directions for future work.

Chapter 2

Background

In this chapter, we review the history and related work of Information Retrieval (IR), and especially Mathematical Information Retrieval (MIR) in Section 2.1. We focus on the datasets the researchers used in the past and the existing methods they proposed, with a discussion of their advantages and disadvantages. Then, in Section 2.2, we introduce the development of pretrained language models (PLMs) which will be used as the encoder of our model. As our model is based on encoder-decoder architecture, we then demonstrate this structure proposed by Cho et al. (2014) in Section 2.3. Finally, we explain the metrics we are going to use for evaluating the models in Section 2.4.

2.1 History

In this section, we review the history of Information Retrieval (IR) and Mathematical Information Retrieval (MIR) and a series of previous approaches to solve the MIR problem.

2.1.1 History of IR and MIR

Bush (1945) is the first one who described the idea of using computers to search for information. Holmstrom detailed the idea on the conference held by the UK's Royal Society in 1948 (BERNAL et al., 1948). It became the first conference where computer started to be used for Information Retrieval. After then in 1950s, several automated Information Retrieval systems are proposed. Luhn (1957) introduced a system which used words as indexing units and assigned scores for documents to indicate the relevance to a given query. The documents with the top ranks were returned to the user.

This is the foundation of ranked retrieval method. In 1960s, Gerard Salton formed the a Information Retrieval research group at Cornell and proposed the SMART system in 1971 (Salton, 1971). Concepts in Information Retrieval such as vector space model, relevance feedback, and Rocchio classification were developed in this research. The system also allowed users to do experiments for improving searching quality and efficiency. 1970s saw many different retrieval techniques built on the advances of the previous methods. One of the most notable work was the large-scale retrieval systems, Lockheed Dialog system. Bourne and Hahn (2003) reviewed the development of the Lockheed Dialog system from 1961 to 1972 including many technical details.

The aforementioned models were proven to be well-performed on small text corpora such as the Cranfield collection (Singhal and Google, 2001), but there was no evidence to show they were scalable to large text collections. Therefore, constructing larger corpora became the crucial thing to push the Information Retrieval techniques to next generation. Text Retrieval Conference[1] (TREC), a series of evaluation conferences, began in 1992 as part of the TIPSTER Text program sponsored by several US Government agencies (Harman, 1993). It aimed at supporting and encouraging research within IR from large corpus by providing researchers with large text collections. Inspired by this, new techniques bloomed in 1990s. Korfhage (1997) and Baeza-Yates and Ribeiro-Neto (1999) were two of the representative work published in late 1990s. In the meantime, IR algorithms had been employed for searching the World Wide Web, which let more and more users benefit from the evolution of IR techniques. Nowadays, web search engines already become a necessary tool for all the Internet users.

The development of Mathematical Information Retrieval (MIR) was not as early as IR. However, thanks to the experience obtained in the previous decades, researchers noticed the importance of large corpus construction in order to build more powerful and reliable IR systems. Líska et al. (2011) constructed a corpus of mathematical texts called MREC, containing 439,423 scientific documents with over 158 million mathematical formulae. The major difference between MIR and traditional IR is that the system needs to have the ability to deal with mathematical expression. So they proposed a math aware, full-text based search engine called MIaS (Math Indexer and Searcher), which addressed the problem. NII Testbeds and Community for Information access Research (NTCIR) started to focus on the MIR task in 2013. Prior to that, limited number of researchers had the chance to access the digital mathematics libraries and MIR had not been paid much attention by the IR community (Aizawa and Kohlhase,

[1] http://trec.nist.gov

2021). NTCIR organized three mathematical tasks from NTCIR-10 to NTCIR-12:

- The NTCIR-10 Math Pilot Task (Aizawa et al., 2013) was the first organized task for mathematical formula search. There were two subtasks involved, one of which was an MIR task: given a math query, the goal was to retrieve relevant documents. The other subtask was to identify the descriptive context given a math formula, which was more related to math understanding. According to the submissions, participants showed more interests on the MIR task and considered it more important.

- The NTCIR-11 Math-2 Task (Aizawa et al., 2014) considered the feedback collected from NTCIR-10 and regenerated the dataset using the latest conversion tool at the time. Given a query, participant systems needed to estimate the relevant paragraphs in the dataset and return the ranked list of the documents containing a matching formula or keywords. Additionally, NTCIR-11 Math-2 Task provided an extra optional subtask using mathematical articles on Wikipedia. The dataset was easier for understanding compared to the main task dataset constructed from arXiv[2].

- The NTCIR-12 MathIR Task (Zanibbi et al., 2016) reused the arXiv corpus of NTCIR-11 Math-2 Task and implemented with more topics. The Wikipedia corpus was also upgraded for unprofessional use cases. The design of the task was similar to NTCIR-11 but the query could be formula+keyword instead of pure formula.

MIR attracted more and more attentions since these NTCIR tasks were released. Researchers started to explore methods to encode the semantic information of mathematical formulae and improve the performance of MIR systems, which will be further discussed in Section 2.1.2.

2.1.2 Early works on MIR

Modern methods of Mathematical Information Retrieval (MIR) were designed for three purposes, document retrieval, formula retrieval, and document synthesis (Guidi and Coen, 2015).

[2]https://arxiv.org/

Document retrieval Given a query combining keywords, free text and mathematical formulae. The system returns a ranked list of documents that match the query.

Formula Retrieval Given a query formula E where E can also be a set of formulae, the system retrieves all formulae that are in some relation $\mathcal{R}$ with the query.

Document synthesis Given a query, sometimes not in natural language but in an expressed query language, the system composes a new mathematical document that is relevant.

Modern NLP works designed for the three purposes above improved MIR using encoding based method, most of which focus on establishing connections between mathematical formulae and natural language text in order to improve the representation of formulae.

The interpretation of variables is highly dependent on the context. For example, the symbol E could denote an expectation in a statistics article, or the energy in a physics article. Some studies use the surrounding context of a formula to assign a definition or a type to the whole formula, or to specific variables. Nghiem Quoc et al. (2010) focus on identifying coreferences between mathematical formulae and mathematical concepts in Wikipedia articles. Kristianto et al. (2012) extract definitions of mathematical expressions. Grigore et al. (2009), Wolska et al. (2011) and Schubotz et al. (2016) disambiguate mathematical identifiers, such as variables, using the surrounding textual context. Stathopoulos et al. (2018) inferred the type of a variable in a formula from the textual context of the formula. Another line of work focused on identifying specialized terms or concepts to improve MIR (Stathopoulos and Teufel, 2015, 2016).

Some work adapted standard NLP tools to the specificity of mathematical discourse, e.g. POS taggers (Schöneberg and Sperber, 2014), with the objective of using linguistic features to improve the search for definitions of mathematical expressions (Pagel and Schubotz, 2014). More recent work focused on typing variables in mathematical articles (Ferreira et al., 2022), modeling formulae (Mansouri et al., 2019; Dadure et al., 2021b), and selecting premises (Ferreira and Freitas, 2020, 2021).

2.2 Pretrained Language Models

Pretrained language models (PLMs) are deep learning models trained over large text corpus.

Skip-Gram (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) are the first generation of PLMs in which the word embeddings are static and not able to represent word senses in different context. Since the development of deep learning comes to a new era, varieties of neural networks such as convolutional neural networks (Kalchbrenner et al., 2014; Kim, 2014; Gehring et al., 2017, CNNs), recurrent neural networks (Sutskever et al., 2014; Liu et al., 2016, RNNs), and attention mechanisms, more specifically, the transformer architecture (Bahdanau et al., 2016; Vaswani et al., 2017a), have been applied to construct PLMs that generate contextual embeddings. Some notable methods are ELMo (Peters et al., 2018, RNN based), OpenAI GPT (Radford and Narasimhan, 2018, transformer based) and BERT (transformer based) (Devlin et al., 2018, transformer based), which are called the second generation PLMs.

The purpose of pretraining is to obtain a large neural network that can "understand" the language and act as a language encoder for downstream tasks. Saunshi et al. (2020) explored the success of PLMs on downstream tasks and provided mathematical explanations by reformulating the tasks as sentence completion problems. Their work shows the evidence of how PLMs help with the NLP tasks. As one of the most popular downstream tasks in Natural Language Processing, Information Retrieval is also boosted by PLMs.

Figure 2.1 illustrates the pipeline of PLMs being used for downstream tasks. The traditional procedure is that language model is first pretrained on a large text corpora and then transferred to a smaller task-specific dataset for finetuning. It will finally return a model capable for the aforementioned task. Gururangan et al. (2020) presented a study to demonstrate the importance of both domain-adaptive pretraining (an intermediate indomain pretraining step between pretraining and finetuning on task-specific dataset) and task-adaptive pretraining (adapating to the task's unlabeled data, which is the traditional finetuning step), which sheds light on the two-stage finetuning illustrated in Figure 2.1.

2.3 Encoder-decoder Architecture

The encoder-decoder structure was first designed to address the case that input sequence and output were both variable-length sequences in translation task (Cho et al., 2014). For example, an English word "they" will be translated to two Chinese characters.

Figure 2.2 simply illustrates the encoder-decoder architecture. The structure con-

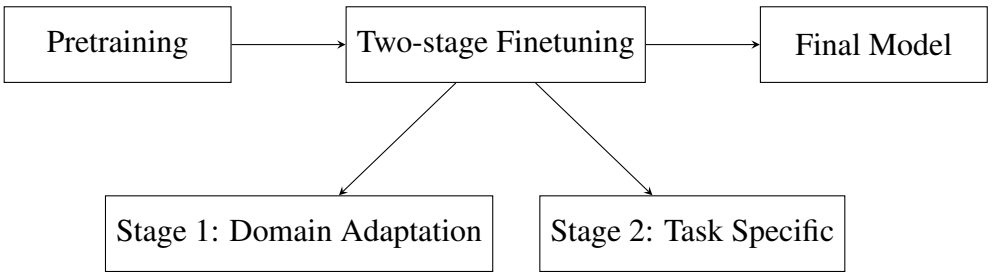

Figure 2.1: Illustration of pretraining and finetuning the model. The model is fed a large amount of unlabelled text data at the first step. In this step, the model forms the parameters and learns how the language is written in general. Two-stage finetuning consists of two steps. The first stage is optional where the model can continue being trained on domain-specific corpus. The other stage is further training the PLM on a labelled small task-specific dataset to get the final model.

tains two connected networks, encoder and decoder. After the encoder accepts a source sequence input, it compresses the variable-length input to fixed-length vectors which we call the hidden states (Yang et al., 2020). This is the procedure of encoding. During decoding, the decoder takes the final hidden state produced from the encoder and processes and converts the vectors to the form we need such as text and probabilities. Some PLMs such as BERT acts as an encoder in the encoder-decoder architecture and the word embeddings generated from BERT are the hidden states. The finetuning layer can be treated as the decoder. In IR, the decoder uses the word embeddings to measure the similarities between the queries and documents.

Figure 2.2: A simple illustration of encoder-decoder architecture.

