MATHEMATICAL LANGUAGE PROCESSING: DEEP LEARNING REPRESENTATIONS AND INFERENCE OVER MATHEMATICAL TEXT

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

2022

By
Deborah Ferreira
School of Computer Science

Contents

Abstract **12**

Declaration **14**

Copyright **15**

Acknowledgements **16**

1 Introduction **17**
1.1 Motivation 17
1.1.1 Mathematical language & natural language 17
1.1.2 AI-based models to support mathematical inference 18
1.2 Methodological Outline 20
1.2.1 Evaluation Framework for MathLP 20
1.2.2 Sentence-level representations 21
1.2.3 Argumentation-level representations 22
1.3 Contributions 24
1.4 Structure 25
1.5 Publications 26

2 Background **28**
2.1 Mathematical Text 28
2.1.1 Objects in the Mathematical Text 29
2.1.2 Languages for Representing Mathematics 32
2.2 Deep Learning Architectures 33
2.2.1 Basic Concepts for Neural Networks 34
2.2.2 Paragraph Vector (`doc2vec`) 37
2.2.3 Long Short-Term Memory Networks 39

2.2.4 Transformers . . . 41
2.2.5 Pre-trained Language Representations . . . 43
2.2.6 Sentence-based Representations . . . 44
2.3 Information Retrieval . . . 45
2.4 Summary . . . 46

3 Literature Review **48**
3.1 Mathematical Language Processing (MathLP) . . . 48
3.2 Mathematical Information Retrieval . . . 49
3.3 Mathematical Element Typing . . . 51
3.4 Mathematical Document Categorisation . . . 53
3.5 Mathematical Problems . . . 55
3.6 Plagiarism Detection for Mathematical Content . . . 57
3.7 Testing Large Language Models for Mathematics . . . 57
3.8 Other Tasks . . . 58
3.9 Discussion . . . 59

4 Natural Language Premise Selection **61**
4.1 Motivation . . . 61
4.2 Premise selection . . . 62
4.3 Definition: Natural Language Premise Selection . . . 64
4.4 Dataset Construction: PS-ProofWiki . . . 65
4.4.1 Parsing the corpus . . . 66
4.4.2 Cleaning wiki tags . . . 66
4.4.3 Curating pages . . . 68
4.4.4 Extraction of categories . . . 68
4.4.5 Extracting premises . . . 69
4.4.6 Annotating mathematical text . . . 70
4.4.7 Structuring the entries into Training, Development and Validation set. . . . 70
4.5 Dataset Analysis . . . 70
4.6 Baselines . . . 73
4.7 Discussion . . . 76
4.8 Reproducibility Statement . . . 78

5 Retrieving Relevant Premises **80**
5.1 Motivation . . . 80
5.2 Proposed Method . . . 82
5.2.1 Statement Similarity Relevance Score (StaSim) . . . 82
5.2.2 Structural Similarity Relevance Score (StructSim) . . . 83
5.3 Conjecture Premise Relevance Score . . . 84
5.4 Experiments . . . 84
5.5 Discussion . . . 87
5.6 Reproducibility Statement . . . 89

6 Linguistic modalities in the mathematical text **90**
6.1 Motivation . . . 90
6.2 Natural Language Premise Selection as Inference Task . . . 91
6.3 Cross-modal STAtement Representation (STAR) . . . 92
6.3.1 Token embedding layer . . . 92
6.3.2 Word/Expression-specific Self-Attention Layer . . . 92
6.3.3 Long Short-Term Memory Layer . . . 94
6.3.4 Training objective . . . 94
6.4 Experiments . . . 95
6.4.1 Quantitative Analysis . . . 96
6.4.2 Qualitative analysis . . . 100
6.5 Discussion . . . 101
6.6 Reproducibility Statement . . . 102

7 Connecting mathematical statements **103**
7.1 Motivation . . . 103
7.2 Encoding mathematical propositions and premises . . . 104
7.2.1 Graph construction . . . 104
7.2.2 Subgraph extraction . . . 104
7.2.3 Node features . . . 105
7.3 Proposed Model: Premise Selection based on DGCNNs . . . 106
7.3.1 Design Principles . . . 106
7.3.2 Detailed Model . . . 107
7.4 Experiments . . . 108
7.4.1 Baseline: BERT . . . 108
7.4.2 Quantitative analysis . . . 109

7.4.3 Scalability & Imbalance Robustness analysis 109
7.4.4 Qualitative analysis 110
7.5 Discussion 111
7.5.1 Reproducibility Statement 112

8 Improving Variable Understanding **114**
8.1 Motivation 114
8.2 Variable Typing problem 116
8.3 Proposed Method: Variable Slots (VarSlot) 117
8.3.1 From types and expressions to hypotheses 117
8.3.2 Encoding the sentences 118
8.3.3 Representing Variables with Slots 118
8.3.4 Classifying hypotheses 119
8.4 Experiments 120
8.4.1 Dataset 121
8.4.2 Baselines 121
8.4.3 Quantitative results 122
8.4.4 Robustness to substitution 124
8.4.5 Ablation Studies 125
8.4.6 Qualitative Analysis 126
8.5 Discussion 128
8.6 Reproducibility Statement 129

9 Conclusion and Future Work **131**
9.1 Summary and Conclusions 131
9.2 Opportunities for Further Research 136

Glossary **138**

A LaTeX vs MathML **149**

B PS-ProofWiki generation **151**
B.1 ProofWiki Source Code Example 151
B.2 Example from PS-ProofWiki 152

Word Count: 40,547

List of Tables

2.1 Sentence-Transformer pre-trained models used in this work. 45

3.1 Tasks that were previously proposed in the field of Natural Mathematical Language Processing. 49
3.2 Summary of different methodologies for addressing MathLP tasks. . . 50
3.3 Example of two mathematical problems with answers. Example obtained from [70]. 56

4.1 Types of mathematical statements present in PS-ProofWiki. The table shows the number divided by the data split. The last columns shows the total unique entries for each mathematical type. 71
4.2 Result for mAP applying BM25 and comparing tokenisation of expressions. The values for mAP are multiplied by 100. 75
4.3 Results for different Sentence-Embedding models. The results are shown for mAP $\times$ 100. 75
4.4 Comparing mAP results for different categories (the number between parenthesis indicates the number of entries for that category). For SBERT, we use the best performing model. 76
4.5 Comparing number of hops needed for obtaining premises. 76

5.1 Results for premise selection retrieval. Here we compare our approach with pre-trained models and BM25. 85
5.2 Ablation studies, comparing the different components that are part of the Conjecture-Premise relevance score. 86

6.1 Number of entries for Training, Validation and Test for different values of n. 95
6.2 Comparison of STAR with a model containing a single self-attention layer. 97

6.3 Results for STAR and baseline for different number of negative examples (n) using similar examples. 97
6.4 Distribution of dataset with different topics. 98
6.5 Testing the transferability across different mathematical areas. For these experiments, we use random examples with $n = 1$. 99
6.6 Comparison of our model with other baselines, using n=1 and random examples. 99
6.7 Some of the premises existing in the dataset, together with the predictions from STAR. 100

7.1 Precision (P), recall (R), and F1-score (F1) for the BERT baseline and the proposed method, with 30 negative examples for each positive case (values are multiplied by 100). 109
7.2 Comparison of BERT and the proposed model for different levels of transitivity between premises (values are multiplied by 100). 110

8.1 Comparison of our method with different baselines for Variable Typing for both the Classic and Renaming Setting. 122
8.2 Comparison of our method for the different settings. 125

List of Figures

1.1 Structure of the thesis, mapping research questions to chapters, and chapters to contributions. 25

2.1 Logistic function $\sigma(x)$ with $c = 1$. This function maps the input to values between 0 and 1. 35
2.2 Hyperbolic tangent $tanh(x)$. This function maps the input to values between -1 and 1. 35
2.3 ReLU activation function. It behaves linearly for $x \geq 0$ 36
2.4 Example of the framework `word2vec`, used to learn vector representations for words. Figure adapted from Le and Mikolov [48]. 37
2.5 Example of the framework `doc2vec` (PV-DM), used to learn vector representations for words and vector representations for the paragraph. Figure adapted from Le and Mikolov [48]. 38
2.6 Paragraph Vector using a Distributed Bag of Words setting (PV-DBOW). The paragraph vector is trained to predict the words in a small window. Figure adapted from Le and Mikolov [48]. 39
2.7 Diagram of an LSTM cell. Figure adapted from Olah [59]. 40
2.8 Diagram of the Encoder and Decoder of a Transformer model. Figure adapted from Vaswani et al. [85]. 41

3.1 Example of a mathematical query and the expected retrieved formula. Example obtained from [100]. 51
3.2 Example of the mathematical element typing task. The types and mathematical elements are marked in this sentence. 51
3.3 Example of the Mathematical Document Categorisation task. 53

4.1 Example of a premise of type definition. 64
4.2 Example of part of a proof, where four premises are present. 65
4.3 Example of a conjecture and its premises. 66

4.4 Pipeline used to build the PS-ProofWiki dataset. 67
4.5 An example where Mathematical text 1 references a passage in Mathematical text 2 using the name of the passage to be referenced between curly brackets. Only the part highlighted is being referenced. 68
4.6 Example of premises found inside a proof. 69
4.7 Distribution of documents per category in the dataset. 72
4.8 Distribution of the number of premises in the ProofWiki corpus. Log transformation is applied to facilitate visualisation for the *y* axis. . . . 73
4.9 Number of times a statement is referred as a premise. Log transformation is applied to facilitate visualisation for the *y* axis. 74
4.10 Example of premises in a multi-hop scenario. 76

5.1 Different λ values and *k* for BM25. 87
5.2 Different λ values and *k* for SBERT. 87

6.1 Example of a positive and a negative pair. 92
6.2 This figure presents the model used to generate a representation for each statement, where we combine two self-attention layers, one for each token modality. 93
6.3 Siamese Network used for classifying the conjecture-premise pair, based on the representations obtained. 94
6.4 Number of training epochs and the obtained validation F1-score. . . . 96
6.5 Comparison of our model and the baseline on the capability of predicting correctly statements with different levels of entanglement. 101

7.1 Sub-graph extraction for link prediction. 105
7.2 Pre-processing workflow of the proof corpus. 106
7.3 Depiction of the DGCNN architecture used in the premise selection task. 108
7.4 Comparison of the proposed model and BERT, showing how both models perform (in terms of F1-score) when adding more negative examples to the training and test set. 113

8.1 Example of a mathematical statement containing one variable. This example shows two different edges (variable→type), with a positive and a negative binding. 117
8.2 Our model takes as input a statement containing words and variables and obtain the classification of the hypothesis. 117

8.3 F1 score obtained for different models when adding the same hypotheses with different variable name substitutions. 125
8.4 Embedding of the words and variables obtained from the test set for SciBERT-NLI and VarSlot. 127
8.5 Results for the probing task, where we compare the representations generated from SciBERT-NLI (yellow) and VarSlot (blue). 130

Abstract

MATHEMATICAL LANGUAGE PROCESSING: DEEP LEARNING REPRESENTATIONS AND INFERENCE OVER MATHEMATICAL TEXT

Deborah Ferreira

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy, 2022

The articulation of mathematical arguments is a fundamental part of scientific reasoning and communication. Across many disciplines, expressing relations and interdependencies between quantities (usually in an equational form) is at the centre of scientific argumentation. One can easily find examples of mathematical discourse across different scientific contributions and textbooks. Nevertheless, despite its importance, the application of contemporary NLP models for performing inference over mathematical text remains under-explored, especially when compared with other advances in natural language processing and domain-specific text mining (e.g. biomedical text). In this work, we contribute to the area of Mathematical Language Processing, which addresses problems in the intersection of Natural Language and Mathematics. While several aspects of the mathematical discourse are still unexplored in this field, we have opted to focus on three main dimensions: (i) defining an evaluation framework for mathematical natural language inference; (ii) learning sentence-level representations of mathematical statements; (iii) leveraging argumentation-level premise-claim discourse relations between mathematical statements.

The discovery of supporting evidence for addressing complex mathematical problems is a semantically challenging task, which is still unexplored in the field of natural language processing for mathematical text. In this work, we propose the Natural Language Premise Selection task, together with a new dataset, which consists in using conjectures written in both natural language and mathematical formulae to recommend premises that most likely will be helpful to prove a particular statement. Another fundamental requirement towards mathematical language understanding is the creation

of models able to represent variables meaningfully. In this work, we propose different deep learning based techniques to address such issues, identifying the challenges associated with such tasks and pave the way for future work in this field.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made **only** in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see `http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=487`), in any relevant Thesis restriction declarations deposited in the University Library, The University Library's regulations (see `http://www.manchester.ac.uk/library/aboutus/regulations`) and in The University's policy on presentation of Theses

Acknowledgements

First, I would like to thank my supervisor Andre Freitas, who has opened the doors for me to grow and become a better researcher. My co-supervisor, Uli Sattler, also played a significant role in my PhD and I feel fortunate to have had her as a mentor. I wish all PhD students had the support of someone like her.

I was also incredibly lucky to be surrounded by fantastic lab colleagues. Mokanarangan and Marco were my rock during the PhD process, helping me as friends and as great researchers. I will miss tremendously working with them, and I hope our work paths will cross again in the future (the friendship path will be there forever).

I would also like to thank Jake and Julia for always being there and showing me a bit of fun during the tough PhD days. I am also grateful for all the people I got to meet and work with, including but not limited to Mauricio, Hanadi, MingYang, Oskar, Salim, Crystal, Guy, and Conor.

My family has also played a crucial role during this process, giving me the emotional support to push through the difficult times. Their choices have paved the way for me to follow this path. My parents have sacrificed so much to ensure my siblings, and I had access to good education, which is not easy where I come from.

Finally, I would like to thank my partner, Alber Flores; every time I thought about giving up or that I was not good enough for this, he was the one to show me otherwise. This work is as much his as it is mine.

Chapter 1

Introduction

1.1 Motivation

1.1.1 Mathematical language & natural language

Mathematics plays a substantial role in all the technological, scientific, medical, educational and economic accomplishments in our society. Mathematics can be conceptualised as an anthology of mathematical theories where each theory is a set of theorems and has models in which the theorems hold. These models are composed of abstract elements, operations on these elements and relations between these elements. Using theorems, we can make claims about the elements, relations, and operations on that set [86]. Mathematics is enriched each year by a mass of new results and theories, rendering it impossible for a mathematician to master all mathematics [14]. Mathematics is continually growing as mathematicians create new theories accepted by the scientific community.

The objects present in Mathematics are entirely abstract and can never be physically observed in the real world. In order to talk about these objects, mathematicians first had to create a new language and conceptual framework to be able to describe and discuss these abstract notions. For example, we cannot physically see an infinite number, but we can detail and operate over infinite values with this language. This renders mathematics distinct from natural language, which has its own syntax and semantics. For example, variables have defined types and expressions have to follow structural rules. Having created both vocabulary and rules for the language, mathematicians seek to discover what new theorems can be proven over numbers and space. A similar phenomenon can also be seen in other sciences. For example, in the biomedical

domain, we can find several concepts (e.g., name of genes) inside the text that is very specific to that domain. In Chemistry, we often see scientific text mentioning chemical compounds and how chemical reactions are formed, which are described with a notation unique to that field.

The *mathematical language*, or the language of mathematics, sits at the union between natural language and mathematics. Mathematical language is the language found in scientific papers or textbooks about mathematics (or subjects that depend heavily on mathematics, such as Physics). One significant difference between Mathematical Language and Natural Language arises from the fact that mathematical language is more precise and less flexible than the structure of natural language [63]. In natural language, we can have a difference between the surface and deep structures of the utterance, while in mathematical language, for each surface structure, there is one equivalent deep structure, where all statements are unambiguous [40].

1.1.2 AI-based models to support mathematical inference

The interactions between computational models and different human languages, such as the mathematical language, has been a concern of the NLP area, which is a sub-field of linguistics and computer science [19]. NLP aims to enable computers to process human languages intelligently and is an essentially interdisciplinary field crossing artificial intelligence, computer science, cognitive science, information processing, and linguistics [23]. Examples of NLP applications include speech recognition, information retrieval, question answering and machine translation.

Symbolic interpreters can readily understand statements written using formal languages employing inference rules. Additionally, a machine can trivially evaluate the expression "$1 + 1 =$?" since the algorithmic/operational aspects of numerical values and addition are already deeply embedded into the machine hardware. The same is not valid for the problem: "*I have one apple, and I get another one, how many apples do I have?*", even though both statements require a similar type of reasoning for humans. The evident differences between the two statements is the level of formality and abstraction.

Recent advances in NLP enabled by Deep Learning-based architectures bring the opportunity to support the interpretation of textual content at scale. The application of these methods can facilitate scientific discovery [44], reducing the gap between current research and the available large-scale scientific knowledge. These models, however, are still limited in their ability to interpret abstract mathematical knowledge.

State-of-the-art models for natural language processing, such as Transformer-based models [24], have defined the recent state-of-the-art for several tasks, such as entity recognition [90], textual entailment [51] and machine translation [26], but there is no evidence that they encode the intricate mathematical background knowledge needed to reason over mathematical discourse [31].

Articulating mathematical arguments is a fundamental part of scientific reasoning and communication. Across many disciplines, expressing relations and interdependencies between the abstract set of mathematical elements (usually in an equational form) is at the centre of scientific argumentation. One can easily find examples of mathematical discourse across different scientific contributions and textbooks. Nevertheless, despite its importance, the application of contemporary NLP models for performing inference over mathematical text remains under-explored, especially when compared with the advances in natural language processing and domain-specific text mining. Fields such as BioNLP (i.e., NLP focused on biomedical and molecular biology texts) are booming with shared tasks [81] and workshops; however, the same is not observed for applications of NLP for mathematical text.

While the Language of Mathematics and Mathematics have an evident relationship, they are distinct concepts. The Language of Mathematics is the language of the notational forms used in Mathematics, while Mathematics is what one does in and with this language [8]. Most of the research in neural network-based models has focused on Mathematics itself. For example, research focusing on inferring solutions to general nonlinear partial differential equations [64] or finding a numerical solution of parametric families of high-dimensional linear Kolmogorov partial differential equations [12]. These works consider Mathematics in isolation, without encoding the language context where equations might have been inserted. Also related to Mathematics, notable achievements have been obtained on the application of neural methods for automatic reasoning performed on top of formal languages such as Mizar [9, 91]. The Mizar system consists of a formal language for writing mathematical definitions and proofs, a proof assistant, which is able to check proofs written in this language mechanically. This type of system relies on formally defined rules, following strict symbolic rules. While such systems help verify proofs that are written in a formal language (such as Mizar), this language is very different from mathematical language. The vast majority of mathematical knowledge currently available is written in natural mathematical language, which is more readable and generally considered easier to use [89].

1.2 Methodological Outline

This thesis aims to understand and quantify how contemporary Deep Learning-based NLP models behave in a mathematical language setting and to explore how to improve the ability of these models to represent and infer over mathematical text. While several aspects of the mathematical discourse are still unexplored in this field, we have opted to focus on three main dimensions: (i) defining an evaluation framework for mathematical natural language inference; (ii) learning sentence-level representations of mathematical statements; (iii) leveraging argumentation-level premise-claim discourse relations between mathematical statements.

Considering the focus of this work, we define one central high-level research question:

RQ: *Can Deep Learning-based representation models support understanding and inference over mathematical text corpora?*

The application of Deep Learning techniques in the area of Mathematical Language Processing is still very unexplored, despite its promising results across different NLP tasks. This work evaluates, improves, and extends state-of-the-art pre-trained language models to verify their performance in suitable mathematical text. With this exploration, we propose new models specific to the mathematical domain using the deep learning representational framework.

In this work, we consider MathLP as a subfield of NLP which focus on Mathematical Text. In order to explore further this RQ, we stated the core target research questions that are investigated and answered across different chapters of this work.

1.2.1 Evaluation Framework for MathLP

RQ1: *How to support the evaluation of models for the interpretation and inference over mathematical text (regarding inference tasks, evaluation measures and supporting datasets)?*

A significant gap in this field is the curation and availability of datasets that can be used to train and test MathLP models. In order to deepen our understanding of the performance of pre-trained language models for the mathematical domain, we provide a new large-scale benchmark for evaluating MathLP, describing a methodology to obtain a new dataset, together with a new Mathematical Language Processing task to answer this research question. We also use another existing large-scale dataset, the Variable Typing Dataset, to further explore RQ1. Investigating this question will allow us to

identify the performance of pre-trained language models and canonical information retrieval baselines, such as BM25, for our newly designed task and the variable typing task, identifying possible representational and inference gains within this domain.

1.2.2 Sentence-level representations

We often alternate between mathematical elements and words in the same statement when writing mathematical statements. In the statement: "Let n be an integer such that $n \geq 2$.", we can observe the entanglement between the mathematical elements and words. While natural for humans, this causes issues for deep learning models to generalise over these two linguistic modalities. Additionally, these models tend to be bound to represent variables as regular tokens (as opposed to a symbol-agnostic semantics), requiring explicit training to encode those elements so that its meaning is independent on the surface and entirely dependent on context. For example, replacing the variable n with another variable name should not change the representation generated by a model.

Similar to what can be seen in the Biomedical domain, mathematical language is full of statements containing a combination of tokens from natural language and highly specialised domains. In the case of Mathematics, we see the combination of mathematical and natural language elements. Some pre-trained language models excel at performing multi-language inference, working for different languages simultaneously (e.g. text in English and German). For example, the model XLM-Roberta[1] works for more than 100 languages; however, it does not include mathematical language. There is still a gap on how to represent mathematical elements together with natural language, directing to the following research question:

RQ2: *Can we improve the performance of deep learning models for inference over mathematical text by generating specialised encodings for mathematical natural language elements?*

Investigating this question will highlight the different phenomena present in the mathematical text. We explore how to improve the understanding mathematical expressions in a textual context, introducing a multi-modal approach that encodes mathematical elements and natural language using one specific self-attention layer for each modality, inducing the model how to represent different modalities into two different semantic spaces. Answering this question will highlight possible new architectures that should be considered when designing models for mathematical text.

[1] https://huggingface.co/xlm-roberta-large

When encoding mathematical elements, we should also consider that it comes with a unique associated challenge: the representation of variables. Variables have a unique behaviour, where their surface representation (their name, such as x or y) adds very little to the meaning of the variable. The meaning of a variable should come entirely from the context it is inserted. While some variable names can give us hints on their meaning, such as F for *force*, their meaning can change if inserted in a different document (e.g. using F for field). Our final research question will investigate how to deal with variables:

RQ3: *How to encode the semantics of variables at sentence-level representations?*

The investigation of this question will highlight the behaviour of variables inside the mathematical text. While these elements have a very well-defined role inside Mathematics, there is no standard representation paradigm on how they should be encoded within MathLP models. To answer this question, we propose a novel slot-based approach that uses slot attention to forget the surface representation of a variable and relearn its representation from context. Answering this question will define methodologies to evaluate the representation of variables, showing what behaviour should be expected with these representations.

1.2.3 Argumentation-level representations

As previously mentioned, another aspect that we explore in this work is the structure of relationships and dependencies between mathematical statements. In Mathematics, every discovery adds more knowledge to the already vast bulk of mathematical knowledge, and often these discoveries are built on top of previous ones. A theorem proved using the severe constraints of logic will be a theorem forever unless it has been disproved or the proof has been found to be flawed. For example, while most of the astronomical theories from several centuries ago have been completely discarded, we still have theorems such as the Pythagorean theorem proved in 300 B.C., which is still valid today. The mathematician Hermann Hankel once stated [69]:

> "In most sciences one generation tears down what another has built, and what one has established another undoes. In mathematics alone each generation adds a new story to the old structure."

The term structure can refer to many levels of abstractions inside the mathematical text. For example, the argumentation structure found inside a mathematical proof,

where one statement leads to the following conclusion, can form a dependency structure. In this work, we use the structure of dependencies inside the mathematical text to refer to the dependencies of related statements. For example, a theorem forms a structure when considering related definitions and other theorems. The former type of structure is outside the scope of this work.

Using pre-trained language models, we can compare the semantic similarity of different mathematical statements. This setup has been widely employed for different NLP tasks, such as semantic search and question answering. However, previous research [84] has found that using direct similarity for retrieving explanations in the science domain for question answering has several limitations. Often the query (e.g. a question about a ball falling) is not semantically or lexically similar to the explanation for that question (e.g. how gravity works). Given that this has been explored for the general science domain, it is sensible also to ascertain if the same is valid also for the mathematical domain, leading us to our next research question:

RQ4: *Can the dependencies between mathematical statements provide a simplified representation of mathematical arguments?*

Traditional retrieval models will often use techniques that rely on the semantic or lexical similarity between query and elements in the knowledge base. We propose a pipeline for mathematical statement retrieval based on the unification score [84], where the guiding principle is the structure of premise-conjecture relationships. Answering this research question will allow us to understand better the problem of retrieving mathematical statements, focusing on our new benchmark (RQ1), delivering a more simplified and pragmatic method for representing how different statements relate and how it compares with the general behaviour of scientific statements.