2.4 Evaluation of MIR Models

In this thesis, we use two evaluation metrics. Assumming that a system predicts a ranking of proofs, instead of providing only a single proof, we evaluate its output with the Mean Reciprocal Rank (MRR) measure. Assuming the user starts looking down ranked list until the target document is found, $\hat{r}_i$ is the rank of the gold document for the i-th query as predicted by the system and N is the total number of documents. The calculation of MRR is shown in Equation 2.1. Craswell (2009) points out that MRR is

a proper evaluation metric for known item search which means the user has seen the exact target document before.

$$\mathrm{MRR}(\{\hat{r}_i\}_{i\in\{1,\dots N\}}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\hat{r}_i} \tag{2.1}$$

As a second evaluation metric, we use a simple accuracy, i.e. the proportion of queries whose first-ranked document is correct (shown in Equation 2.2).

$$\mathrm{Acc} = \frac{\text{Number of queries whose first-ranked document is correct}}{\text{Number of queries}} \tag{2.2}$$

In our case, queries are the mathematical statements and documents are the proofs. By construction (see Section 3.3), it is possible though unlikely that the same mathematical statement occurs several times in the dataset. It is more unlikely that several occurrences have exactly the same formulation and use the same variable names. Therefore, we consider a match to be correct if and only if it is associated with its original proof.

Chapter 3

Dataset Construction

This chapter describes the construction of the MATcH dataset of statement-proof pairs (see Figure 1.4 for an example).

3.1 Source Corpus

In this project, we use the MREC corpus[1] (Líška et al., 2011) as a source. The MREC corpus contains around 450k articles from ArxMLiV (Stamerjohanns et al., 2010), an on-going project aiming at converting the arXiv[2] repository from LaTeX to XML, a format more suited to machine processing, using the LaTeXML[3] tool. Instead of copying all the content from arXMLiv corpora, they make sure it only contains XML documents marked as converted successfully so that no problem will be issued in practical usage. The source corpus covers several scientific domains such as Physics, Mathematics, Computer Science, Quantitative Biology, Quantitative Finance and Statistics. In this collection, mathematical formulae are represented in the MathML[4] format, a markup language.

3.2 MathML

Mathematical Markup Language (MathML) (Ausbrooks et al., 2010) is an application of XML to represent mathematical formulae, which can also capture the stucture and

[1] https://mir.fi.muni.cz/MREC/, version 2011.4.439.
[2] https://arxiv.org/
[3] https://math.nist.gov/~BMiller/LaTeXML/
[4] https://www.w3.org/Math/

content. The purpose of using MathML is to properly show mathematical formulae on World Wide Web pages and other documents.

MathML has two versions with different flavours: Presentation MathML and Content MathML. Presentation MathML aims at capturing the notational structure and focuses on displaying the equations. Rather than the layout, Content MathML pays more attention on the meaning of the expressions. Figure 3.1 and Figure 3.2 respectively shows an example of Presentation MathML and Content MathML for the formula: $ax^2 + bx + c$. The mathematical contents in the MREC corpus are written in Presentation MathML, which is better for displaying on the webpage. Since BERT takes linear sequences as input, using Presentation MathML will also be easier to recover the original tokens order in the documents.

```
<math>
 <mi>a</mi>
 <msup>
    <mi>x</mi>
    <mn>2</mn>
 </msup>
 <mo>+</mo>
 <mi>b</mi>
 <mi>x</mi>
 <mo>+</mo>
 <mi>c</mi>
</math>
```

Figure 3.1: A Presentation MathML example to represent the formula: $ax^2 + bx + c$.

Presentation MathML contains token elements and layout elements. The top three frequent token elements are:

- `<mi>a</mi>`: identifiers
- `<mo>+</mo>`: operators
- `<mn>2</mn>`: numbers

Some significant layout elements are:

```
<math>
    <apply>
        <plus/>
        <apply>
            <times/>
            <ci>a</ci>
            <apply>
                <power/>
                <ci>x</ci>
                <cn>2</cn>
            </apply>
        </apply>
        <apply>
            <times/>
            <ci>b</ci>
            <ci>x</ci>
        </apply>
        <ci>c</ci>
    </apply>
</math>
```

Figure 3.2: A Content MathML example to represent the formula: $ax^2 + bx + c$.

- `<mrow>`: a horizontal row of items
- `<msup></msup>`: superscripts
- `<msub></msub>`: subscripts
- `<mfrac></mfrac>`: fractions
- `<<msqrt>></<msqrt>>`: roots

In the example shown in Figure 3.1, we can observe all the top three token elements and the superscript layout element.

3.3 Construction

Statement-proof identification For each XML article (corresponding to a single arXiv article), we extract pairs of consecutive `<div>` tags such that: (i) the `class` attribute of the first `div` node contains the string `"theorem"`; (ii) the `class` attribute of the second `div` node is the string `"proof"`. Articles that do not contain such pairs of tags are discarded, as well as articles that are not written in English (representing 143 articles in French, 11 in Russian, 5 in German, 2 in Portuguese and 1 in Ukrainian), as identified by the `polyglot` Python package.[5]

In the remaining collection of pairs of statements and proofs, we filter out pairs for which either the statement or the proof is too short.[6] Indeed, the short texts were often empty (only consisting of a title, e.g. "5.26 Lemma."), which we attribute to the noise inherent to the conversion to XML, or not self-contained. In particular, we identified several prototypical cases:

- Omitted (or easy) proofs contain usually a single word ('omitted', 'straightforward', 'well-known', 'trivial', 'evident'), but are sometimes more verbose ('This is obvious and will be left to the readers').
- Proofs that consist of a single reference to
  - An appendix ('See Appendix A');
  - Another theorem ('This follows immediately from Proposition 4.4 (ii).');
  - The proof method of another theorem ('Similar to proof of Lemma 6.1')
  - Another article ('See [BK3, Theorem 4.8].');
  - Another part of the article ('The proof will appear elsewhere.', 'See above.', 'Will be given in section 5.').

Filtering on the number of tokens also exclude self-contained short proofs, such as 'Take $Q\prime = ph_i - p_i$.' However, such proofs were very infrequent on manual inspection of the discarded pairs (2 in a manually inspected random sample of 100 discarded proofs).

[5] www.github.com/aboSamoor/polyglot/

[6] We used a minimum length of 20 tokens for both statements and proofs, based on a manual inspection of the shortest examples. We also exclude proofs and statements longer than 500 tokens.

Statements	Proofs	%
20	7	0.0
10	80	0.2
5	1027	1.9
2	11949	22.6
= 1	19531	37.0
<1	21275	40.3

Table 3.1: Cumulative distribution of proofs in the development set, by number of statements to which they are assigned with the local decoding method.

Preprocessing: linearizing equations Mathematical formulae in the XML articles are enclosed in a `<math>` markup tag, that materializes the switch to the MathML format, and whose internal structure represents the formula as an XML tree. As a preprocessing step, we linearize each formula to a raw sequence of strings.

In MathML, an equation can be encoded in a content-based (semantic) way or in a presentational way, using different sets of markup tags, as we mentioned before. We first convert all MathML trees to presentational MathML using the XSL stylesheet from the Content MathML Polyfill repository.[7] Then we perform a depth-first search on each tree rooted in a `<math>` tag to extract the text content of the whole tree.

During this preprocessing, we tested several processing choices:

- **Font information**. In mathematical discourses, fonts play an important role. Their semantics depend on conventions shared by researchers. If both x and $\mathbf{x}$ appear in the same article, they are most likely to represent different mathematical objects, e.g. a scalar and a vector. Therefore, we use distinct symbols for tokens that are in distinct fonts.

- **Math-English ambiguity**. Some symbols can be used both in natural language text and in formulae. For example, 'a' can be a determiner in English, or a variable name in a formula. To avoid increasing ambiguity when linearizing formula, we type each symbol (as math or text) to make the mathematical vocabulary completely disjoint from the text vocabulary.

Both these preprocessing steps had a beneficial effect on the baselines in preliminary experiments.

[7] https://github.com/fred-wang/webextension-content-mathml-polyfill

Number of articles in the MREC corpus		439,423
Extracted articles with statement-proof pairs		27,841
Total number of statement-proof pairs		184,094
Number of (primary) categories		(120) 135
Average number of categories per article		1.7
Most represented primary categories	# articles	# pairs
math.AG Algebraic Geometry	2848	22029
math.DG Differential Geometry	2030	12440
math.CO Combinatorics	1705	10548
math.GT Geometric Topology	1539	9234
math.NT Number theory	1454	9521
math.PR Probability	1422	7660
math.AP Analysis of PDEs	1386	6981
math-ph Mathematical Physics	1249	6491
math.FA Functional Analysis	1143	8011
math.GR Group Theory	970	7806
math.DS Dynamical System	961	6424
math.QA Quantum Algebra	944	8074
math.OA Operator Algebras	923	8050

Table 3.2: Statistics about the dataset and categories of mathematical articles

Statistics We extract statement-proof pairs as described above. Our processing of MREC includes the identification of statement-proof pairs through meta tags and the linearization of the representation of mathematical equations.

We report in Table 3.2 some statistics about the dataset we collected. The extracted articles were from a diverse set of mathematical subdomains, and connected domains, such as computer science (746 articles from 30 subcategories) and mathematical physics (2562 from 31 subcategories). There are in average 6.6 statement-proof pairs per article. Table 3.1 depicts the cumulative distribution of proofs and the number of statements they are assigned to.

We report statistics about the size of statements and proofs in number of tokens in Table 3.3. We report the number of tokens in formulae (math), in the text itself (text) and in both (text+math). On average, proofs are much longer than statements. Statements and proofs have approximately the same proportion of text and math. Overall, the variation in number of tokens across statements and proofs is extremely high, as

Statements	Min	Max	Mean±SD
Text+math	20	500	80±57
Text only	1	398	30±20
Math only	0	470	58±20
Math proportion	0%	99.5%	58%±20
Proofs			
Text+math	20	500	210± 127
Text only	1	467	81 ± 56
Math only	0	495	129 ± 96
Math proportion	0%	99.6%	56%± 21

Table 3.3: Number of tokens in the dataset. We report for statements and proofs the minimum, maximum and average number of tokens broken down by type (‘math’ for tokens extracted from formulae and ‘text’ for the others). A value of 0 for, e.g. the ‘math only’ row, means that the statement or proof does not contain mathematical symbols or formulae.

illustrated by the standard deviation (SD) of all presented metrics.

Chapter 4

Symbol Replacement

This chapter introduces the motivation and the technical details about our symbol replacement method.

4.1 Motivation of Symbol Replacement

With our current dataset setup, we implicitly make the assumption that both the theorem and the proof are authored by the same authors. This assumption is incongruent with the MIR-flavor of our task. First, it is not useful for researchers to match proofs they authored. Second, each person has a unique writing style expressed by unique mathematical jargon and notations. For example, Aggarwal (2017) wrote the Abel's Theorem as the form shown in Figure 4.1. The Abel's Theorem page on Wikipedia[1] introduces the theorem in another way with different mathematical notations shown in Figure 4.2. In such case, the proof provided by Aggarwal (2017) might not be retrieved by simple mathematical symbol matching while using the Abel's Theorem on Wikipedia as the query.

To relieve of this assumption, we introduce four symbol replacement levels for changing the names of the proof's variables. Then, we train and test our models using these altered datasets. These replacement levels also provide insight on the ability of our models to semantically analyze the input statement-proof pairs, which allow us to explore what information have the models learned from the data.

[1] https://en.wikipedia.org/wiki/Abel%27s_theorem

Let $f(x) = \sum_{n=0}^{\infty} a_n x^n$ be a power series with finite positive radius of convergence R. If the series converges at $x = R$, then f is continuous at $x = R$. If the series converges at $x = -R$, then f is continuous at $x = -R$.