An essential feature of mathematics that should be regarded when conceiving models to encode mathematical statements is that mathematical statements do not live in isolation. For example, when we read about a theorem, we will often have to understand all the background knowledge associated with that theorem. If one wants to understand the *Euclid's lemma*, they first need to learn what a *prime number* is. This hierarchical structure, where one statement depends on the knowledge provided by a previous statement, can be used to create a more suitable representation of mathematical statements, as we explore in the subsequent research question:

RQ5: *How to encode the inter-statement dependency structure to support inference for the natural language premise selection task?*

Mathematical statements naturally entail a dependency structure, where new proofs

are built on top of previous ones. We leverage this structure using Deep Graph Convolutional Neural Networks, structures capable of encoding the relationship between different elements in a text. By exploring this research question, we will further comprehend how mathematical statements relates to one another, reinforcing the idea that Mathematics is a constructive field, where each discovery is built on top of previous ones.

We hope that the creation of methods and models that can interpret mathematical text and discourse will pave the path towards developing systems capable of complex mathematical inference, which will lead to automated scientific discovery in fields that depend on mathematical knowledge. We expect that this thesis can act as an initial push towards applying Deep Learning methods for encoding abstract and scientific discourse and to Mathematical Language Processing.

1.3 Contributions

The contributions of this thesis can be summarised as follows:

- **C1:** We provide a literature review on current research on Mathematical Language Processing, together with an analysis of explored tasks and datasets that are available. We provide a taxonomy of these different tasks according to different features, allowing for easy comparison and identification of current gaps in the literature. We are the first to provide a unified comparison of Mathematical Language Processing tasks. (**RQ1**)

- **C2:** We propose a new task for Mathematical Language Processing, the Natural Language Processing Task, together with the construction and analysis of this new task. We also design and implement different baselines for this task, comparing Bag-of-word type of models with transformer-based approaches. (**RQ1**)

- **C3:** We demonstrate how to leverage the properties of related statements to improve the large scale retrieval of mathematical statements for natural language premise selection. We empirically show that semantic related statements have a higher number of shared dependencies. (**RQ4**)

- **C4:** We propose a novel cross-modal embedding that captures the different modalities inside mathematical text: mathematical elements and words, providing a systematic analysis of the transferability of this representation across

different mathematical domains and an empirical evaluation, comparing our approach with state-of-the-art models the performance of supporting ablation studies. (**RQ2**)

- **C5:** We introduce a new approach addressing the natural language premise selection task using link prediction under a Deep Convolutional Graph Neural Network representation, providing quantitative and qualitative evaluation against existing baselines. We find that we can perform better than large pre-trained language models by leveraging the graph structure that can be built using the dependencies between statements. (**RQ5**)

- **C6:** We propose a new evaluation framework for testing the property of generalisation to variable renaming while systematically analysing the understanding of variables in large language models, demonstrating their limitations when handling variable renaming. We also propose a state-of-the-art model for the variable typing task, demonstrating, at the same time, that it significantly outperforms large language models when analysing the property of generalisation to variable renaming. (**RQ3**)

1.4 Structure

This work is structured as follows (Figure 1.1):

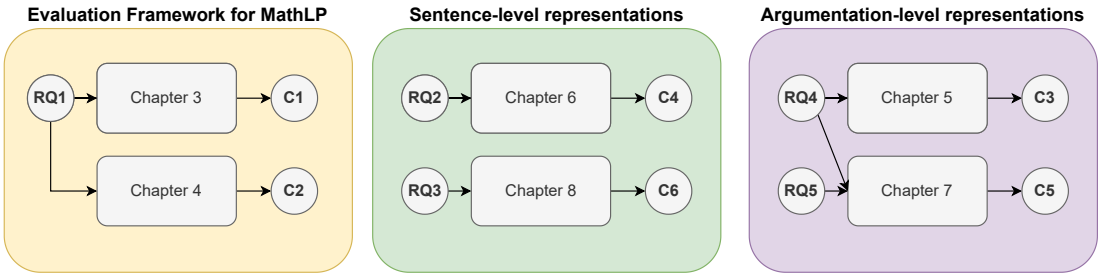


Figure 1.1: Structure of the thesis, mapping research questions to chapters, and chapters to contributions.

Chapter 2 introduces the basic building blocks of mathematical text, defining what mathematical elements and their surrounding concepts are. This chapter also introduces the Deep Learning models that are used across this work, together with the definition of essential baselines. In **Chapter 3**, we analyse previous work that has been explored in the field of mathematical language processing, together with an overview

of the state-of-the-art in this field. This chapter identifies the gaps in this field, paving the way for the following chapters. In **Chapter 4**, we introduce a new dataset together with a new task: Natural Language Premise Selection. We also introduce a set of baselines to verify the challenge associated with such a task. After performing the initial baselines, we found that this is not a trivial task and the rest of the thesis focuses on different techniques to improve those results. In **Chapter 5**, we design an approach to leverage the structural dependencies in the mathematical domain. We show that to improve retrieval for mathematical statements for the natural language premise selection task, and we need not only to pay attention to the lexical/semantic similarity but also to the structural similarity between them.

Chapter 6 presents a new model to represent mathematical elements and natural language in different spaces. This chapter explores the performance gain that can be achieved through having specific self-attention layers for mathematical elements and natural language, allowing the model to learn about the difference between these modalities of objects. In **Chapter 7**, we come back to the idea of dependencies between mathematical statements. We design an approach based on deep graph convolutional neural networks to improve inference over mathematical text. **Chapter 8** looks at how variables are represented internally inside large language models and propose a new benchmark for testing the robustness of these representations. We design a new approach for typing variables that results in better inference and obtains promising results in terms of robustness. Finally, in **Chapter 9**, we return to the research questions proposed, providing a discussion on the answers obtained, together with the future research areas still open for exploration in this field.

1.5 Publications

The list of publications obtained during the production of this thesis, and associated chapters, are:

- **Chapter 5**: *Deborah Ferreira and André Freitas. "Natural Language Premise Selection: Finding Supporting Statements for Mathematical Text." Proceedings of the 12th Language Resources and Evaluation Conference (LREC). 2020.*

- **Chapter 7**: *Deborah Ferreira and André Freitas. "STAR: Cross-modal STAtement Representation for selecting relevant mathematical premises." Proceedings*

of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL): Main Volume. 2021.

- **Chapter 8**: *Deborah Ferreira and André Freitas. "Premise selection in natural language mathematical texts." Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL).*

- **Chapter 9**: *Deborah Ferreira, Mokanarangan Thayaparan, Marco Valentino, Julia Rozanova and André Freitas. "To be or not to be an Integer? Typing Variables inside the mathematical discourse." Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics(ACL Findings), 2022.*

- **Chapter 9**: *Deborah Ferreira, Mokanarangan Thayaparan, Marco Valentino, Julia Rozanova and André Freitas. "Does My Representation Capture X? Probe-Ably" Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP): System Demonstrations, 2021.*

- **Workshop Organisation**: Organisation of the first workshop on MathLP: Mathematical Language Processing (Accepted for the Conference on Empirical Methods in Natural Language Processing (EMNLP) 2022).

Chapter 2

Background

This chapter introduces the essential concepts for understanding the contributions of this thesis, with a particular emphasis on defining the objects present in mathematical text from a linguistic perspective, together with different approaches that are employed across this work to encode those mathematical objects. However, this chapter is not intended to be an extensive explanation of such concepts. For such purposes, the references to main literature related to these topics have been included here.

2.1 Mathematical Text

The *Language of Mathematics*, or Natural Mathematical Language, emerged out of the need created by mathematicians to be able to reason logically about things. Some notable aspects of this language are its precision and lack of ambiguity. Mathematical objects can only be used once formally defined, ensuring that their meaning maps precisely to the original intent. Unlike natural language, where one can often see ambiguous statements, this is highly discouraged when writing mathematics and can lead to incorrect statements and invalid proofs. Mathematical statements should not be subjective to reader interpretation: they need to be valid for any person who has the background knowledge to understand them. These aspects render mathematics a powerful tool for describing and reasoning over technical documents.

This language succeeds through the entwining of natural language, mathematical symbolism and visual cues, effortlessly shifting across these three semiotic modalities, where each one of these has a specific contribution or function within mathematical discourse [58]. In this work, we focus on the *Mathematical Text*, which is discourse written using the Language of Mathematics.

2.1.1 Objects in the Mathematical Text

Mathematical objects have been widely studied and have well-established definitions. While this work does not attempt to redefine these objects in mathematics, it is paramount to define how these concepts surface in the mathematical text from a linguistic perspective. The following definitions should be considered in the context of the mathematical text.

Definition 1. Let X be set of variable symbols, C a set of constant symbols, and F a set of function symbols where each function symbols have an n-arity. The set of *mathematical expressions* $\mathcal{M}$ over X, C, and F is the smallest set such that:

(i) every variable symbol x in X is an expression;

(ii) every constant symbol c in C is an expression;

(iii) if $m_1,...,m_n$ are all expressions and f in F is an n-ary function symbol, then $f(m_1,...,m_n)$ is an expression.

We consider variables and context as *simple expressions* given its usual smaller size in terms of the number of symbols. For example, the composed expression $x+y+1$ contains two variables (simple expressions x and y) and a binary function symbol ($+$). The set of possible function symbols will be different depending on the mathematical category of the text.

Definition 2. Let R be the set of relationship symbols, and $\mathcal{M}$ the set of expressions, then the set of *formulas* $\mathcal{F}$ is defined as:

(i) If $m_1,...,m_n$ are all expressions and r in R is an n-ary relationship symbol, then $r(m_1,...,m_n)$ is a formula;

(ii) If $f_1,...,f_n$ are all formulas and r in R is an n-ary relationship symbol, then $r(f_1,...,f_n)$ is a formula.

A formula is a specific type of composed expression, where a function acts as a verb to establish the relationship between different expressions [49]. *Equations* and *inequations* are all types of formulas. For example, the equation/inequation $x+1=3$ and $y+1\neq t$ are both formulas, containing the binary functions $=$ and $\neq$, respectively, where these functions have a property analogous to a verb.

Definition 3. In this work, formulae and mathematical expressions are interchangeably referred to as *mathematical elements*.

This choice is mainly made for brevity and avoids redundancy when referring to such elements.

Definition 4. A *mathematical statement* μ can be usually defined as:

(i) One or more sentences;

(ii) One or more sentences with one or more expressions acting as a noun phrase, and/or one or more formulas acting as clause;

(iii) A formula by itself.

Statements do not include exclamations, questions, or orders. A statement cannot be true and false at the same time [20]. Examples of mathematical statements of each type are:

(i) "A Euclidean domain is a principal ideal domain."

Note that for a natural language sentence to be considered a mathematical statement, it must be inserted in a Mathematical context. For example, the sentence *"Every group is finite"* can be a mathematical statement when referring to the mathematical concept of groups, yet, it can also refer to a group of people, which would make it a simple natural language sentence.

(ii) "Let G be a group such that $|G| = kp^n$"

This statement contains one expression (G) with a noun phrase function and one formula $|G| = kp^n$ with a clause function.

(iii) "$\int_0^{\frac{\pi}{2}} \sin^2 x \, \mathrm{d}x = \frac{\pi}{4}$"

Often the elements inside the formula require some introduction, appearing in the sentence before the introduction of the formula.

Mathematical statements usually follow certain modalities, categorising the mathematical statement in the context it appears. Some of the common modalities are [104]:

- Assumed truth: statements used to create assumptions valid for the proposition context, commonly used to define/declare variables and functions.

  Example: "Let p be a prime number."

- Accepted Truth: statements that are universally accepted in the field, applying definitions and axioms or even reusing past proven results.

  Example: "Since we have that $p|pb$ ..."

- Operation: statements describing operations or actions that are performed in order to achieve a certain result.

  Example: "Multiplying both sides by 3 ..."

- Conclusion: a statement that presents a conclusion derived from an accepted/assumed truth or from an operation.

  Example: "... then y is an integer and $x+1=2y$."

- Clarification: a statement that details a previous statement.

  Example: "That is, the only harmonic numbers that are integers are H_0 and H_1."

- Subjective: statements that are subjective to human interpretation, i.e., are not universally accepted as true or false.

  Example: "The proof is trivial."

Definition 5. A *mathematical text* $\mathcal{T}$ is a finite non-empty sequence $\{\mu_1, \mu_2, \mu_3, ..., \mu_n\}$ of mathematical statements.

In mathematical text, words, expressions and formulas, can be directly related through a relationship between *definiendum* and *definiens*, where a mathematical element, the definiendum, is defined by a mathematical statement or part of a mathematical statement, the definiens is also used to determine the *set of values* and properties associated with an expression.

For example, in the statement "Let n be a natural number greater than 0.", the expression n has one composed definiens: *natural number* composed with *greater than zero*.

Mathematical text can belong to particular categories; the ones relevant for this work are definitions, axioms, theorems, corollaries, lemmas and proofs.

Definition 6. A *definition* is an accurate description of a word or phrase in terms of other, more basic words or phrases [5]. Definitions do not require proof.

For example, the definition of a prime number can be given as *"A prime number is an integer greater than one that has no factors other than one and itself"*. The definition of concepts allows future references to such concepts concisely. For instance,

given that the 'prime number' is already defined, it can be reused across different statements without the need for redefinition.

Definition 7. An *axiom* is a mathematical text taken to be true, it can be used as starting point for further reasoning and arguments. Axioms are accepted without proof.

Axioms act as fundamental building blocks for more complex theories. Other types of mathematical texts require proofs, that is the case for theorems, lemmas and corollaries.

Definition 8. A *theorem* is a mathematical text whose truth can be established using logical reasoning based on certain assumptions [20].

Definition 9. A *lemma* is also a theorem, whereas it is used to support the argument for another theorem.

Definition 10. A *corollary* is a theorem that follows logically and directly from a previously proven theorem.

These concepts highlight the highly connected nature of mathematical discourse, where definitions and axioms are used to prove theorems, lemmas, and corollaries. Lemmas support the proof of theorems and theorems aid in the output of new corollaries.

2.1.2 Languages for Representing Mathematics

Representing mathematical elements in the computer is more complex for humans than writing them down by hand. Basic algebraic operations can be easily written without any tool or particular language, but with the addition of more complex maths, the task becomes trickier. While most of the standard characters used in mathematics can be found in the Unicode Standard (e.g., the summation Σ sign is encoded as U+2211), it is far from convenient to use this set for writing mathematical expressions and formulas.

First, some text editors cannot encode/decode this type of encoding for mathematical objects and second, most general-use keyboards only contain keys for symbols for basic algebraic operations. Different representations have been designed from this need to create tools for easily/meaningfully writing mathematics. There are two primary resources available for writing mathematics: LaTeXand MathML.

Donald Knuth created the TeX system (which later became LaTeX) as a way to create a tool that allowed simple construction of mathematical formulae while looking

professional when printed. While LaTeX is used for all text types, typesetting mathematics is one of its greatest strengths [10].

While LaTeX focuses on creating documents, MathML[1] is designed to facilitate the use and reuse of mathematical and scientific content on the Web. Unlike LaTeX, which focuses only on presentation aspects, MathML can also encode mathematical content (i.e., considering the semantics of mathematical objects). This semantics functionality was later extended with OpenMath[2].

While both languages have advantages and disadvantages, when dealing with Deep Learning models, one needs to be concerned with the length of a sequence, given that several models have a limited input size. In Appendix A we show how to represent the quadratic formula $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$ using both MathML and LaTeX. When using the pre-trained tokeniser from BERT-base to tokenise both formats, we found that the LaTeX format is converted to 24 tokens while the MathML version totals 201 tokens. The difference in length is evident. We also include the tokenisation results to Appendix A.

Several tools are available to tokenise natural language text; however, the same is not valid for mathematical text. Often each token is separated by whitespaces, but sometimes we have composed tokens, such as the word "New York". Expressions and formulas can have different tokenisation realisations. For example, the equation $(x+1)*y=z$ could be tokenised at the level of variables, constants and operators (e.g.,(['(','x','+',..., 'y','=', 'z']) or at the level of expressions ([' $(x+1)*y$','=','z']).

As we will see in Chapter 3, there is no standard for representing mathematical notation for NLP models, but the two mentioned here are the most used.

2.2 Deep Learning Architectures

A language model is traditionally trained to predict the joint probability of sequences of words [43]. Simple statistical language models have limitations in scalability and fluency because of its sparse representation of language. Neural language models, based on representation learning, overcome this limitation by assigning a distributed vector for each word and then using a neural network to predict the next word, allowing the model to approximate words without being misled by rare or unknown values.

[1] https://www.w3.org/Math/
[2] https://openmath.org/

With large-scale training corpora, these models can learn how to model the joint probability of sentences while predicting the distributed combination of weights in continuous space, generating word embeddings or word representations (i.e., low-dimensional word vectors) as learned parameters of the model.

As shown in this section, recent models integrate contextual representations. Instead of assigning each word with a fixed vector, state-of-the-art language models use multi-layer neural networks combined with self attention mechanisms to calculate context sensitive representations for the words based on their context [52]. Given the improvements that Deep Learning architectures achieved in a diverse range of NLP tasks, we use deep learning models as a representational foundation to support the interpretation and inference over mathematical text. This section reviews some of the concepts on Deep Learning architectures which are used in this work.

2.2.1 Basic Concepts for Neural Networks

Neural Networks is the defining representation behind Deep Learning models. A *neural network* is an interconnected set of simple processing elements, units or nodes. The processing ability of the network is stored in structures called *neurons*, obtained by the process of learning from a set of training patterns [33]. In this section, we define essential components for understanding Deep Learning architectures.

Definition 11. The *abstract neuron* is the building block of any neural network. It is defined as a quadruple (x, w, Φ, y), where $x^\top = (x_0, x_1, \ldots, x_n)$ is the input vector, $w^\top = (w_0, w_1, \ldots, w_n)$ is the weights vector, with $x_0 = -1$ and $w_0 = b$, the bias, and Φ is an activation function that defines the outcome function $y = \Phi(x^\top w) = \Phi(n_i = w_i x_i)$ [16].

The neurons are optimised towards the prediction of target variable z by tuning the weights of the vector w such that the difference between y and z becomes small.

In order to learn a nonlinear target function, a neural network uses nonlinear activation functions. The ones relevant for this work are defined following.

A *sigmoid* activation function is used to approximate a step function. The *logistic function* (Figure 2.1), which maps real numbers into the unit interval $(0,1)$, with parameters $c > 0$ is defined as:

$$\sigma_c(x) = \sigma(c,x) = \frac{1}{1+e^{-cx}} \tag{2.1}$$

where the parameters $c > 0$ define the rate of change values from 0 to 1.

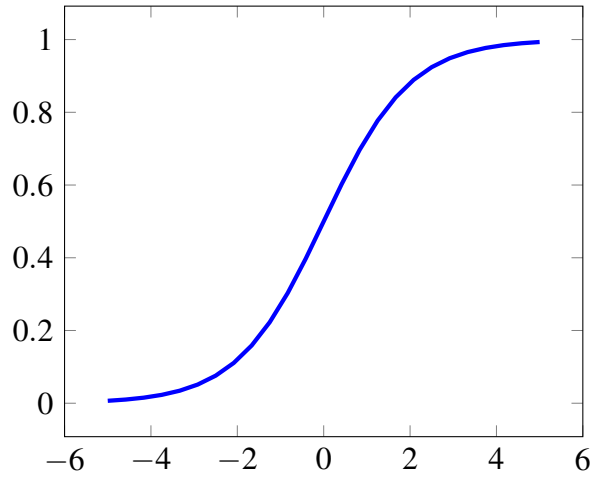


Figure 2.1: Logistic function $\sigma(x)$ with $c = 1$. This function maps the input to values between 0 and 1.

Another sigmoid function is the bipolar sigmoidal function, also known as the *hyperbolic tangent* (Figure 2.2). It maps real numbers into the interval $(-1, 1)$, computed as:

$$tanhx = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.2}$$

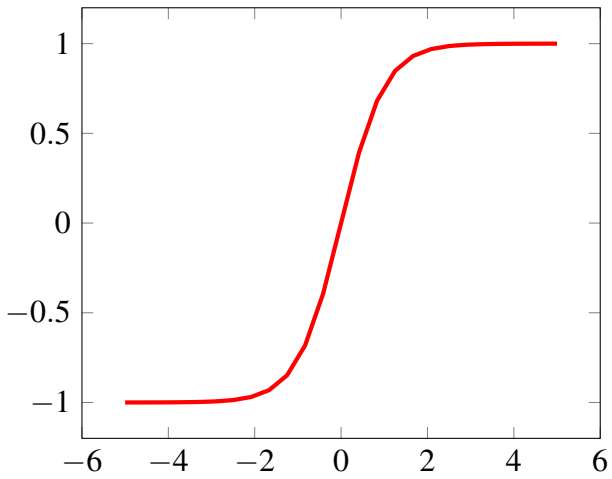


Figure 2.2: Hyperbolic tangent $tanh(x)$. This function maps the input to values between -1 and 1.

Another class of activation functions are the hockey-stick functions. While sigmoid functions have a shape similar to the letter S, this other class has a shape similar to a

hockey stick, as the same suggestion. One type of function that we use in this work is the *Rectified Linear Unit (ReLU)* (Figure 2.3) defined as:

$$ReLU(x) = max\{x, 0\} = \begin{cases} 0, \text{ if } x < 0 \\ x, \text{ if } if x \geq 0 \end{cases} \tag{2.3}$$

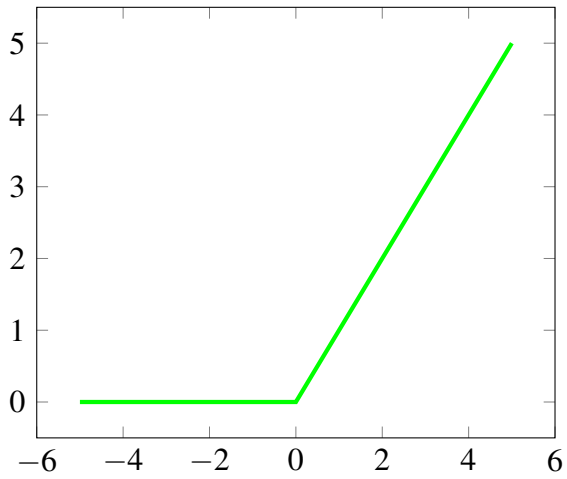


Figure 2.3: ReLU activation function. It behaves linearly for $x \geq 0$

ReLU activation function does not saturate (i.e., state in which a neuron mainly outputs values close to the asymptotic ends of the bounded activation function), unlike sigmoid functions and it tends to learn several times faster than a similar network with saturating activation functions.

A *layer* is a broad term that applies to a group of neurons collaborating at a specific depth within a neural network. The *input layer* takes the input and passes it on to the rest of the network. The *output layer* holds the result of the prediction. The *hidden layer* are the ones that are neither output nor input layers, and they perform the computations that assure good performance and complexity for neural networks. A neural network can have one or more hidden layers.

The *linear layer* is a classic type of layer, which is composed of single layers of linear neurons. It works by applying the following transformation:

$$y = x * W^{\top} + b \tag{2.4}$$

where x is the input to the layer, W is the weights vector and b is the bias.

Another notable architectural element in the definition of Deep Neural Network

models is the softmax layer. It is usually positioned before the output layer. This layer is particularly useful when we have a classification problem with two or more classes.

The standard softmax function $\sigma : R^K \rightarrow [0,1]^K$ for a classification problem with K classes is computed as:

$$\sigma(\mathbf{z}_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1,\ldots,K \text{ and } \mathbf{z} = (z_1,\ldots,z_K) \in^K . \tag{2.5}$$

This layer applies an exponential function to each element z_i of an input vector $\mathbf{z}$ and normalises these values by dividing by the sum of all these exponential terms. This layer will return the probability for each class in K, with values adding up to 1.

2.2.2 Paragraph Vector (`doc2vec`)

Deep Learning models often requires textual input to be represented as a fixed-length vector. A straightforward and efficient approach is simply using BOW to represent a document or sentence. However, this approach loses the order and context of the words, which plays an essential factor when modelling semantics.

Paragraph Vector [48] is an unsupervised model that aims to bridge this gap between the creation of efficient (in terms of inference and training time) and still semantically rich representations by learning continuous distributed vector representations textual fragments. This framework extends word2vec [56], and it is applicable to variable-length pieces of texts, from single sentences to large documents.

In order to learn distributed representations of words, a model can be trained in an unsupervised language modelling task (Figure 2.4).

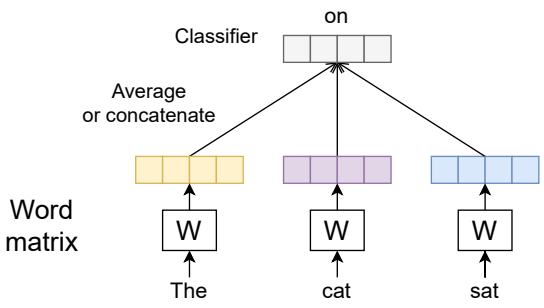


Figure 2.4: Example of the framework `word2vec`, used to learn vector representations for words. Figure adapted from Le and Mikolov [48].

Given a sequence of training words $w_1, w_2, \ldots, w_T$, the training objective is to maximize the average log probability:

$$\frac{1}{T}\sum_{t=k}^{T-k} logp(w_t|w_{t-k},\ldots,w_{t+k}) \tag{2.6}$$

This prediction task is usually performed via a softmax classifier, obtaining:

$$p(w_t|w_{t-k},\ldots,w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \tag{2.7}$$

where each y_i is the un-normalized log-probability for each output word i. It is computed as

$$y = b + Uh(w_{t-k},\ldots,w_{t+k};W) \tag{2.8}$$

where U, b are the Softmax parameters and h is constructed by concatenating or averaging the word vectors extracted from W, a matrix which maps each word to a unique vector.