Figure 4.1: Abel's Theorem represented in Aggarwal (2017)'s paper.

Let the Taylor series $G(x) = \sum_{k=0}^{\infty} a_k x^k$ be a power series with real coefficients a_k with radius of convergence 1. Suppose that the series $\sum_{k=0}^{\infty} a_k$ converges. Then $G(x)$ is continuous from the left at $x = 1$, that is,

$$\lim_{x \to 1^-} G(x) = \sum_{k=0}^{\infty} a_k$$

Figure 4.2: Abel's Theorem represented on Wikipedia.

4.2 Replacement Levels

We propose different levels of symbol replacement, focusing on mathematical notation. More precisely, we aim to replace the proof variable names if they appear in the statement without damaging the proof semantics. As we mentioned, we also expect to simulate the situation that the statement and proof are written by different authors. To do that, we change symbols that appear both in the proof and the statement. We do not change constant symbols such as π, as they often carry semantic meaning outside of the proof scope.

We experiment with four levels of symbol replacement (examples in parenthesis):

- **Symbol conservation:** All symbols remain intact, so the theorem and the proof overlap. All previous work uses that. $(a_n = a_{n-1} + a_{n-2})$
- **Partial symbol replacement:** A fraction of α of all the symbols in the proof remain the same, and the rest are changed. In our experiments, we use $\alpha = 0.5$. $(x_n = x_{n-1} + x_{n-2})$
- **Full symbol replacement:** All symbol names are changed ($\alpha = 1.0$ as above). $(x_i = x_{i-1} + x_{i-2})$
- **Symbol transposition:** We permute the variables' names such that no symbol remains the same, thus changing their original functionality. $(n_a = n_{a-1} + n_{a-2})$

4.3 Details

We follow the following rules when replacing symbols:

- Only the proof symbols are being replaced as there is no need to replace both statement and proof symbols. In Figure 4.3, we do not replace the mathematical symbols in the statement.
- We replace symbols only if they appear in both the statement and the proof. In Figure 4.3d, only α, T, r, A, G and λ are replaced by random English or Greek letters. γ, ρ and s remains unchanged since they only appear in the proof.
- We preserve the format of the mathematical symbols. For example, in Figure 4.3e, when we create the mapping $a \to t$, we will also apply $A \to T$ on A.
- We do not replace the double-struck letters, e.g., $\mathbb{R}$, since they usually represent fields and constant. We do not replace standard constant symbols such as π.
- We observe that some variables carry meaning. For example, the variable dim can be interpreted as $d * i * m$ or as one variable named dim. To avoid this ambiguity, we find all the variables' names that are longer than two English letters and use an English spellchecker[2] to detect which of those are English words. Suppose a multiple-letter variable is detected as an English word. In that case, we will add it to the denylist to ensure we do not treat it as multiple single-letter variables' names. This can also be observed in Figure 4.3 where we preserve "ker" to avoid breaking the meaning of "kernel".
- While randomly picking mathematical symbols to create mappings, we avoid selecting those symbols already appeared in the proof. For example in Figure 4.3c, we will exclude g, a, h and s while picking an English letter to replace letter r. After we randomly pick o for replacement, we will remove o from the candidate list as well. This is also a consideration of keeping the original semantic information.

[2] https://pyenchant.github.io/pyenchant/

Theorem: Proposition 3.2. The map $\lambda \mapsto \lambda|_{G0}$ is a bijection between the left invariant sections of $|\Omega|^{\alpha}(\ker Tr)$ and the sections of $|\Omega|^{\alpha}(A(G))$. This bijection preserves smoothness as well as positivity.

(a) An example statement in the dataset.

Proof: The restriction is well defined, since $|\Omega|^{\alpha}(\ker Tr)$ contains $|\Omega|^{a}(A(G))$. To prove that the map in question is bijective, it is sufficient to check that its inverse is $\rho \mapsto \lambda$, where $\rho \in |\Omega|^{\alpha}(A(G))$ and $\lambda(\gamma) = \gamma \cdot \rho(s(\gamma))$.

(b) Symbol conservation which is also the original proof.

Proof: The restriction is well defined, since $|\Omega|^{\Psi}(\ker To)$ contains $|\Omega|^{\Psi}(A(H))$. To prove that the map in question is bijective, it is sufficient to check that its inverse is $\rho \mapsto \lambda$, where $\rho \in |\Omega|^{\Psi}(A(H))$ and $\lambda(\gamma) = \gamma \cdot \rho(s(\gamma))$.

(c) Partial symbol replacement with mapping $\alpha \to \psi$, $r \to o$, $G \to H$.

Proof: The restriction is well defined, since $|\Omega|^{\zeta}(\ker Xv)$ contains $|\Omega|^{\zeta}(B(J))$. To prove that the map in question is bijective, it is sufficient to check that its inverse is $\rho \mapsto \iota$, where $\rho \in |\Omega|^{\zeta}(B(J))$ and $\iota(\gamma) = \gamma \cdot \rho(s(\gamma))$.

(d) Full symbol replacement with mapping $\alpha \to \zeta$, $T \to X$, $r \to v$, $A \to B$, $G \to J$, $\lambda \to \iota$

Proof: The restriction is well defined, since $|\Omega|^{\lambda}(\ker Ga)$ contains $|\Omega|^{\lambda}(T(R))$. To prove that the map in question is bijective, it is sufficient to check that its inverse is $\rho \mapsto \alpha$, where $\rho \in |\Omega|^{\lambda}(T(R))$ and $\alpha(\gamma) = \gamma \cdot \rho(s(\gamma))$.

(e) Symbol transposition with mapping $\alpha \to \lambda$, $T \to G$, $r \to a$, $A \to T$, $G \to R$, $\lambda \to \alpha$.

Figure 4.3: An example for demonstrating the four symbol replacement levels.

Chapter 5

Statement-Proof Matching model

This Chapter introduces the structure of our matching model and goes through each part of the model with more details. In Section 5.1, we start with illustrating the overall structure of our model. Afterwards, Section 5.2 introduces the self-attention mechanism and transformer architecture, which are the crucial components of our BERT encoder. We also give more details about the NPT and BERT encoders we use in the experiments. Finally, we finish up with an introduction to the two decoding methods of the our model in Section 5.3.

5.1 Structure Overview

Figure 5.1 briefly illustrates the architecture of our matching model. The mathematical statement and proof pair will be respectively fed into the encoder (see Section 5.2), which transforms them into two vector representations. In the next step, the decoder (see Section 5.3) takes these two vectors and returns the matching score of the input pair.

5.2 Encoders

5.2.1 Preliminaries

5.2.1.1 Self-attention

In the sentence “I arrived at the bank after crossing the river”, the word “bank” has two different meanings: 1) an organization where people and businesses can invest or borrow money, change it to foreign money, etc., or a building where these services are

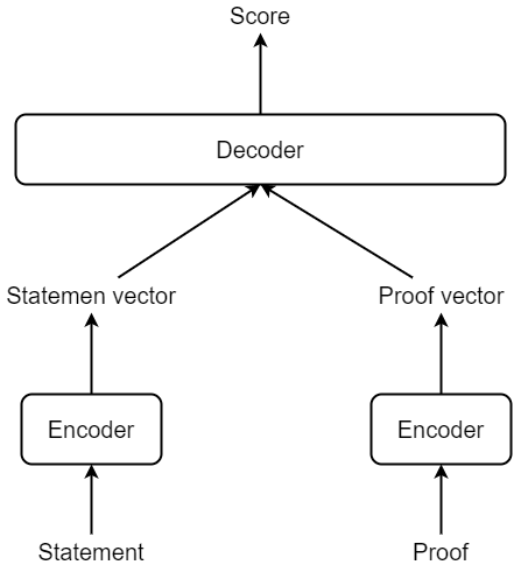

Figure 5.1: The structure overview of our matching model.

offered, 2) sloping raised land, especially along the sides of a river. While translating the sentence, to figure out which meaning "bank" refers to, the model needs to make decision based on the other words in the sentence. Recurrent neural networks (RNNs), a typical neural model architecture in machine translation, determine the word sense by reading each word between "bank" and "river" step by step. However, with the information accumulated in each time step, the more steps the decisions require, the harder it is for the RNNs to correctly choose the correct word sense. Vaswani et al. (2017a) proposed self-attention mechanism to solve the problem by directly connecting two tokens using one step of calculation. It no longer depends on time-dependent state like what RNNs require.

What is self-attention? Attention is simply a matrix showing the relativity of words in a sentence. In the general encoder-decoder structure, the source and target sentences are different. A typical example, in English-Chinese machine translation task, the source sentences are written in English and the target sentences are in Chinese. The word "self" in "self-attention" indicates that the self-attention mechanism focuses on the intra-relationship within a source or target sentence.

How does self-attention work? First, for a input sentence, the tokens in the sentence are embed to a branch of vectors `input#1`, `input#2` and `input#3`. For each input, we have three representations, query Q, key K and value V, shown in Figure 5.2. The

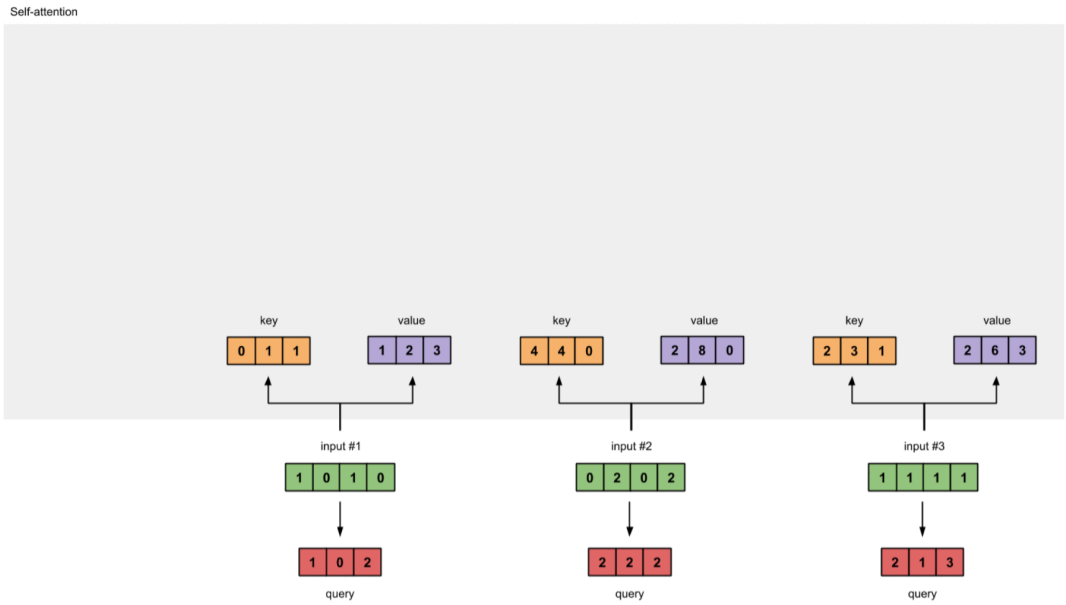

Figure 5.2: The three representations, query, key and value of an input sentence. Here we suppose every input has a dimension of 4 and the representations have a dimention of 3. Retrieved from https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

query, key and value are derived from the multiplication of the input and three sets of weights W^q, W^k and W^v, shown in Equation 5.1b to Equation 5.1c, where a^i is the i-th input. Figure 5.3 illustrates the following steps. The attention scores for `input#1` are calculated by taking the dot product between `input#1`'s query with the keys of other input tokens, which ends up with three attention scores. All the attention scores will pass through a softmax function and then multiplied with their corresponding value to get the alignment vectors (yellow). Eventually, `output#1` is obtained from the sum of these three alignment vectors.