Extending this idea, in the Paragraph Vector framework (Figure 2.5), each paragraph (or document) is mapped to a single vector, represented by a column in matrix D, and, again, every word is mapped in matrix W. The average of the paragraph vector and the words are averaged or concatenated to predict the next word in a context.

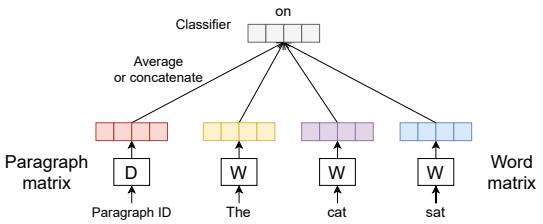


Figure 2.5: Example of the framework `doc2vec` (PV-DM), used to learn vector representations for words and vector representations for the paragraph. Figure adapted from Le and Mikolov [48].

This new paragraph token can be considered as a new word, which acts as a memory for each context considered inside a paragraph. This model is also known as Distributed Memory Model of Paragraph Vectors (PV-DM) due to this property. The contexts have a fixed length window, where the paragraph vector is shared across all contexts.

Another task for training Paragraph Vectors is teaching the model to predict words randomly sampled from the paragraph in the output. This variation is named Distributed Bag of Words (Figure 2.6) version of Paragraph Vector (PV-DBOW). In this

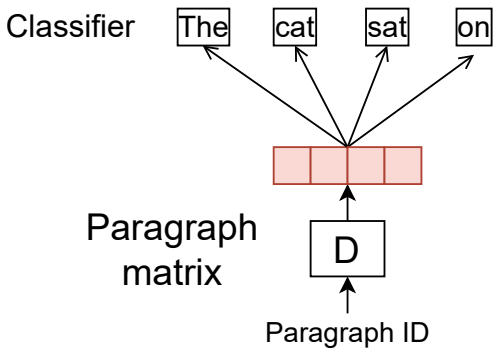


Figure 2.6: Paragraph Vector using a Distributed Bag of Words setting (PV-DBOW). The paragraph vector is trained to predict the words in a small window. Figure adapted from Le and Mikolov [48].

scenario, the derived models are smaller in size as only the softmax weights are being stored, without the requirements for averaging or concatenating vectors.

2.2.3 Long Short-Term Memory Networks

Recurrent neural networks (RNNs) are a family of neural networks for processing sequential data, one example of such architecture is Long Short-Term Memory networks (LSTMs) [38]. While RNNs are known to suffer from the *vanishing gradient* problem over long-distance dependencies, LSTMs have introduced new mechanisms to minimise this issue. The vanishing gradient problem happens when the influence of a given input on the hidden layer either decays (tending to zero) or increases exponentially (tending to infinity) as it cycles around the network's recurrent connections [30].

The LSTM architecture is composed of a set of recurrently connected memory blocks. These blocks are responsible for preserving the more distant input signals. Each block contains one or more self-connected memory cells and three multiplicative units [30].

Figure 2.7 illustrates one LSTM unit. Each unit takes in its previous cell state (C_{t-1}), previous hidden state (h_{t-1}) and the unit input element (x_t). The first component of the unit (1) is used to decide what information will be kept from h_{t-1}, with the use of a sigmoid layer:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.9}$$

The second step (2) determines what the cell will store in memory, using an *input gate layer* (a sigmoid layer) and a `tanh` layer to create a vector of new candidate values

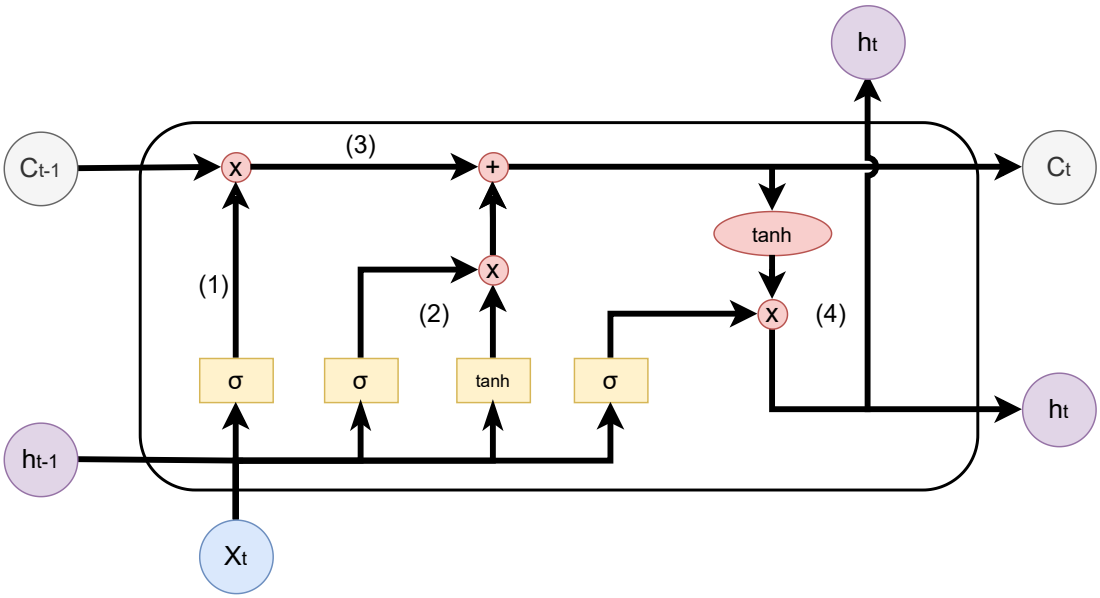


Figure 2.7: Diagram of an LSTM cell. Figure adapted from Olah [59].

$\tilde{C}_t$:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.10}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.11}$$

The new cell state C_t is updated (3) using the previous computations:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.12}$$

Finally, the output is computed (4) based on a filtered version of the obtained cell state:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.13}$$

$$h_t = o_t * tanh(C_t) \tag{2.14}$$

where W_f, W_i, W_C and W_o are the learnable weight parameters and b_f, b_i, b_C and b_o are the biases.

LSTMs can also be extended to process sequences in both directions; these architectures are called Bidirectional (Bi) LSTMs [76], where the recurrent layer of an LSTM is duplicated. The first layer receives the original input, while the second encodes its reverse. The output of those layers are combined (e.g., through concatenation)

to obtain the final output.

2.2.4 Transformers

Transformers [85] implement a encoder-decoder architecture, with the encoder taking as input a sequence $(x_1, \ldots, x_n)$ and mapping it to a continuous representation $\mathbf{z} = (z_1, \ldots, z_n)$. The decoder takes $\mathbf{z}$ and maps it to an output sequence $(y_1, \ldots, y_m)$. The **encoder** is composed of a stack of $N = 6$ layers. Each layer has a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. The output of each layer is `Layernorm(x + Sublayer(x))`, where `Sublayer(x)` is the function implemented by the sub-layer itself. The sub-layers used inside a Transformer are Feed Forward and Multi-Head Attention (Figure 2.8). Each layer also contains a fully connected feed-forward network, which transforms the input across all positions.

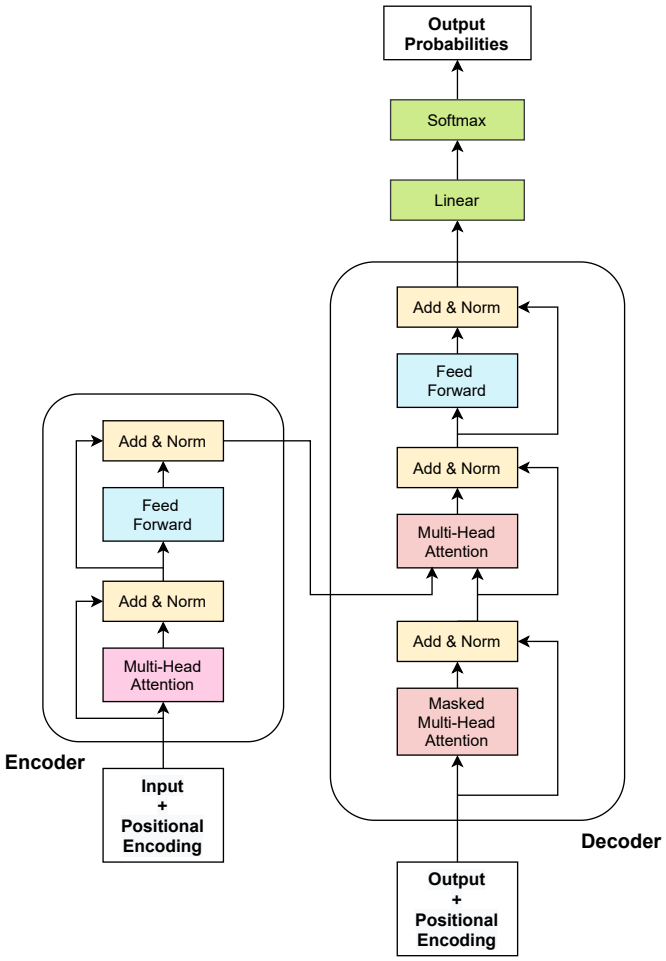


Figure 2.8: Diagram of the Encoder and Decoder of a Transformer model. Figure adapted from Vaswani et al. [85].

Since the model does not contain layers capable of capturing sequences, it makes use of *positional encodings*, which are added to the input embeddings, computed as follows:

$$\begin{aligned} PE(pos,2_i) &= sin(pos/10000^{2i/d_{model}}) \\ PE(pos,2_i+1) &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{2.15}$$

where pos is the position of the element and i is the dimension. The positional encodings have the same dimension d_{model}.

The self-attention used by Transformers is called *Scaled Dot-Product Attention*, which takes as input a set of queries Q and keys K of dimension d_k, and values V of dimension d_v:

$$Attention(Q,K,V) = softmax(\frac{QK^\top}{\sqrt{d_k}})V \tag{2.16}$$

The values of Q, K and V are obtained from a linear transformation over the input embeddings.

The attention is then expanded with the use of multi-head attention as:

$$\begin{aligned} MultiHead(Q,K,V) &= Concat(head_1,\ldots,head_h)W^O \\ \text{where } head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.17}$$

where the projections are parameters matrices $W_i^Q \in \mathbb{R}^{d_{model}\times d_k}$, $W_i^K \in \mathbb{R}^{d_{model}\times d_k}$, $W_i^V \in \mathbb{R}^{d_{model}\times d_v}$ and $W_i^O \in \mathbb{R}^{hd_v\times d_{model}}$.

The feed-forward layer consists of two linear transformations with a ReLU activation in between.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{2.18}$$

The **decoder** part is also composed of a stack of $N = 6$ layers. It also contains the same layers as the encoder, with the addition of a third sub-layer that performs multi-head attention over the output of the encoder stack. This self-attention sub-layer in the decoder stack is modified to prevent positions from attending to subsequent positions, using attention masks for the output. This ensures that predictions made for position i can depend only on the known outputs at positions $i-1$.

2.2.5 Pre-trained Language Representations

Transformers were used to develop a range of expressive language representation models, often refereed to as Pre-trained language representations, Large Language Models (LLM), or Transformer-based models. One example of such model is the Bidirectional Encoder Representations from Transformers (BERT) [25]. BERT is trained using two steps: *pre-training* and *fine-tuning*. During pre-training, the model is trained on unlabelled data over two different tasks, the *Masked Language Modeling* (MLM) task and the *Next Sentence Prediction (NSP)* task. In the Masked LM task, the model predicts a masked word in a sentence, replacing a random word in a sentence for the `[MASK]` token or another random word.

In the second pre-training task, NSP, the model is trained on a binary classification task of predicting if sentence B follows sentence A. A set of negative examples sentences is randomly selected, such that 50% of the examples are correct and the rest are negative examples.

This unsupervised training process can be applied at large-scale corpora, and allows the induction of a model which can capture syntactic and semantic features [67]. BERT is training on BooksCorpus (800M words) and English Wikipedia (2,500M words).

BERT uses WordPiece embeddings to represent inputs. WordPiece [95] is a subword-based tokenisation algorithm. It works by training a language model on a base vocabulary (starting with all the characters), then it selects the pair of elements in the vocabulary with the highest likelihood, adding this new pair to the vocabulary. The language model is trained again using this new vocabulary, repeating this process up to a limit in vocabulary size or when a predefined likelihood threshold of new combinations is reached.