$$Q^i = W^q a^i \tag{5.1a}$$

$$K^i = W^k a^i \tag{5.1b}$$

$$V^i = W^v a^i \tag{5.1c}$$

5.2.1.2 Transformers

Vaswani et al. (2017a) proposed the Transformer architecture that aims to solve sequence-to-sequence tasks when facing long-range dependency problem. It is based on the

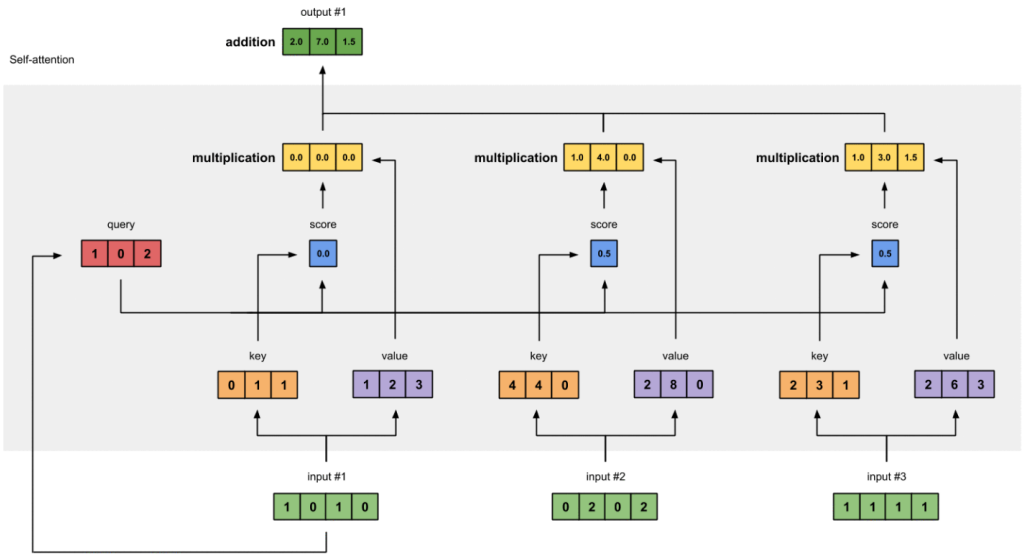

Figure 5.3: The illustration of the calculation of attention scores for Input 1. From Raimi Karim. Retrieved from https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

multi-head self-attention mechanism, which is actually doing the same thing as what we introduced in Section 5.2.1.1 with multiple heads or agents instead of doing it by one.

Multi-head Self-attention The multi-head self-attention makes the computation parallel and independent. Equation 5.2 gives the definition of multi-head attention (where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{\times d_k}$, $W_i^K \in \mathbb{R}^{\times d_k}$, $W_i^V \in \mathbb{R}^{\times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times}$). After the independent attention heads are computed in parallel, they will be concatenated and mapped to the output dimension through a linear layer.

$$\begin{aligned} \text{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) &= \text{Concat}(\text{head}_1,\ldots,\text{head}_h)\mathbf{W}^O \\ \text{where head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{5.2}$$

Encoder and Decoder Transformer consists of a pair of encoder-decoder stacks (illustrated in Figure 5.4). The encoder consists of 6 identical layers. Each of them contains two sub-layers: 1) a multi-head self-attention layer and, 2) a simple, positional fully connected feed-forward network. In each sub-layer, layer normalization and residual connection are applied. The decoder also consists of 6 identical layers,

consisting of an extra masked multi-head attention layer.

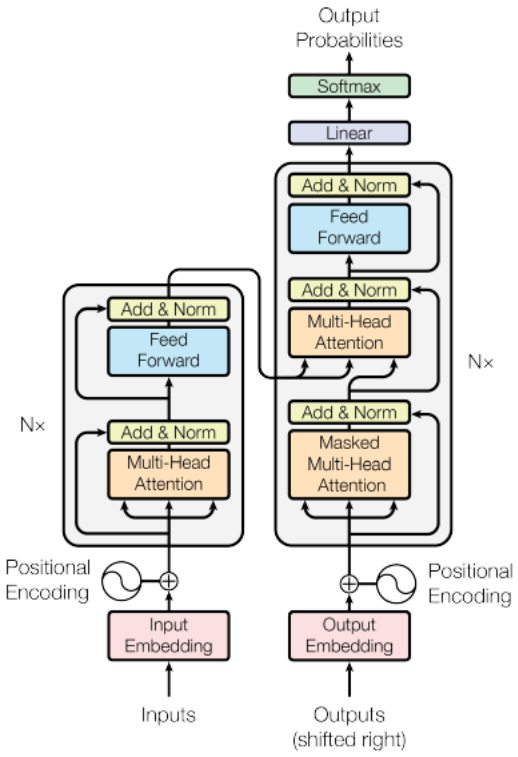

Figure 5.4: The Transformer - model architecture. From Vaswani et al. (2017a).

Positional Embeddings Since the transformer model does not contain recurrence and convolution. We need an alternative way to preserve the order of the sequence. Vaswani et al. (2017a) introduced the "positional encodings" to the input embeddings at the beginning of the encoder and decoder stacks. These encodings follow a specific periodic function as shown in Equation 5.3 and 5.4.

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}}) \tag{5.3}$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}}) \tag{5.4}$$

5.2.2 NPT encoder

The "no pre-training" encoder (NPT) is built on the self-attention mechanism (Vaswani et al., 2017b), which originally proposed as the self-attention layer of Transformers (as introduced in Section 5.2.1.1).

With the self-attentive encoder we describe above, the sequence of tokens can be encoded to a a sequence of fixed-size vector representations for statements and proofs. In particular, we run l self-attention layers to get the contextualized embedding for each input token. The final vector representation for the text will be constructed by a max-pooling layer over the embeddings obtained from the last self-attention layer.

5.2.3 BERT encoder

5.2.3.1 BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers, is basically a stack of Transformer encoder layers with multiple self-attention heads. It uses the attention mechanism to learn the contextual relations in the text. Opposing to the previous pretrained language models, BERT is the first model which introduces the concept of "bidirectional". It no longer strictly process the text from left to right, but jointly consider both left and right context in all layers. This setup was proved to be more powerful than the unidirectional language models (Devlin et al., 2018).

The complete version of BERT contains both an encoder for reading the input sentences and a decoder to offer the prediction for the task. In our model, we have a separate trainable decoder (see Section 5.3), so only the encoder is needed.

As it is illustrated in Figure 5.5, the input embeddings of BERT is the sum of token embeddings, sentence embeddings and positional embeddings. The input embeddings will be passed forward and updated using two tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP).

- **MLM:** Before the input text sequences are fed into the encoder,15% of the tokens are "masked out". 80% of the selected tokens will be replaced with the `[MASK]` token. 10% of them are replaced with random tokens and the last 10% are left unchanged. BERT will try to predict the masked tokens based on the provided context (i.e. the non-masked tokens).

- **NSP:** The Next Sentence Prediction task was designed to understand the relationship a pair of sentences. Given two sentences, the model learns to predict if the second sentence is the subsequent sentence of the other sentence in the document. However, Liu et al. (2019) suggested that removing the NSP loss slightly improved the performance of downstream task. Therefore, some researchers chose not to get NSP involved while training BERT model.

A survey concludes that BERT has become a ubiquitous baseline after investigating over 150 studies of the popular BERT model (Rogers et al., 2020). One of the most significant advantages of BERT is that the contextual embeddings generated from BERT form distinct clusters to word senses, which solves the issue of the first generation PLMs (Wiedemann et al., 2019; Schmidt and Hofmann, 2020).

5.2.3.2 Domain Adaptation of BERT

In Section 2.2, we introduced the two-stage finetuning in Figure 2.1. Researchers started to use this pipeline to adapt BERT to various domains including mathematics (Shen et al., 2021, MATHBERT), clinical text (Alsentzer et al., 2019, CLINICALBERT), biomedical text (Lee et al., 2019, BIOBERT), financial text (Araci, 2019, FINBERT), and scientific text (Beltagy et al., 2019, SCIBERT). All of these domain adapted version of BERT outperformed the out-of-box BERT trained on general domain corpora on the tasks in their specific domains.

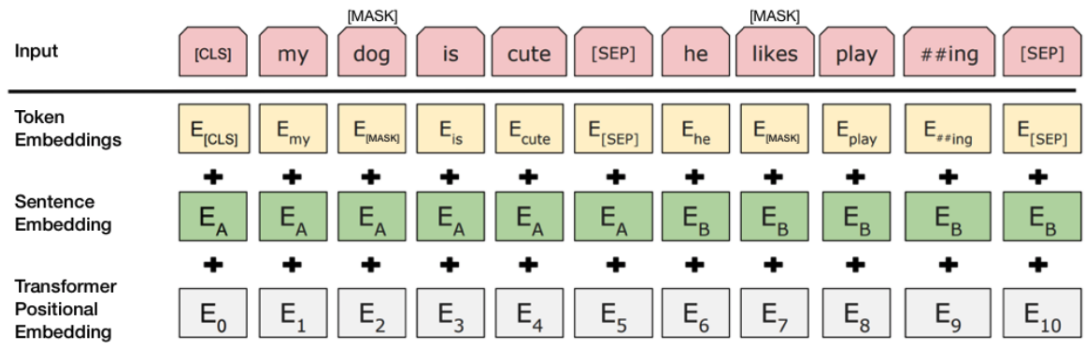

Figure 5.5: BERT input representation (Devlin et al., 2018).

5.3 Decoder

Trainable bilinear similarity function Given the encoded representations of a statement $\mathbf{s} = \text{enc}(s)$ and a proof $\mathbf{p} = \text{enc}(p)$, we compute an association score with the following bilinear form:

$$\text{score}(\mathbf{s}, \mathbf{p}) = \mathbf{s}^\top \cdot \mathbf{W} \cdot \mathbf{p} + b,$$

where $\mathbf{W}$ and b are parameters that are learned together with a self-attentive encoder or a pretrained BERT encoder parameters (Section 5.2).

Local decoding For a collection of n statements and proofs, we first score all possible pairs (s, p), and construct a matrix $M = (m_{ij}) \in \mathbb{R}^{n \times n}$, with

$$m_{ij} = \text{score}(\mathbf{s}^{(i)}, \mathbf{p}^{(j)}),$$

where $\mathbf{s}^{(i)}$ and $\mathbf{p}^{(j)}$ are the encoded representations of, respectively, the i^{th} statement and the j^{th} proof. Then we can straightforwardly sort each row by decreasing order and assign the proof ranking to the corresponding statement. The best ranking proof $\hat{p}$ for statement i satisfies $\hat{p}_i = \arg\max_j m_{ij}$. We call this decoding method 'local', since it does not take into account dependencies between assignments. In particular, several statements may have the same highest-ranking proof.

Global decoding The local decoding method overlooks a crucial piece of information: a proof should correspond to a single statement. In a worst-case situation, a small number of proofs may score high with most statements and be systematically assigned as highest-ranking proof by the local decoding method.

In preliminary experiments, we analysed the output of our system with local decoding on the development set, focusing on the distribution of the single highest-ranking proof for each statement. We found that 23% of the proofs were assigned to at least two different statements, whereas more than 40% of proofs were assigned to no statement. See also Table 3.1.

We propose a second decoding method based on a global constraint on the output: a proof can be assigned only to a single statement. Intuitively, the constraint models the fact that if a proof is assigned by the system to a certain statement with high confidence, we can rule it out as a candidate for other statements. Under this constraint, the decoding problem reduces to a classical maximum weighted bipartite matching problem, or equivalently, a Linear Assignment Problem (LAP). In more realistic scenarios (e.g. if the input sets of statements and proofs do not have the same size), the method would require some adaptation.

Formally, we define an assignment A as a boolean matrix $A = (a_{ij}) \in \{0,1\}^{n \times n}$ with the following constraints:

$$\forall i \forall j, \sum_j a_{ij} = \sum_i a_{ij} = 1,$$

i.e. each row and each column of A contains a single non-zero coefficient. The score of an assignment A is the sum of scores of the chosen edges:

$$\text{score}(A, M) = \sum_i \sum_j a_{ij} m_{ij}.$$

Finally, global decoding consists in solving the following LAP:

$$\hat{A}(M) = \underset{\substack{A \in \{0,1\}^{n \times n} \\ \text{s.t. } \forall i \forall j, \sum_j a_{ij} = \sum_i a_{ij} = 1}}{\arg\max} \text{score}(A, M).$$

The LAP is solved in polynomial time by the Hungarian algorithm Kuhn (1955), the LAP-Jonker-Volgenant algorithm (LAP-JV; Jonker and Volgenant, 1987), or the push-relabel algorithm Goldberg and Kennedy (1995). These methods have a $O(n^3)$ time complexity where n is the number of pairs, and $O(n^2)$ memory complexity. This is too expensive in our case, due to our dataset size.

To remedy this limitation, when we perform decoding on a large set, we only consider the k best-scoring proofs (i.e. outgoing edges in the bipartite graph) for each statement, which makes the number of edges linear in the number of pairs n (considering k fixed). Moreover, we use a modification of the LAP-JV algorithm specifically designed for sparse matrices (LAP-MOD; Volgenant, 1996).

Chapter 6

Local and Global Training

In this chapter, we introduce two training methods for the similarity model: a local training method that only considers statements in isolation (see Section 6.1) and a global model trained to predict a bipartite matching (see Section 6.2), with a hybrid global-local objective.

6.1 Local Training

We would like to train our model to assign a high similarity to the gold statement-proof pair, and a low similarity to all other statment-proof pairs. This corresponds to the following objective, for a single statement s and its gold proof p:

$$\begin{aligned}\mathcal{L}_{\text{LOC}}(s,p,P;\theta) &= -\log \mathbb{P}(p|s;\theta) \\ &= -\log\left(\frac{\exp(\text{score}(\mathbf{s},\mathbf{p}))}{\sum_{p'\in P}\exp(\text{score}(\mathbf{s},\mathbf{p}'))}\right),\end{aligned}$$

where P is the set of proofs, and θ are the parameters of the model. Directly optimizing this loss function requires the computation of $\mathbf{p} = \text{enc}(p)$ for every proof in the dataset, for a single optimization step. This is not realistic considering memory limitations, the size of the train set, and the fact that our self-attentive encoder is the most computationally expensive part of the network.

Instead, we sample minibatches of b pairs and optimize the following proxy loss for the sequence $S' = (s_1,\ldots,s_b)$ of statements and the sequence $P' = (p_1,\ldots,p_b)$ of

corresponding proofs:[1]

$$\mathcal{L}'_{\text{LOC}}(S', P'; \theta) = \sum_{i=1}^{b} \mathcal{L}_{\text{LOC}}(s_i, p_i, P'; \theta).$$

In practice, we sample uniformly and without replacement b pairs from the training set at each stochastic step.

6.2 Global Training

The local training method only considers statements in isolation. Even though we expect a locally trained model to perform better with global decoding, we hypothesize that a model that is trained to predict the full structure (a bipartite matching) will be even better.

For a collection of n proofs and n statements, the size of the search space (i.e. the number of bipartite matchings) is $n!$, since each matching corresponds to a permutation of proofs. As a result, the use of a globally normalized model is impractical. We turn to a max-margin model that does not require normalization over the full search space.

We use the following max-margin objective, for a set B of n pairs corresponding to matrix M:

$$\begin{aligned}\mathcal{L}_{\text{GLOBAL}}(B; \theta) = &\max(0, \Delta(\hat{A}, I) \\ &+ \text{score}(\hat{A}, M) - \text{score}(I, M)),\end{aligned}$$

where θ is the set of all parameters $\hat{A}$ is the predicted assignment and I is the gold assignment, i.e. the identity matrix. The structured cost

$$\Delta(\hat{A}, I) = \sum_{ij} \max(0, (\hat{A} - I)_{ij})$$

aims at enforcing a margin for each individual assignment. $M' = M + (\mathbf{1} - \mathbf{I})$.

The computation of this loss requires exact decoding for each optimization step. Since exact decoding is only feasible for a small n, and since we need to keep track of all intermediary vectors to compute the backpropagation step,[2] we perform each stochastic optimization step on a minibatch of pairs of size b. Since this global objective had a slow convergence rate, in practice, we use a hybrid local-global objective: $\mathcal{L}'_{\text{LOC}} + \mathcal{L}_{\text{GLOB}}$.

[1] We also experimented with a Noise-Contrastive Estimation approach Gutmann and Hyvärinen (2012). However, it exhibited a much slower convergence rate.

[2] In particular, the computation graph needs to conserve all encoding layers for the $2n$ texts involved.

Chapter 7

Experiments and Results

In this chapter, Section 7.1 introduces how we set up the experiments. Section 7.2 presents the results and discusses the patterns we have observed. At last, Section 7.3 illustrates the qualitative analysis using Local Interpretable Model-Agnostic Explanations (LIME), to offers some "insight" on how the BERT model makes decision on our task.

7.1 Experimental Setup

Dataset We use the dataset whose construction is described in Section 3. We shuffle the collection of statement-proof pairs before performing a 80%/10%/10% train-development-test split, corresponding to 147278 pairs for the training sets and 18408 pairs for the development and tests.

Encoders We experiment with several encoders to obtain neural representations of the theorem and proof pairs. Our first encoder is a simple self-attentive encoder. We use $\ell = 2$ self-attentive layers with 4 heads to obtain contextualized embeddings of dimension $d = 300$. The query and key vectors have size $d_k = 128$. We construct a vector representation for the text with a max-pooling layer over the contextualized embeddings of the last self-attention layer. We do not use any form of pre-training for this encoder and hence name it "no pre-training encoder" (NPT). In addition, we experiment with a BERT model Devlin et al. (2019) as an encoder. We do not use the pre-trained version provided by Devlin et al., but rather pre-train the base version from scratch (SCRATCHBERT), but we do compare our results against a math-tailored pre-trained version of BERT (Peng et al. 2021; see below). Both the NPT and

SCRATCHBERT vocabularies are customized for our dataset, as preliminary experiments revealed the importance of the model-task vocabulary match.[1]

To further demonstrate how crucial this vocabulary match is, we experiment with math-BERT (MATHBERT; Shen et al. 2021), a state-of-the-art pre-trained model for mathematical formula understanding. This model is pre-trained on a large mathematical corpus ranging from pre-kindergarten, to high-school, to college graduate level mathematical content, including professional mathematical papers, using the BERT masked language modeling (MLM) task. We use the pre-trained version provided by the authors, *without vocabulary customization*. All of our encoders are fine-tuned on the matching task. In addition, we experiment with a naive token-matching system that computes cosine similarities between TF-IDF representations of statements and proofs. We discovered that their performance was very low, ranging from 11.4 to 29.8 (MRR), so we did not experiment with them further.

Hyperparameters For pretraining SCRATCHBERT, we first train a new word piece tokenizer[2]. Next, we train the SCRATCHBERT model on the MLM task for 60 epochs (around 3 days) using four NVIDIA v100 GPUs. We evaluate the language model every 500 steps, where one step stands for training on one sentence example, and choose the one with the best performance on the validation set.

We perform local and global training / finetuning respectively for the NPT model, MATHBERT, and SCRATCHBERT. NPT has 15M parameters while MATHBERT and SCRATCHBERT have 110M parameters. We observed in initial experiments that training only with the global objective required a long time to converge. Therefore, we used the following global-local objective: $\mathcal{L}'_{\text{LOC}} + \mathcal{L}_{\text{GLOB}}$, that we optimized by alternating one stochastic step for each loss.

We train the NPT model for 400 epochs (around 1 day with two GPUs) over the whole training set for local and global training. We use batches of size $b = 60$ and set learning rate $l = 5 \times 10^{-3}$ with the Averaged Stochastic Gradient Descent (ASGD; Polyak and Juditsky 1992) optimizer. We use an exponential learning rate scheduler (the learning rate multiplied by 0.996 after each epoch) to stabilize the optimizer in the latter training procedure (after 300 epochs). We evaluate the performance of the model on the validation set every 20 epochs during training and select the best one among these intermediate models.

[1] This supports the findings of chalkidis-etal-2020-legal, for example, in a different domain.

[2] https://huggingface.co/docs/transformers/tokenizer_summary#wordpiece

We use four NVIDIA v100 GPUs to fine-tune MATHBERT and SCRATCHBERT on the training set for 60 epochs (around 2 days) with a learning rate of $l = 2 \times 10^{-3}$, an ASGD optimizer, batches of size $b = 16$, and a scheduler that multiplies the learning rate by 0.99 after each epoch. We choose the best model on the validation set, evaluating the models every five epochs.

Global decoding Recall that exact global decoding is only feasible for a small subset of pairs. During global training, we chose a batch size small enough to perform exact decoding. However, it is not feasible to perform exact decoding on the whole development and test corpora. Therefore, we prune the search space by keeping only the 500-best candidate proofs for each statement, and use the LAP-MOD algorithm designed for sparse matrices. In practice, we used the implementations of the LAP-JV and LAP-MOD algorithms from the `lap` Python package,[3] for respectively exact decoding on minibatches during global training and decoding on whole datasets during evaluation.

7.2 Results

Our experiments address several questions. First, we assess the task's difficulty under different replacement levels using different encoders and schemes (global or local training, global or local decoding). In particular, we are interested in assessing whether global decoding improves accuracy when training is only local, and how the more complex global training method fares with respect to local training. We then measure the informativeness of different types of input: text, mathematical formulae, or both. The comparison of these settings is meant to provide insight into which type of information is crucial to the task. Finally, we experiment with a cross replacement levels setup, i.e., when a model is tested on a different symbol replacement level from the one that was used during training. We hope this experiment will shed some light on the importance of training models on real-world datasets.

7.2.1 Main Results

Table 7.1 presents our results. We report MRR (if relevant) and accuracy scores across different levels of symbol replacement.