Using BERT as a basis, other models have been proposed. SciBERT [11] is a BERT model trained on scientific text. SciBERT is trained following the same process as BERT, but using papers from the corpus of Semantic Scholar[3] as training data. Changing the training data allows the model to perform better than BERT on various tasks over scientific domain, such as sentence classification and sequence tagging. On a similar fashion, MathBERT [60] was trained over a large mathematical corpus ranging from pre-kindergarten and high-school, to college graduate level mathematical content. The model is tested over three tasks: prediction of knowledge component, auto-grading open-ended QA, and knowledge tracing, obtaining higher scores than by testing using

[3] https://www.semanticscholar.org/

plain BERT.

The training procedure performed for BERT has also been improved in different Transformer-based models. RoBERTa [51] removes the NSP training objective, finding that removing the NSP loss slightly improves downstream task performance. RoBERTa also differs from BERT in specific hyperparameters, while BERT is trained for 1M steps with a batch size of 256 sequences, RoBERTa is trained with 125 steps of 2K sequences and 31K steps with 8k sequences of batch size. RoBERTa also improves from a single static mask to a dynamic masking in which different masking is performed at every epoch the data passes through the network. This model obtains state-of-the-art results for almost all of the GLUE [88] tasks.

MPNet [78] is another model that modifies the pretraining procedure of BERT. While BERT uses only Masked Language Modeling, MPNet is pretrained using a combination of Masked and Permuted Language Modeling. Permutation language models are trained to predict one token given preceding context like traditional language model, but instead of predicting the tokens in sequential order, it predicts tokens in a random order. This technique allows the models to capture bidirectional dependencies.

ALBERT [47] modified BERT with two parameter reduction techniques to obtain a smaller and scalable model. The first one is a factorised embedding parameterization which decomposes the vocabulary embedding matrix into two small matrices, making it easier to grow the hidden size without increasing the parameter size of the vocabulary embeddings. The second technique is cross-layer parameter sharing which prevents the parameter from growing with the depth of the network.

2.2.6 Sentence-based Representations

Sentence-BERT (SBERT) or Sentence-Transformer [65] is a modification of the pretrained Transformer-based models using Siamese Network structures [45] to obtain sentence-level embeddings. In this work, we use this type of model for both obtaining sentence embeddings for improving our features and also for obtaining representations to support information retrieval tasks.

A Siamese Neural Network [45] is a class of neural network architectures containing two or more identical subnetworks, i.e., networks with the same configuration with the same parameters and weights. The parameters updated are shared across all these subnetworks. In SBERT this is used to find the similarity of the inputs by comparing its feature vectors.

SBERT adds a pooling operation to the output of Transformer-based models, deriving a fixed-sized sentence embedding. There are three main pooling strategies: Using the output of the `[CLS]` token, computing the mean of all output vectors, and computing the max of all output vectors. The type of pooling used will depend on the usage and model initially for pre-training. This technique then fine-tunes the pre-trained language model on a semantic similarity tasks, such as paraphrasing and natural language inference.

Model	**Base Model**	**Model Size**	**Short Description**
all-mpnet-base-v2	MPNet-base	420 MB	All-round model tuned for many use-cases.
multi-qa-mpnet-base-dot-v1	MPNet-base	420 MB	This model was tuned for semantic search: Given a query/question, check if can find relevant passages.
paraphrase-multilingual-mpnet-base-v2	MPNet-base and xlm-roberta-base	970 MB	Trained using as teacher-student distillation. Teacher: paraphrase-mpnet-base-v2; Student: xlm-roberta-base
paraphrase-albert-small-v2	ALBERT-small	43 MB	Smaller model, optimized for speed.
scibert-nli	SciBERT	420 MB	Model fine-tuned for scientific natural language inference
roberta-base-nli-mean-tokens	ROBERTA-base	480 MB	Model fine-tuned for general language inference
nli-bert-base	BERT-base	420 MB	Model fine-tuned for general language inference

Table 2.1: Sentence-Transformer pre-trained models used in this work.

Table 2.1 presents the different SBERT models that are used in this work. This table presents a brief description of the model together with the size and base model used for the finetuning.

2.3 Information Retrieval

Information retrieval (IR) is the retrieval of material of an unstructured nature (usually text) that fulfils an information need from within extensive collections, such as a knowledge base [77]. In this work, we rely on a technique called BM25 which is described in this section.

Okapi BM25, or simply BM25, is a ranking function employed by search engines to evaluate the relevance of records to a given search query [66].

BM25 works by ranking a set of documents based on the query terms which appears in each document. Given a query Q, with tokens $q_1, \ldots, q_n$, the BM25 score of a document D is:

$$score(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)} \tag{2.19}$$

where $f(q_i,D)$ is the q_i term's frequency in the document D, $|D|$ is the length of the document in the number of tokens, $avgdl$ is the average document length, considering all documents in the knowledge base, k_1 and b are parameters chosen to optimise the score, usually default values are used. $\text{IDF}(q_i)$ is the inverse document frequency weight of the query term q_i. The term's frequency for q_i inside a document D is computed as:

$$\text{tf}(q_i,D) = \frac{f_{q_i,D}}{\sum_{q_j \in D}^{n} f_{q_j,D}} \tag{2.20}$$

where $f_{q_i,D}$ is the number of times the token q_i appears inside the document. This value is divided by the frequency of all other tokens inside document D to create a frequency relative to the other tokens in the document.

The IDF score is computed as:

$$\text{IDF}(q_i) = \ln\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right) \tag{2.21}$$

where N is the total number of documents in the knowledge base and $n(q_i)$ is the number of documents containing the token q_i in that knowledge base.

2.4 Summary

This chapter defines the building blocks of the mathematical text, highlighting the main difference between natural language and mathematical language. While natural language tends to allow space for metaphors, ambiguities, and subjective interpretation, mathematical language needs to have a precise vocabulary and follow well-defined rules, and each statement should have a single and accurate interpretation. In this chapter, we also introduce different Deep Learning architectures that have been successfully applied in different NLP tasks, but as we will show in the next chapter, they still are widely unexplored in the field of MathLP.

This work presents the performance for MathLP of the different Deep Learning models introduced here, showing how it compares with information retrieval baselines. Across the chapters of this thesis, we propose, detail and evaluate different techniques to improve these models to encode the particularities of the mathematical text. The next

chapter will describe what MathLP tasks have been investigated in the past, together with different proposed approaches, identifying the gaps that exist in this field.

Chapter 3

Literature Review

In this work, we consider MathLP as a subfield of NLP which focus on Mathematical Text. In order to better understand the field and identify its gaps, we explore previous work that has been performed in the area. In comparison with other NLP research fields, such as Question Answering and Machine Translation, the field of MathLP is still unexplored. As discussed previously, the corpora of mathematical text are constantly growing, rendering this area a suitable candidate for NLP methods to aid in the understanding of this large-scale corpora. This chapter provides a definition of MathLP together with tasks the tasks explored previously in this field: Mathematical Information Retrieval, Mathematical Element Typing, Mathematical Problems, Mathematical Document Categorisation, Plagiarism Detection, Testing Large Language Models for Mathematics, together with other tasks with less traction.

3.1 Mathematical Language Processing (MathLP)

The field of MathLP can be defined as a subfield of NLP, which concentrates on methodologies for addressing learning, understanding, and producing Mathematical Text. As with other subareas in NLP, the field has a bias towards empirical methodologies and is largely defined by a set of supporting tasks and supporting empirical analysis frameworks. A summary of the previously proposed MathLP tasks can be found in Table 3.1.

Table 3.2 presents a summary of all the recent models and supporting methodologies used to address contemporary MathLP tasks. These are classified in terms of the type of learning, the dataset used, metrics for evaluation, the format of the mathematical notation and the type of structure used for mathematical expressions. The type of

Task	Type	Input	Output
Mathematical Information Retrieval	Retrieval	Formula Query	Similar Formulas
Mathematical Element Typing	Inference	Sentence/passage with variable/mathematical expression	Type of variable or mathematical expression
Mathematical Problem Solving	Generation	Passage/Sentence containing a mathematical problem	Answer for the problem
Mathematical Document Categorisation	Inference	Document containing mathematical notation	Category of the document (e.g. mathematical area)
Plagiarism Detection	Retrieval	Scientific work containing mathematical notation	Similar works that could potentially flag a case of plagiarism
Math Summarisation	Generation	Question about mathematical concepts	Summary of the question
Reference Inference	Generation/Retrieval	Mathematical statement	References found in the proof for the mathematical statement
Statement-Proof Matching	Retrieval	Mathematical Statement	List of possible proofs for the mathematical statement

Table 3.1: Tasks that were previously proposed in the field of Natural Mathematical Language Processing.

structure refers to how the expressions are given as input to the different models. For example, an expression can be considered a Sequential structure, or it can also be seen as a tree composed of sub-expressions. In the following sections, we will describe the tasks that have been previously investigated in MathLP, together with the approaches that address such tasks. We only discuss works involving text that combines both Natural Language and Mathematics (i.e., content written using Mathematical Language, instead of only Mathematics corpora) and limit our analysis to more recent work (from 2017 onwards). For example, this review does not include works involving solely mathematical formulas/expressions, such as the retrieval of formulae in isolation or finding the solutions for a certain equation.

3.2 Mathematical Information Retrieval

Mathematical information retrieval (Figure 3.1) requires specific inference skills since traditional text retrieval systems do not encode the unique behaviour mathematical expressions [22].

While several applications for MIR with a focus on the retrieval of mathematical expressions can be found [1], in this work, we are interested in scenarios that handle

Work	Learning	Approach	Dataset	Metrics	Maths Format	Expression
Mathematical Information Retrieval						
Kristianto, Topić, and Aizawa [46]	UNS	String Matching + TF-IDF	NTCIR-11 Math-2	mAP, P@N	MathML	Seq
Dadure, Pakray, and Bandyopadhyay [21]	UNS	Bit Position Information	NTCIR-11 Math-2	P@N	MathML	Seq
Yasunaga and Lafferty [97]	S	Topic Modelling + RNN	ContextEq-400	Perplexity	LaTeX	Seq
Mathematical Element Typing						
Schubotz et al. [75]	S	SVM	NTCIR MathIR	P,R,F1	LaTeX	Seq
Stathopoulos et al. [80]	S	Bi-LSTM	VTDS	P,R,F1	LaTeX	Seq
Lin et al. [50]	S	Naive Bayesian Inference	NTCIR Math Understanding	P,R,F1	LaTeX	Seq
Alexeeva et al. [4]	UNS	Rule-based approach	Math-Align	P,R,F1	LaTeX	Seq
Yuan et al. [98]	S + UNS	TRG + ILP	NTCIR MathIR	T-means	LaTeX	Seq
Mathematical Problem Solving						
Saxton et al. [70]	S	Transformers	Mathematics	Acc	Simple Maths	Seq
Schlag et al. [72]	S	TP-Transformer	Mathematics	Acc	Simple Maths	Graph
Huang et al. [39]	S	Neural Solver	HSMP	Acc + BLEU + ROUGE	Simple Maths	Graph
Hendrycks et al. [35]	S	GPT-3	MATH	Acc	LaTeX	Seq
Mathematical Document Categorisation						
Suzuki and Fujii [82]	S	Structural Kernel Method	MDC MO + MDC arXiv	P+R+F1	MathML	Tree
Scharpf et al. [71]	S + UNS	Dov2Vec + MLP	MDC SML + MDC NTCIR	Acc + CP	MathML	Seq
Wankerl, Götz, and Hotho [92]	S	ResNet	SE Maths	P+R+F1	LaTeX	Seq
Plagiarism Detection						
Meuschke et al. [54]	UNS	BoW	NTCIR-11 MathIR	MRR	MathML	Seq
Meuschke et al. [55]	UNS	BoW	NTCIR-11 MathIR	MRR	MathML	Seq
Math Summarisation						
Yuan et al. [99]	S	Self-attention	EXEQ-300k + OFEQ-10k	ROUGE + BLEU + METEOR	LaTeX	Seq
Reference Inference						
Welleck et al. [94]	S	LLM for Generation and Retrieval	NATURALPROOFS	mAP + R+ BLEU	LaTeX	Seq
Statement-Proof Matching						
Coavoux and Cohen [18]	S	Self-Attention	SPM	MRR + Acc	MathML	Seq

Table 3.2: Summary of different methodologies for addressing MathLP tasks.

the combination of mathematical notation and the natural language.

Kristianto, Topić, and Aizawa [46] propose the use of dependency relationships between each mathematical expression and its constituent sub-expressions in a document to enrich the textual information for each expression for MIR. This work encodes this dependency relationship as a graph, building using a rule-based heuristic technique that has limited scalability. The authors examine the impact of this representation in a MIR setting, finding that significantly better precision can be obtained using dependency-based enriched textual information rather than using only the math expressions' textual information.

Dadure, Pakray, and Bandyopadhyay [21] proposed a formula embedding and generalisation approach for MIR. This generalisation converts similar formulae into a unified form (e.g., representing both $x+y$ and $a+b$ as $var1+var2$). In this work, documents are pre-processed, and the formulas are extracted along with their surrounding text. The formulas, surrounding text and query are encoded using a binary vector and indexed by the indexer with their surrounding text.