[3] https://github.com/gatagat/lap

	Symbol Replacement Level							
	Conservation		Partial		Full		Transposition	
Encoder-Decoder	MRR	Acc	MRR	Acc	MRR	Acc	MRR	Acc
NPT-Local-Local	63.22	56.08	47.19	39.24	40.36	32.52	56.17	48.30
NPT-Local-Global	-	61.89	-	42.55	-	35.43	-	53.49
NPT-Global-Global	-	62.14	-	43.68	-	35.85	-	55.28
SCRATCHBERT-Local-Local	**73.73**	67.12	**64.79**	57.20	**60.67**	52.54	**73.17**	66.51
SCRATCHBERT-Local-Global	-	**74.68**	-	**62.80**	-	**57.69**	-	**74.03**
SCRATCHBERT-Global-Global	-	71.38	-	58.06	-	52.31	-	70.32
MATHBERT-Local-Local	54.51	46.45	44.31	36.10	38.91	30.62	52.57	44.52
MATHBERT-Local-Global	-	49.77	-	37.92	-	32.03	-	47.43
MATHBERT-Global-Global	-	45.38	-	33.64	-	28.47	-	43.41

Table 7.1: The MRR and accuracy scores for different combinations of encoders, decoders, and symbol replacement levels. All the models are trained and tested on the same replacement level. Best result in each column is bolded. Following the model name, we include its encoder and decoder type (both being either Local or Global).

Encoders While MATHBERT is pre-trained on millions of examples curated from mathematical contents, it performs worse than the less complex NPT encoder, which is trained solely on the downstream task across all symbol replacement levels and decoders.[4] SCRATCHBERT, which shares MATHBERT architecture and NPT customized vocabulary, is outperforming both consistently. These results demonstrate the vocabulary importance for learning from mathematical texts.

Symbol Replacement Levels Difficulty We obtain the best scores across all of our models is the Conservation level, as the models can match identical symbol names across theorem-proof pairs. The models achieve similar performance with Transposition replacement. These results suggest that the symbols' order, context, and function within the mathematical text do not play a significant role when the theorem and proof share the same symbols' names. In contrast, when the symbol names are changed (Partial and Full replacements), we observe a sharp decline in results.

Training and Decoding Effects In all settings, global decoding substantially improves accuracy and MRR. These improvements are more noticeable for the top two encoders, NPT and SCRATCHBERT. For NPT, we observe better performance when

[4]We observe similar trends when fine-tuning the out-of-the-box BERT model on the matching task.

using global training, but not for SCRATCHBERT and MATHBERT. Due to the lack of computational resources, we can not reach the global training full potential when using highly expressive encoders such as SCRATCHBERT and MATHBERT, which share BERT-base architecture (see Section 8.2 for more details).

7.2.2 Effect of Input Type Analysis

To better understand the importance of each input type, we examine SCRATCHBERT-Local-Local and NPT-Local-Local performance when fed with text, mathematical formulae, or both (Table 7.2). We test them on the Conservation and Full symbol replacement levels. The mathematical formulae input plays a more significant role for both models than the textual input. When trained and tested on the Conservation replacement level, NPT-Local-Local makes better use of the mathematical formulae input than the more expressive, pre-trained SCRATCHBERT-Local-Local. When trained and tested on with Full replacement, where the models cannot rely on simple token-matching, NPT-Local-Local suffers from a sharper performance decline than SCRATCHBERT-Local-Local when fed with mathematical formulae input. These results suggest that when applied to the Conservation data, a less expressive model can get high results by harnessing simple token matching. SCRATCHBERT-Local-Local performs better for both replacement levels when fed with text and complete input.

	Symbol Replacement			
	Conservation		Full	
Input	MRR	Acc	MRR	Acc
	NPT			
Text	22.51	16.68	22.51	16.68
Math	65.08	58.47	34.55	27.30
Both	63.22	56.08	40.36	32.52
	SCRATCHBERT			
Text	36.85	29.18	36.85	29.18
Math	63.10	55.92	41.64	34.01
Both	**73.73**	**67.12**	**60.67**	**52.54**

Table 7.2: SCRATCHBERT-Local-Local and NPT-Local-Local performance for different input types. Both stands for the original and complete input.

7.2.3 Cross Replacement Setup

We examine the effect of testing a model on a different symbol replacement level from the one it was trained on (Table 7.3). We use the SCRATCHBERT-Local-Local model for all of our experiments. We observed a sharp decline in results when SCRATCHBERT-Local-Local is trained on the Conservation replacement level and tested on Partial or Full. These drops in performance suggest the model developed a strong dependency on exact symbol names matching. In addition, the replacement shift from Conservation to Transposition and vice versa resulted in a minor performance drop. These results provide additional evidence for the lack of importance of mathematical functionality, order, and context of symbols' names shared across theorem and proof pairs.

The model trained on the Partial symbol replacement level demonstrated a significant resilience when tested with other symbol replacement levels. It outperforms the rest of the models when applied to out-of-domain replacement levels and the Conservation replacement level in-domain model.

Source \ Target	Symbol Replacement							
	Conservation		Partial		Full		Transposition	
	MRR	Acc	MRR	Acc	MRR	Acc	MRR	Acc
Conservation	73.73	67.12	43.87	36.36	29.74	25.36	69.56	62.23
Partial	**74.21**	**67.96**	**64.79**	**57.20**	53.77	45.40	72.13	65.42
Full	65.26	57.63	63.01	55.13	**60.67**	**52.54**	64.59	56.92
Transposition	73.78	67.40	43.67	36.02	29.76	25.47	**73.17**	**66.51**

Table 7.3: Cross-replacement levels performance for the SCRATCHBERT-Local-Local model.

7.3 Qualitative analysis: LIME

Deep learning models have shown their advantages due to their higher complexity. However, machine learning models are black boxes in most cases. People have no insight of what is happening behind a model but are asked to trust the results. Although the accuracy on cross validation can somehow validate the ability of the model, real-world data sometimes can be significantly different, which makes the evaluation metric not indicative of the product's goal (Ribeiro et al., 2016). Therefore, it is sig-

nificant to explore the explanations behind the predictions provided by the models, as a worthwhile solution in addition to such metrics. Ribeiro et al. (2016) proposed the Local Interpretable Model-Agnostic Explanations (LIME), a novel explanation technique which can interpret the predictions of any classifier. The name of the technique reflects what we desire in the explanations:

Local Local refers to local fidelity. It is hard to guarantee that an explanation is completely faithful unless it is exactly the description of the model. Therefore, we expect the explanation to at least reflect the behaviour of the classifier on the instances being predicted, which is called "locally faithful". The author admitted that carrying out a globally faithful explanations still remains a challenge.

Interpretability The representations in modern machine learning models such as word embeddings are not intuitive to humans. LIME trys to explain the models on the interpretable level such as text so that humans can understand.

Model-agnostic To make it be able to apply on all the models, LIME does not go deeply into the model. Instead, LIME tweaks the feature values and observes the influence on the output brought by the modifications. For example, if an input sentence is "Math is so interesting", LIME will perturb the input and get new predictions on the modified sentences such as "Math so interesting", "Math so interesting", "Math so", etc. If the sentence is encoded as a sequence of word embeddings, LIME can also perturb the word embeddings to get the same effect.

To study which tokens affect our model predictions, we use LIME to examine SCRATCHBERT-Local-Local trained with the Conservation setup, and with Full replacement. Both are applied to original test examples. We observe that the Conservation SCRATCHBERT-Local-Local model heavily relies on the mathematical tokens and barely benefits from the text ones. In contrast, the SCRATCHBERT-Local-Local model that was trained in the full symbol replacement setup strongly relies on textual tokens with mathematical meaning, such as "module", "supplement", and "semistable", to name a few. A visualization is given in Figure 7.1.

Lemma 3.2. Let M be a module and H a local submodule of M. Then H is a supplement of each proper submodule $K \leq M$ with $H + K = M$.

Proof. Since K is a proper submodule of M and $K + H = M$, we have $K \cap H$ is a proper submodule of H. Therefore $K \cap H \ll H$, since H is local. That is, H is a supplement of K in M.

(https://arxiv.org/pdf/0810.0041.pdf)

(a) Example statement/proof 1 - Symbol conservation

Lemma 3.2. Let M be a module and H a local submodule of M. Then H is a supplement of each proper submodule $K \leq M$ with $H + K = M$.

Proof. Since K is a proper submodule of M and $K + H = M$, we have $K \cap H$ is a proper submodule of H. Therefore $K \cap H \ll H$, since H is local. That is, H is a supplement of K in M.

(https://arxiv.org/pdf/0810.0041.pdf)

(b) Example statement/proof 1 - Full symbol replacement

Lemma 4.1.9. If F is a μ-semistable X-twisted sheaf of rank r then dim Hom (F , F) $\leq r^2$.

Proof. Any endomorphism of F must preserve the socle (see Lemma 1.5.5ff of [4]); moreover, the quotient F / Soc(F) is also semistable. The result follows by induction from the polystable case, which itself follows immediately from the fact that stable sheaves are simple.

(https://arxiv.org/pdf/0803.3332.pdf)

(c) Example statement/proof 2 - Symbol conservation

Lemma 4.1.9. If F is a μ-semistable X-twisted sheaf of rank r then dim Hom (F , F) $\leq r^2$.

Proof. Any endomorphism of F must preserve the socle (see Lemma 1.5.5ff of [4]); moreover, the quotient F / Soc(F) is also semistable. The result follows by induction from the polystable case, which itself follows immediately from the fact that stable sheaves are simple.

(https://arxiv.org/pdf/0803.3332.pdf)

(d) Example statement/proof 2 - Full symbol replacement

Figure 7.1: LIME visualizations for the model that was trained in the symbol conservation setup (a and c) and their corresponding LIME visualizations for the model that was trained in the full symbol replacement setup (b and d). The LIME "match" class supporting features are colored in orange, and the "mismatch" is in blue. The darker the color, the higher (in absolute value) the feature importance.

Chapter 8

Conclusions and Future Work

This Chapter gives a summary of our findings in Section 8.1. Section 8.2 and Section 8.3 introduce the limitations we have observed in our work and the potential directions to continue on this research topic.

8.1 Conclusion

We developed a bilinear similarity model and a large dataset (MATcH) for a task focusing on the domain of mathematical research articles. The task consists in matching a proof to a mathematical statement. We proposed two ways to train and do inference with our model and dataset: local matching and global matching. We assessed the difficulty of the task with several pre-trained encoders, demonstrating the importance of the vocabulary support for these models. Further assessment relies on the use of a symbol replacement procedure, which helps test the type of mathematical reasoning the encoders can perform. While our model performs well on this task, we observe through the symbol replacement procedure that the model makes a relatively shallow use of the text and formulae to obtain this performance. Our experiments show that both mathematical formulae and the text surrounding them is an important source for making accurate predictions. Finally, we have introduced a global neural model for addressing the structured prediction problem of maximum weighted bipartite matching. The model is based on a self-attentive encoder and a bilinear similarity function. Our experiments show that bag-of-words baselines are insufficient to solve the task, and are outperformed by our proposed model by a wide margin. We found that decoding is crucial to achieve high results, and is further enhanced by a global training loss. Finally, our results show that mathematical formulae are the most informative source

of information for the task but are best taken into account with the self-attentive neural model.

8.2 Limitations

Our work has two main limitations. First, we aim to simulate a setup where the same author did not write both a theorem and its corresponding proof. We reduce the symbols' names intersection between the statement and the proof, which leads to more challenging setups. In real-world scenarios, authors differ not only in their notation choices but also in their writing styles. In contrast to notations, writing styles can not be altered using simple rule-based methods. Therefore, additional effort is needed to obtain a truly real-world setup for mathematical information retrieval. Second, due to computational limitations, we can not reach the full potential of our global training method. More precisely, our GPUs can not facilitate large batch sizes for large models such as MATHBERT and SCRATCHBERT. We use NVIDIA v100 GPUs that allow us to experiment with a batch size of 16 for MATHBERT and SCRATCHBERT, compared to 60 with NPT.