Yasunaga and Lafferty [97] introduce a novel topic model that jointly generates mathematical equations and their surrounding text (TopicEq), where the context is

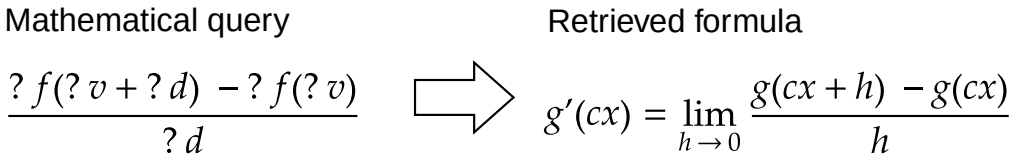


Figure 3.1: Example of a mathematical query and the expected retrieved formula. Example obtained from [100].

generated from a mixture of latent topics, and the equation is generated by an RNN that depends on the latent topic activations. The results show that this joint model significantly outperforms existing topic models and equation models for scientific texts.

3.3 Mathematical Element Typing

Mathematical Element typing (Figure 3.3) is the task of assigning types to variables and expressions inside the mathematical text [80]. Different methodologies for this task have been proposed and are outlined in this section.

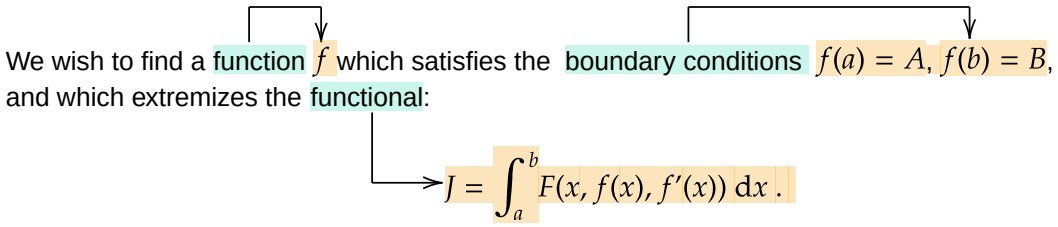


Figure 3.2: Example of the mathematical element typing task. The types and mathematical elements are marked in this sentence.

Schubotz et al. [75] propose the use of Support Vector Machines to decide if an identifier (i.e., the definiens) belongs to a mathematical element (i.e., the definiendum) or not. Each mathematical element is compared with all possible identifiers, and the model is trained on all these possible combinations. The authors found that Support Vector Machines perform better for this task than rule-based pattern matching and statistical scoring of identifier-definiens pairs.

A statistical analysis of mathematical corpora has shown that most of the variables are declared inside the same sentence, where it is first mentioned [32]. Previous work [80] has leveraged this to create a new setting for the variable typing task, where

a comparison between possible types and variables is only made at the sentence level. Stathopoulos et al. [80] presents three different models for addressing this problem, framing it as a binary classification between a variable and a type. The models proposed are an SVM model using features that are type and variable-centric, a ConvNet using pre-trained embeddings to represent the input tokens and a Bidirectional LSTM model that takes as input the complete sentence along with the pairs for classification.

The task can be facilitated by adding additional steps to the inference process. Lin et al. [50] proposes a three-level framework composed of a customised POS tagger, extraction of NP-chunks as potential definiendums for mathematical elements, and finally, a predictive decision procedure that determines whether an NP refers to a mathematical element or not. The prediction is performed using Naive Bayesian Inference to identify such references based on distance, word stem, and POS tag.

Similarly, Alexeeva et al. [4] propose a rule-based approach for extracting identifier descriptions that are readily interpretable, maintainable, and domain-agnostic. However, the authors do not compare their results with baselines, making it challenging to assess the model in a broader context.

A different methodology for this task is proposed in Yuan et al. [98]. This work constructs descriptions for mathematical expressions automatically, proposing a hybrid model which contains two modules: Selector and Summarizer. In the Selector, a Topic Relation Graph (TRG) is proposed to obtain the relevant documents which encapsulate the citations around math expressions. TRG is a graph built according to the citations between expressions. In the Summarizer, a summarisation model under the Integer Linear Programming (ILP) framework is proposed. This module constructs the final description with the help of a timeline that is extracted from TRG. The results show that the proposed method is appropriate for this task and outperforms the baselines.

The main datasets used for the task of Mathematical Element typing are:

- NTCIR-11 Math Retrieval Wikipedia (Subset) [74]: The dataset contains 100 formulae taken from 100 unique Wikipedia articles and contains 310 identifiers.

- NTCIR-10 Math Understanding task [74]: Dataset containing 45 manually annotated papers with natural language descriptions of mathematical formulae in each document for their semantic interpretation.

- Variable Typing Dataset (VTDS): The first large-scale dataset for typing elements inside mathematical text. It contains manually annotated sentences with

variables and types, with a total of 7,803 sentences and 3,501 variable and types pairs.

- MathAlign: Dataset consisting of over 700 identifiers and their descriptions and the original span of LaTeX used to generate the identifier.

3.4 Mathematical Document Categorisation

MDC is the task of classifying mathematical documents, such as scientific papers and articles, into different mathematical areas (e.g., probability theory and set theory). Such a task is essential to aid users in the process of searching over large digital libraries [82]. Past research has explored similar tasks in natural language, although very few works address the issue of classifying documents containing mathematical notation.

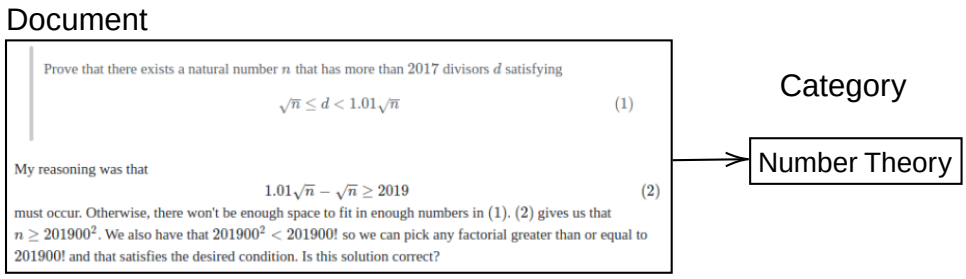


Figure 3.3: Example of the Mathematical Document Categorisation task.

Mathematical notation plays a significant role in determining the category of a document. For example, a document containing the letter G as part of the mathematical notation will likely link to *Graph Theory* or *Group Theory*, requiring the understanding and disambiguation of mathematical notation. Therefore, how to utilise the information of mathematical notation is a vital matter for MDC.

Past research has leveraged the representation of mathematical elements for better categorisation of mathematical documents. Some works focus on creating distinct representations for text and mathematical elements in order to obtain a measure of similarity between different documents [82, 71]. In contrast, research also found it possible to classify documents only by looking at the mathematical elements [92]. These results show the expressive power that expressions and equations can have inside a text

since, in all cases, there was an improvement when creating specific representations for mathematical elements.

The mathematical elements can be encoded using different techniques. Suzuki and Fujii [82] proposed a tree-based representation, where each expression, written in MathML, is converted to a tree of sub-expressions and then encoded using tree kernel similarity. Other works [71, 92] chose to represent expressions as a linear structure and were still able to achieve good performance regardless. Since expressions can also be represented as images when rendering the LaTeX representation, image encoding techniques can also be used.

Wankerl, Götz, and Hotho [92] compared different encoders for both image and text and found that ResNet could achieve the best results. The authors claim that the image representation is better at capturing the non-linear nature of expressions. Scharpf et al. [71] investigated the correlation between mathematical notation and natural language, determining whether if two documents are similar in their encoding, they are also similar in their math encoding. Such correlations were relatively low, hinting that text and math should be encoded as different features.

A quantitative analysis between the different approaches was not possible, given that each of the mentioned works opted for using different datasets. The datasets used for MDC are:

- MDC MathOverflow (MO) [82]: Dataset extracted from the MathOverflow community, composed of 3,339 documents, with categories based on the tags assigned by users. A total of 15 categories were selected.
- MDC arXiv [82]: Dataset extracted from arXiv, with 23 selected categories from the manually assigned arXiv categories. A total of 37,735 papers were selected, with roughly 100 to 2,000 papers in each category.
- Subset of SigMathLing arXMLiv-08-2018 (MDC SML): Dataset also extracted from arXiv, but containing a total of 4,900 selected documents spread across 14 different categories.
- Subset of NTCIR-11/12 MathIR arXiv [2, 100] (MDC NTCIR): Dataset extracted from arXiv by NTCIR, with a total of 3,500 sections and 1,400 abstracts extracted from papers.
- Stack Exchange Mathematics (SE Maths) [92]: Data extracted from the Mathematics website's content, which is part of the large Stack Exchange network.

It consists of questions concerning the field of mathematics and corresponding answers. Each question is labelled with one or more tags, with three different datasets: (i) with 10 tags and 332,892 questions, (ii) 50 tags and 573,445, (iii) with 100 tags and 619.285 questions.

All found datasets contain a similar style of text, with different complexity of mathematical notation.

In order to classify the documents into the correct categories, Suzuki and Fujii [82] proposes a classification method using structural kernel methods to generate features in tree-structure objects automatically. A tuple $d = (T, v)$ represents each document, where T are the MathML trees of expressions and v are the vectors of the text, encoded via BoW. The similarity between documents is given by a tree kernel similarity between expressions and the kernel over feature vectors of the text. Scharpf et al. [71] encodes both text and maths using doc2vec and TF-IDF for classification and clustering to find the best combination that obtains the best performance, concluding that Doc2Vec representations outperform TF-IDF encodings and that the best classification algorithm is an MLP. Wankerl, Götz, and Hotho [92] uses pre-trained language models and ResNet for classification.

3.5 Mathematical Problems

A large proportion of the NLP literature have focused on math word problems, i.e., mathematical problems written in plain English, usually without any symbolic mathematical notation. One example of such a problem would be: “There are three bumper cars. Each car holds two people. How many people altogether?”. Despite their reference to basic arithmetic operations, these are still open problems [101].

In this work, we consider for this literature review a smaller subset of mathematical problems (Table 3.3), specifically the ones that involve more complex mathematical notation, which require the integration of natural language and mathematical expressions. This problem takes complex mathematical questions as input and is required to generate the correct output. This particular type of problem remains generally unexplored.

The datasets available for complex Mathematical Problems are:

- Mathematics [70]: A synthetic dataset with mathematical problems on different mathematical areas such as Calculus and Probability. Most questions involve evaluating one or more randomly generated mathematical objects.

Question	Solve -42*r + 27*c = -1167 and 130*r + 4*c = 372 for r.
Answer	4
Question	Let x(g) = 9*g + 1. Let q(c) = 2*c + 1. Let f(i) = 3*i - 39. Let w(j) = q(x(j)). Calculate f(w(a))
Answer	54*a - 30

Table 3.3: Example of two mathematical problems with answers. Example obtained from [70].

- MATH [35]: A dataset of 12,500 challenging competition mathematics problems. Each problem in MATH has a step-by-step solution for teaching models to generate answer derivations and explanations.
- High School Math Problems (HSMP) [39]: Dataset collected with mathematical problems for senior high school students from their exercises. Contains a total of 31,500 problems with mathematical notation taking take nearly 69% of the whole problem tokens on average.

Saxton et al. [70] investigate two classes of models that have demonstrated themselves to be state-of-the-art on sequence-to-sequence problems: recurrent neural architectures and transformers architecture. The results have shown that question types that required finding the place value in a number and rounding decimals and integers could achieve almost perfect results, while questions involving Number Theory and detecting primality were the most challenging ones. From the tested approaches, Transformers were able the achieve the highest scores. With larger Transformers, the results are even more remarkable [36].

Being able to understand the relations between mathematical tokens is fundamental for solving this type of problem. Transformers have been extended [72] to learn to encode structural relations using Tensor-Product Representations explicitly, learning to cluster symbol representations based on their structural position and relation to other symbols. With this technique, the model can learn different roles for each token in the text, distinguishing between numbers in the numerator and denominator. Huang et al. [39] proposes enhancing formula structures, with formulas being represented as a formula graph network. Specifically, the formulas are first represented as a TEX dependency graph and then extended using a formula graph network (FGN) to capture its mathematical relations. This work also found that their proposed approach outperformed different Transformers and seq2seq models.

One important aspect of explainability in this task is understanding how a model

generates a certain answer. Hendrycks et al. [35] tests different models for generating full step-by-step solutions. The results on pre-trained language models show that accuracy remains relatively low, indicating that the task is still open and remains challenging.