In addition, while our symbol replacement method provides a coarse way to test the language model use of the symbols and text in mathematical articles, it presents cases in which the replacement is not precise. These cases arise because the use of symbols in mathematical language is rich and context-dependent (for example, while π often refers to the constant pie, it might also refer to a tuple-projection function).

8.3 Future Work

Successfully applying natural language processing (NLP) methods to mathematical texts is a highly challenging task due to their unique jargon, notations, and complex nature. We have shown that the pretrained language models rely much on the overlapping mathematical symbols instead of focusing on the semantic information behind the mathematical text.

There are two types of experiments to further test with. Firstly, it is worthwhile to carry out experiments on other existing statement-proof matching datasets such as NaturalProof (Welleck et al., 2021), for validating the naive matching pattern of PLMs. As we mentioned before, the current global training results of the SCRATCHBERT model can only use a batch size with 16 due to the limitation of computing resources

and it does not show the advantage of global training with such a small batch size. We can further use global training with batch size of 60 to validate the power of our global method.

The future improvement is three-fold. Since the statement-proof pairs are extracted from professional mathematical papers, we notice that pairs that come from the same paper have strong references. In the example we show in Figure 1.4, the proof refers to "Theorem 1.1". Such references contain no meaning when separated out from the context. Therefore, we can consider either further filtering our dataset or trying to ease the problem from the model level. Secondly, some mathematical symbols contain special meanings and people have some preferences while choosing the symbols. For example, E sometimes stands for "energy" in physics. i and j are more likely to be used as subscript to represent the index. Some mathematical symbols can even represent constants in a specific domain. For example, g usually stands for the gravity constants in physical math. We can consider more complicated rules while doing the symbol replacement or even set several "unchanged list" for each domain to store the symbols that are not going to be replaced in that specific area. One more possible improvement is applying the contrastive learning (CL) technique. Contrastive learning was proposed to learn effective representation by making the similar data points closer and pushing apart the non-neighbours (Hadsell et al., 2006). Chen et al. (2020) built a self-supervised contrastive learning framework called SimCLR to learn the visual representations. Later, Gao et al. (2021) aggregated the framework to learn sentence embeddings. They used their SimCSE framework to further train the $\text{BERT}_{\texttt{base}}$ model and achieved substantial improvements on several semantic textual similarity benchmarks. In contrastive learning, the strategy to generate positive and negative samples is the crucial part (Arora et al., 2019). In our case, we already have the symbol replacement method which has no harmness to the semantic information. Therefore, it can act as an ideal sampling technique in contrastive learning. Furthermore, we can let the model learn the differences among the multiple domains we have in the dataset and also manipulate hard-negative proofs that share the same mathematical symbols as the statement. If the model achieves better performance after training on these setups, it means contrastive learning can successfully enhance the PLMs' ability to encode the semantic information in mathematical text.

Finally, some researchers have pointed out that BERT showed its weakness on encoding the numeracy in the text (Wallace et al., 2019) and sequential order of words was not significant in natural language understanding tasks (Pham et al., 2021). Adapt-

ing PLMs to mathematical domain remains challenges since number and symbols' order matter a lot in the formulae and a tiny modification can completely change the meaning. Specifically, for our task, the ignorance of value and tokens order might also be one of the reasons for resulting in naive symbol matching we showed in the previous experiments. There are a lot of things to explore regarding mathematical language understanding. By introducing our work, we hope it can inspire more researchers to work on this field and explore further on applying natural language processing techniques to mathematical domain.

Appendix A

Complete Result Tables

Table A.1, Table A.2 and Table A.3 show the complete results of all the experiments we have run.

Encoder	Method	Mode	Replacement	Conservation test mrr	Conservation test acc	Partial test mrr	Partial test acc	Transposition test mrr	Transposition test acc	Full test mrr	Full test acc
NPT	local+local	both	conservation	63.22	56.08	33.74	27.35	40.48	33.58	23.23	19.81
NPT	local+local	text	conservation	22.51	16.68	-	-	-	-	-	-
NPT	local+local	math	conservation	65.08	58.47	33.05	27.23	40.55	34.19	23.15	20.26
NPT	local+local	both	partial	61.37	54.12	47.19	39.24	47.06	39.34	34.59	27.69
NPT	local+local	math	partial	59.85	52.95	43.77	36.14	43.81	36.61	31.19	25.09
NPT	local+local	both	transposition	60.25	52.82	34.56	27.58	56.17	48.30	24.04	19.81
NPT	local+local	math	transposition	61.51	54.62	33.77	27.50	57.03	49.34	23.28	19.80
NPT	local+local	both	full	47.87	40.13	43.69	35.86	43.78	35.94	40.36	32.52
NPT	local+local	math	full	45.29	37.81	39.08	31.71	38.47	30.96	34.55	27.30
MathBERT	local+local	both	conservation	54.51	46.45	25.44	19.87	49.76	41.41	19.56	16.32
MathBERT	local+local	text	conservation	22.60	16.09	-	-	-	-	-	-
MathBERT	local+local	math	conservation	51.33	43.52	22.73	17.82	47.76	39.77	17.95	14.99
MathBERT	local+local	both	partial	55.79	48.03	44.31	36.10	53.64	45.69	34.52	26.89
MathBERT	local+local	math	partial	49.93	42.62	34.80	27.50	47.48	39.84	25.40	19.64
MathBERT	local+local	both	transposition	54.46	46.42	26.30	20.49	52.57	44.52	19.76	16.25
MathBERT	local+local	math	transposition	51.90	44.14	22.98	17.88	50.06	42.13	18.12	15.06
MathBERT	local+local	both	full	42.36	34.02	40.56	32.25	41.94	33.70	38.91	30.62
MathBERT	local+local	math	full	34.28	27.10	30.89	24.09	33.66	26.62	27.98	21.20
ScratchBERT	local+local	both	conservation	73.73	67.12	43.87	36.36	69.56	62.23	29.74	25.36
ScratchBERT	local+local	text	conservation	36.85	29.18	-	-	-	-	-	-
ScratchBERT	local+local	math	conservation	63.10	55.92	32.60	26.09	59.15	51.42	23.05	19.57
ScratchBERT	local+local	both	partial	74.21	67.96	64.79	57.20	72.13	65.42	53.77	45.40
ScratchBERT	local+local	math	partial	61.71	54.70	47.85	40.07	58.43	51.09	36.23	29.41
ScratchBERT	local+local	both	transposition	73.78	67.40	43.67	36.02	73.17	66.51	29.76	25.47
ScratchBERT	local+local	math	transposition	63.39	56.42	32.65	26.43	62.46	55.13	23.42	20.05
ScratchBERT	local+local	both	full	65.26	57.63	63.01	55.13	64.59	56.92	60.67	52.54
ScratchBERT	local+local	math	full	49.50	42.13	44.99	37.41	47.52	40.05	41.64	34.01

Table A.1: Complete result table using local training and local decoding.

Encoder	Method	Mode	Replacement	Conservation test acc	Partial test acc	Transposition test acc	Full test acc
NPT	local+global	both	conservation	61.89	29.30	36.14	19.88
NPT	local+global	text	conservation	17.67	-	-	-
NPT	local+global	math	conservation	64.11	28.16	36.21	20.01
NPT	local+global	both	partial	59.38	42.55	42.81	29.66
NPT	local+global	math	partial	58.13	39.52	39.86	26.93
NPT	local+global	both	transposition	58.52	30.07	53.49	20.11
NPT	local+global	math	transposition	59.58	29.26	54.86	19.95
NPT	local+global	both	full	43.86	39.24	39.45	35.43
NPT	local+global	math	full	41.88	34.63	34.27	30.25
MathBERT	local+global	both	conservation	49.77	19.94	44.70	15.46
MathBERT	local+global	text	conservation	15.90	-	-	-
MathBERT	local+global	math	conservation	47.09	17.92	42.60	14.25
MathBERT	local+global	both	partial	51.06	37.92	48.61	28.14
MathBERT	local+global	math	partial	44.78	28.18	41.79	19.91
MathBERT	local+global	both	transposition	49.77	20.28	47.43	15.42
MathBERT	local+global	math	transposition	47.35	18.29	45.59	14.43
MathBERT	local+global	both	full	35.83	33.74	35.54	32.03
MathBERT	local+global	math	full	28.35	24.65	27.89	21.82
ScratchBERT	local+global	both	conservation	74.68	40.60	70.46	26.91
ScratchBERT	local+global	text	conservation	31.25	-	-	-
ScratchBERT	local+global	math	conservation	60.93	29.05	57.28	20.22
ScratchBERT	local+global	both	partial	74.49	62.80	72.00	50.45
ScratchBERT	local+global	math	partial	59.37	43.52	55.56	31.10
ScratchBERT	local+global	both	transposition	74.23	40.35	74.03	26.38
ScratchBERT	local+global	math	transposition	60.84	28.78	60.39	20.43
ScratchBERT	local+global	both	full	63.22	60.37	62.14	57.69
ScratchBERT	local+global	math	full	46.03	40.93	43.70	36.81

Table A.2: Complete result table using local training and global decoding.

Encoder	Method	Mode	Replacement	Conservation test acc	Partial test acc	Transposition test acc	Full test acc
NPT	global+global	both	conservation	62.14	29.69	36.83	20.32
NPT	global+global	text	conservation	18.28	-	-	-
NPT	global+global	math	conservation	65.23	29.88	37.47	20.45
NPT	global+global	both	partial	60.88	43.68	44.30	30.52
NPT	global+global	math	partial	60.91	41.66	41.73	27.58
NPT	global+global	both	transposition	60.10	31.78	55.28	21.13
NPT	global+global	math	transposition	60.24	28.91	55.02	20.06
NPT	global+global	both	full	45.83	40.53	40.10	35.85
NPT	global+global	math	full	42.52	35.68	34.98	30.36
MathBERT	global+global	both	conservation	45.38	17.82	39.84	14.12
MathBERT	global+global	text	conservation	15.96	-	-	-
MathBERT	global+global	math	conservation	44.13	16.17	39.59	13.15
MathBERT	global+global	both	partial	46.22	33.64	43.22	24.77
MathBERT	global+global	math	partial	42.61	27.22	40.03	18.93
MathBERT	global+global	both	adversarial	46.05	18.11	43.41	14.22
MathBERT	global+global	math	adversarial	42.75	16.12	40.28	13.27
MathBERT	global+global	both	full	32.05	30.11	31.61	28.47
MathBERT	global+global	math	full	27.12	23.36	26.29	20.88
ScratchBERT	global+global	both	conservation	71.38	36.55	66.94	24.67
ScratchBERT	global+global	text	conservation	30.86	-	-	-
ScratchBERT	global+global	math	conservation	60.26	27.34	55.81	19.36
ScratchBERT	global+global	both	partial	70.87	58.06	67.54	45.15
ScratchBERT	global+global	math	partial	58.23	41.31	54.41	29.75
ScratchBERT	global+global	both	transposition	71.41	36.32	70.32	24.79
ScratchBERT	global+global	math	transposition	59.68	27.06	58.62	19.57
ScratchBERT	global+global	both	full	57.73	54.99	56.74	52.31
ScratchBERT	global+global	math	full	43.34	38.17	41.06	34.72

Table A.3: Complete result table using global training and global decoding.

Bibliography

Aggarwal, J. (2017). *Power series*.

Aizawa, A. and Kohlhase, M. (2021). Mathematical Information Retrieval. In Sakai, T., Oard, D. W., and Kando, N., editors, *Evaluating Information Retrieval and Access Tasks: NTCIR's Legacy of Research Impact*, pages 169–185. Springer Singapore, Singapore.

Aizawa, A., Kohlhase, M., and Ounis, I. (2013). Ntcir-10 math pilot task overview. In *NTCIR*.

Aizawa, A., Kohlhase, M., Ounis, I., and Schubotz, M. (2014). Ntcir-11 math-2 task overview.

Alsentzer, E., Murphy, J., Boag, W., Weng, W.-H., Jindi, D., Naumann, T., and McDermott, M. (2019). Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Araci, D. (2019). Finbert: Financial sentiment analysis with pre-trained language models.

Arora, S., Khandeparkar, H., Khodak, M., Plevrakis, O., and Saunshi, N. (2019). A theoretical analysis of contrastive unsupervised representation learning. *CoRR*, abs/1902.09229.

Ausbrooks, R., Buswell, S., Carlisle, D., Chavchanidze, G., Dalmas, S., Devitt, S., Diaz, A., Dooley, S., Hunter, R., Ion, P., Kohlhase, M., Lazrek, A., Libbrecht, P., Miller, B., Miner, R., Sargent, M., Smith, B., Soiffer, N., Sutor, R., and Watt, S. (2010). Mathematical markup language (mathml) version 3.0. *World Wide Web - WWW*.

Baeza-Yates, R. A. and Ribeiro-Neto, B. A. (1999). *Modern Information Retrieval*. ACM Press / Addison-Wesley.

Bahdanau, D., Cho, K., and Bengio, Y. (2016). Neural machine translation by jointly learning to align and translate.

Beltagy, I., Lo, K., and Cohan, A. (2019). SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

BERNAL, J. D., CHADWICK, D., HOLMSTROM, J. E., and FOX, H. M. (1948). The Royal Society Scientific Information Conference. *Nature*, 162(4112):279–286.

Bourne, C. P. and Hahn, T. B. (2003). *Lockheed DIALOG and Related Systems, 1961–1972*, pages 141–183.

Bush, V. (1945). As We May Think. *Atlantic Monthly*, 176(1):641–649.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Coavoux, M. and Cohen, S. B. (2021). Learning to match mathematical statements with proofs. *CoRR*, abs/2102.02110.

Craswell, N. (2009). *Mean Reciprocal Rank*, pages 1703–1703. Springer US, Boston, MA.

Dadure, P., Pakray, D. P., and Bandyopadhyay, S. (2021a). *Mathematical Information Retrieval Trends and Techniques*, pages 74–92.

Dadure, P., Pakray, P., and Bandyopadhyay, S. (2021b). Embedding and generalization of formula with context in the retrieval of mathematical information. *Journal of King Saud University - Computer and Information Sciences*.

de Moura, L. M., Kong, S., Avigad, J., van Doorn, F., and von Raumer, J. (2015). The lean theorem prover (system description). In *CADE*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ferreira, D. and Freitas, A. (2020). Premise selection in natural language mathematical texts. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7365–7374, Online. Association for Computational Linguistics.

Ferreira, D. and Freitas, A. (2021). STAR: Cross-modal [STA]tement [R]epresentation for selecting relevant mathematical premises. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3234–3243, Online. Association for Computational Linguistics.

Ferreira, D., Thayaparan, M., Valentino, M., Rozanova, J., and Freitas, A. (2022). To be or not to be an integer? encoding variables for mathematical text. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 938–948, Dublin, Ireland. Association for Computational Linguistics.

Gao, T., Yao, X., and Chen, D. (2021). Simcse: Simple contrastive learning of sentence embeddings. *CoRR*, abs/2104.08821.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning.

Goldberg, A. V. and Kennedy, R. (1995). An efficient cost scaling algorithm for the assignment problem. *Mathematical Programming*, 71(2):153–177.

Grigore, M., Wolska, M., and Kohlhase, M. (2009). Towards context-based disambiguation of mathematical expressions. In *The joint conference of ASCM 2009 and MACIS 2009. 9th international conference on Asian symposium on computer mathematics and 3rd international conference on mathematical aspects of computer and information sciences, Fukuoka, Japan, December 14–17, 2009. Selected papers.*, pages 262–271. Fukuoka: Kyushu University, Faculty of Mathematics.

Guidi, F. and Coen, C. S. (2015). A survey on retrieval of mathematical knowledge. *CoRR*, abs/1505.06646.

Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., and Smith, N. A. (2020). Don't stop pretraining: Adapt language models to domains and tasks.

Gutmann, M. and Hyvärinen, A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361.

Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742.

Han, J. M., Rute, J., Wu, Y., Ayers, E. W., and Polu, S. (2021). Proof artifact co-training for theorem proving with language models. *CoRR*, abs/2102.06203.

Harman, D. K. (1993). The first text retrieval conference (trec-1).

Jonker, R. and Volgenant, A. (1987). A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland. Association for Computational Linguistics.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Korfhage, R. R. (1997). *Information Storage and Retrieval*. John Wiley amp; Sons, Inc., USA.

Kristianto, G. Y., quoc Nghiem, M., Matsubayashi, Y., and Aizawa, A. (2012). Extracting definitions of mathematical expressions in scientific papers. In *In JSAI*.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. (2019). BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Líska, M., Sojka, P., Ruzicka, M., and Mravec, P. (2011). Web interface and collection for mathematical retrieval :webmias and mrec.

Liu, P., Qiu, X., and Huang, X. (2016). Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 2873–2879. AAAI Press.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.

Líška, M., Sojka, P., Růžička, M., and Mravec, P. (2011). Web Interface and Collection for Mathematical Retrieval: WebMIaS and MREC. In Sojka, P. and Bouche, T., editors, *Towards a Digital Mathematics Library.*, pages 77–84, Bertinoro, Italy. Masaryk University.

Luhn, H. P. (1957). A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):309–317.

Mansouri, B., Rohatgi, S., Oard, D., Wu, J., Giles, C., and Zanibbi, R. (2019). Tangent-cft: An embedding model for mathematical formulas.

Mathematical Sciences, C. o. P. a. L. o. t. (2014). Developing a 21st Century Global Library for Mathematics Research. *ArXiv e-prints*, abs/1404.1905.

Meadows, J. and Freitas, A. (2022). A survey in mathematical language processing.

Megill, N. D. (1969). Metamath a computer language for pure mathematics.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality.

Nghiem Quoc, M., Yokoi, K., Matsubayashi, Y., and Aizawa, A. (2010). Mining coreference relations between formulas and text using wikipedia. In *Proceedings of the Second Workshop on NLP Challenges in the Information Explosion Era (NLPIX 2010)*, pages 69–74. Coling 2010 Organizing Committee.

Padó, S. and Lapata, M. (2006). Optimal constituent alignment with edge covers for semantic projection. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1161–1168, Sydney, Australia. Association for Computational Linguistics.

Pagel, R. and Schubotz, M. (2014). Mathematical language processing project. *CoRR*, abs/1407.0167.

Peng, S., Yuan, K., Gao, L., and Tang, Z. (2021). Mathbert: A pre-trained model for mathematical formula understanding. *arXiv preprint arXiv:2105.00377*.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. volume 14, pages 1532–1543.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Pham, T., Bui, T., Mai, L., and Nguyen, A. (2021). Out of order: How important is the sequential order of words in a sentence in natural language understanding tasks? In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1145–1160, Online. Association for Computational Linguistics.

Piotrowski, B. and Urban, J. (2019). Guiding theorem proving by recurrent neural networks. *CoRR*, abs/1905.07961.

Polu, S. and Sutskever, I. (2020). Generative language modeling for automated theorem proving. *CoRR*, abs/2009.03393.

Polyak, B. T. and Juditsky, A. B. (1992). Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim.*, 30(4):838–855.

Radford, A. and Narasimhan, K. (2018). Improving language understanding by generative pre-training.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. *CoRR*, abs/1602.04938.

Rogers, A., Kovaleva, O., and Rumshisky, A. (2020). A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Sakai, T., Oard, D., and Kando, N. (2021). *Evaluating Information Retrieval and Access Tasks NTCIR's Legacy of Research Impact: NTCIR's Legacy of Research Impact*.

Salton, G. (1971). *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., USA.

Saunshi, N., Malladi, S., and Arora, S. (2020). A mathematical exploration of why language models help solve downstream tasks.

Schmidt, F. and Hofmann, T. (2020). Bert as a teacher: Contextual embeddings for sequence-level reward.

Schöneberg, U. and Sperber, W. (2014). POS tagging and its applications for mathematics. *CoRR*, abs/1406.2880.

Schubotz, M., Grigorev, A., Leich, M., Cohl, H. S., Meuschke, N., Gipp, B., Youssef, A. S., and Markl, V. (2016). Semantification of identifiers in mathematics for better math information retrieval. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 135–144, New York, NY, USA. ACM.

Shen, J. T., Yamashita, M., Prihar, E., Heffernan, N. T., Wu, X., and Lee, D. (2021). Mathbert: A pre-trained language model for general NLP tasks in mathematics education. *CoRR*, abs/2106.07340.

Singhal, A. and Google, I. (2001). Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, 24.

Stamerjohanns, H., Kohlhase, M., Ginev, D., David, C., and Miller, B. (2010). Transforming large collections of scientific publications to xml. *Mathematics in Computer Science*, 3(3):299–307.

Stathopoulos, Y., Baker, S., Rei, M., and Teufel, S. (2018). Variable typing: Assigning meaning to variables in mathematical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 303–312. Association for Computational Linguistics.

Stathopoulos, Y. and Teufel, S. (2015). Retrieval of research-level mathematical information needs: A test collection and technical terminology experiment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 334–340. Association for Computational Linguistics.

Stathopoulos, Y. and Teufel, S. (2016). Mathematical information retrieval based on type embeddings and query expansion. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2344–2355. The COLING 2016 Organizing Committee.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Taskar, B., Lacoste-Julien, S., and Klein, D. (2005). A discriminative matching approach to word alignment. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017a). Attention is all you need.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017b). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Volgenant, A. (1996). Linear and semi-assignment problems: A core oriented approach. *Computers & Operations Research*, 23(10):917 – 932.

Wallace, E., Wang, Y., Li, S., Singh, S., and Gardner, M. (2019). Do NLP models know numbers? probing numeracy in embeddings. *CoRR*, abs/1909.07940.

Welleck, S., Liu, J., Bras, R. L., Hajishirzi, H., Choi, Y., and Cho, K. (2021). Naturalproofs: Mathematical theorem proving in natural language.

Wiedemann, G., Remus, S., Chawla, A., and Biemann, C. (2019). Does bert make any sense? interpretable word sense disambiguation with contextualized embeddings.

Wolska, M., Grigore, M., and Kohlhase, M. (2011). Using discourse context to interpret object-denoting mathematical expressions. In *DML 2011 - Towards a Digital Mathematics Library, Proceedings*.

Yang, S., Wang, Y., and Chu, X. (2020). A survey of deep learning techniques for neural machine translation.

Zanibbi, R., Aizawa, A., Kohlhase, M., Ounis, I., Topic, G., and Davila, K. (2016). NTCIR-12 mathir task overview. In *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies, National Center of Sciences, Tokyo, Japan, June 7-10, 2016*.