3.6 Plagiarism Detection for Mathematical Content

Academic plagiarism is the use of ideas, concepts, words, or structures without appropriately acknowledging the source to benefit in a setting where originality is expected [54]. Previous research has explored the comparison of different documents through mathematical notation for detecting plagiarism. This task involves retrieving similar documents given a query document.

Meuschke et al. [54] extracts features from mathematical notation (i.e., identifiers, numbers and operators) using frequency-based representations. This work uses the absolute difference of the occurrence frequencies to compute the similarity between documents. The authors confirm the potential of analysing mathematics to identify possibly suspicious documents independent of literally matching text.

Meuschke et al. [55] extended the previous work to also consider similarity in terms of citations found in different documents. Despite the promising results, both works rely on only ten cases of retracted papers as the query documents, which does not confirm the generalisation of the approach to other settings. The pool of documents for comparison is obtained from the NTCIR-11 MathIR Task dataset [2], which contains a total of 105,120 scientific papers.

3.7 Testing Large Language Models for Mathematics

Given the popularity of pre-trained language models such as BERT, there has been some investment in testing what type of mathematical knowledge such models can understand. Wallace et al. [87] tested different NLP models' ability to understand and work with numbers and found that a degree of numeracy is naturally present in standard embeddings. Older models such as GloVe and word2vec seems capable of encoding numbers up to 1,000 (one thousand), while BERT seems less precise given the subword units it uses. Such results were reinforced by a different study [83], showing that the default subword segmentation used for words is suboptimal for numbers and that there is still no consensus on how to solve numeracy holistically.

Ryskina and Knight [68] presents probing for the embeddings of integers for mathematical knowledge and found that by learning the representations from mathematical sequence data, it is possible to improve the embeddings of numerical values significantly. However, it is still not obvious how to integrate these embeddings with larger pre-trained models.

Mathematical problem solving with Pre-trained language models has also been put into the spotlight. [62] investigated if BERT possesses problem-solving mathematical abilities and found that the model decreases in performance depending on the order of the provided alternatives, showing a lack of stability in inferring the correct result.

MathBERT [60] was proposed as an endeavour to solve some of the mentioned issues, created by pre-training BERT-base on a sizeable mathematical corpus ranging from pre-kindergarten to college graduate level mathematical content. The model was tested on different mathematical tasks (prediction of knowledge component, autograding open-ended Q&A, and knowledge tracing), and the results demonstrate the superiority of MathBERT over BERT-base.

3.8 Other Tasks

While other MathLP tasks have been proposed, some still need to gain some traction.

Yuan et al. [99] proposes a task for generating a concise math headline from a detailed math question. This task is a particular type of summarisation, with the generation of a short sequence of words and preserving the essential meaning of a math question document. They introduce two new datasets, EXEQ-300k and OFEQ-10k, with 346,202 and 13,408 questions pairs, respectively. This work proposes a generative model based on multi-head attention blocks, and they found that enriching the representation of mathematical elements achieves better results.

As highlighted by Welleck et al. [94], there is a gap in terms of large-scale benchmarks for tasks involving MathLP. In this thesis, we address this gap, which was later further explored by Welleck et al. [94]. The authors extend the Natural Language Premise Selection task defined in this thesis (Chapter 4) and propose two new tasks: reference retrieval and reference generation. In both cases, the task is designed to infer correct premises given a mathematical statement. The difference between tasks is on the inference method, either with retrieval or with generation. A new dataset is realised (NATURALPROOFS) consisting of 32k theorem statements and proofs, 14k definitions, and 2k other types of pages (e.g. axioms, corollaries).

Coavoux and Cohen [18] introduces a task consisting in assigning a proof to a given mathematical statement. A new dataset, Statement Proof Matching dataset (SPM), is proposed for the task, with over 180k statement-proof pairs extracted from mathematical research articles. They introduce a self-attention-based model that outperforms BoW baselines.

3.9 Discussion

From Table 3.2, we can observe that the combination of mathematics and NLP is still widely unexplored, particularly when contrasting with more popular fields such as general machine comprehension and question answering. Researchers have proposed various relevant tasks for the field; however, some are still waiting for follow-up work (e.g. Math Summarisation task). The gap is even greater if we consider the use of Deep Learning Techniques in the field.

In terms of data availability, there is still a need for large-scale datasets. Tasks such as Mathematical Element typing and Plagiarism Detection still rely on small datasets, creating a problematic scenario to show the generalisation of approaches to a broader scenario. We can see that each new work proposes a new dataset for some of the tasks instead of comparing it with previous ones, likely due to the unavailability or challenging access of previous datasets. In order to solve such an issue, the field needs to promote the creation of better datasets that are easier to access. **RQ1** (*How to support the evaluation of models for the interpretation and inference over mathematical text (regarding inference tasks, evaluation measures and supporting datasets)?*) is designed to address such a gap: we propose a new large-scale dataset, together with a novel task, creating a new benchmark for MathLP. This novel dataset allows for the study of different properties of the mathematical text, such as the multi-modality of the discourse and the dependency relationship that exists between statements. We also present the performance of different deep learning NLP methods for this new task compared with traditional baselines. We make the dataset available and the dataset construction process transparent in order to allow for easy data reuse and reproducibility. Our dataset and task were later extended for novel MathLP scenarios [94].

Previous work [71] suggested no correlation between the representation of mathematical and natural language tokens. This finding indicates that, in a neural scenario, both types of elements should be encoded into different semantic spaces. The exploration of this phenomenon is still a gap in this field, further investigation is required

for finding the best way to encode and combine these two different types of modalities of elements and how this can affect the performance on different tasks. We address this gap in both **RQ2** (*Can we improve the performance of deep learning models for inference over mathematical text by generating specialised encodings for mathematical natural language elements?*) and **RQ3** (*How to encode the semantics of variables at sentence-level representations?*). In order to answer **RQ2**, we investigate how to improve mathematical language inference through the design of a method capable of creating separate encodings for mathematical and natural language elements. After creating these separate encodings, we also explore how to harmonise both encodings into a single representation. For **RQ3**, we consider the separate encodings of more granular mathematical elements: variables. Variables have the unique property of having their meaning fully dependent on the context they are inserted. In order to encode the semantics of variables inside the text, we further explore **RQ3**, proposing a novel method that improves the representation of variables in pre-trained language models.

As highlighted previously, Mathematics by nature is a highly connected field, where each theorem is built on top of definitions, axioms, lemmas and other theorems. When performing inference over mathematical text, one needs to also consider all the background mathematics that each mathematical statement is dependent on. From this Literature Review, we could conclude that this relationship structure between statements has yet to be leveraged. This is a research gap that has not been previously addressed in this field, which we explore in **RQ4** (*Can the dependencies between mathematical statements provide a simplified representation of mathematical arguments?*) and **RQ5** (*How to encode the inter-statement dependency structure to support inference for the natural language premise selection task?*). For **RQ4**, we present the performance of traditional information retrieval pipelines and contrast with our novel methodology, which retrieves statements not only based on lexical and semantic similarity but also considering the relationship structure between statements. **RQ5** expands on **RQ4**, exploring how to encode this relationship between conjectures-premises using Graph Neural Networks. While in **RQ4** the focus was on the investigation of how to improve traditional retrieval models with the addition of structural similarity, **RQ5** looks into the design of a new method that encodes this relationship structure in a latent semantic space.

Chapter 4

Natural Language Premise Selection

One of the gaps identified in the Mathematical Language Processing research are datasets which reflect the complexity of the mathematical inference/argumentation as expressed in mathematical textual discourse. This chapter proposes a new task for this field and a dataset to support its evaluation. Along with the dataset, this chapter also presents the performance of different Information Retrieval baselines, comparing Bag-Of-Words based approaches with contemporary neural language models (e.g. transformer-based). In this chapter, we provide a first-line analysis on the performance of state-of-the-art models in performing inference over mathematical text, connecting with our first research question **RQ1**: *How to support the evaluation of models for the interpretation and inference over mathematical text (regarding inference tasks, evaluation measures and supporting datasets)*?.

4.1 Motivation

As described in Chapter 3, different tasks have explored the understanding and automatic processing of mathematical text. The most explored task is the Mathematical Knowledge Retrieval [2]. Aspects of the mathematical discourse related to the embedding and representation of mathematical expressions and encoding of relationship between statements still need further exploration, with a lack of datasets for evaluating these properties.

In order to address these challenges, we propose a new evaluation task which is an integral part of mathematical inference, together with a dataset for evaluating different approaches for this new task. The task proposed by this work is the Natural Language Premise Selection (NLPS), inspired by the field of automated theorem proving. This

new task can also be referred to as *Informal Premise Selection* or *Premise Selection for Natural Language*.

NLPS takes as input mathematical text, written in natural language and outputs relevant mathematical statements that could support an end-user finding proof for that mathematical text. The premises are composed of supporting definitions and propositions that can act as explanations for the proof process.

For example, the famous *Fermat's Little Theorem* [93] has different possible proofs, one of them using the *Euclid's Lemma*. In this example, Euclid's Lemma would be considered helpful for a human trying to prove Fermat's Little Theorem; therefore, it is a premise for the conjecture that Fermat's Little Theorem presents.

In order to evaluate this task, we propose a new dataset: PS-ProofWiki (Premise Selection-ProofWiki), using as a basis the human-curated website ProofWiki[1]. This dataset opens possibilities of applications for the premise selection task and for evaluating semantic representations for mathematical discourse (including embeddings), textual entailment for mathematics, and natural language inference in the context of mathematical texts.

4.2 Premise selection

The original task of (formal) premise selection appeared initially as a task of selecting a (useful) part of an extensive formal library in order to limit the search space for an ATP system, increasing the chance of finding proof for a given conjecture [13]. Premises considered relevant are those that ATPs use for the automatic deduction process of finding proof for a conjecture.

Unlike the original task, NLPS is based not on formally structured mathematics but human-generated mathematical text; therefore, methods that are designed for Formal Premise Selection cannot be straightforwardly applied in a natural language setting. Formal systems expect an explicit definition of every notion used in a mathematical text, together with full disambiguation of used notions and have a clear dependence of definitions, axioms and theorems. The contrast between formal mathematics and mathematical text is made even more apparent with an example. The following is a theorem and proof written in Natural Language:

[1] https://proofwiki.org/wiki/Main_Page

Theorem: There are infinitely many primes: for every number n there exists a prime $p > n$
Proof: Given n. Consider $k = n! + 1$, where $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$. Let p be a prime that divides k. For this number p we have $p > n$: otherwise $p \leq n$; but then p divides $n!$, so p cannot divide $k = n! + 1$, contradicting the choice of p. QED

The same example in a formal setting can be written as:

```
reserve n,p for Nat;
theorem Euclid: ex p st p is prime & p > n
proof
set k = n! + 1;
n! > 0 by NEWTON:23;
then n! >= 0 + 1 by NAT_1:38;
then k >= 1 + 1 by REAL_1:55;
then consider p such that
A1: p is prime & p divides k by INT_2:48;
A2: p <> 0 & p > 1 by A1,INT_2:def 5;
take p;
thus p is prime by A1;
assume p <= n;
then p divides n! by A2,NAT_LAT:16;
then p divides 1 by A1,NAT_1:57;
hence contradiction by A2,NAT_1:54;
end;
theorem {p: p is prime} is infinite
from Unbounded(Euclid);
```

In the formal setting, definitions need to be mapped to their concepts (e.g., the definition of natural number mapped to `NAT 1 : 3 8`), and the proof is longer and more explicit when compared to its natural language counterpart. While mathematical language avoids ambiguity and favours clarity, it often assumes that the reader have background knowledge on some concepts and uses them without an explicit definition. Such details still render it challenging to apply formal language techniques into natural language or even convert natural language mathematics to formal mathematics in an automated manner.

However, despite the difference between NLPS and premise selection, we can still draw some conclusions from past research in this field. One valuable lesson is how well machine learning and deep learning models can perform on this task. Irving et al. [41] propose a neural architecture based on LSTMs for premise selection using formal statements written in Mizar. Other authors have used machine learning approaches such as Kernel-based Learning [3], k-NN algorithms [29] and Random Forests [27]. However, neural approaches previously presented [41] have obtained higher scores at the (formal) premise selection task.

4.3 Definition: Natural Language Premise Selection

A mathematical statement can be a definition, an axiom, a theorem, a lemma, a corollary or a conjecture. Premises are composed of universal truths and accepted truths. Definitions and axioms are *universal truths* since the mathematical community accepts them without proof. *Accepted truths* include statements that need a proof before being adopted. Theorems, lemmas and corollaries are such types of statements. These statements were, at some point, framed as a conjecture before they were proven. As such, they can be grounded on past mathematical discoveries, referencing their own supporting premises (i.e., the background knowledge that was used to prove the conjecture). This network structure of available premises can be used as a foundation in order to predict new ones. The relationship between these statements is leveraged across this work to build models that can better perform inference for mathematical text.

A **collection of premises of type definitions** D_i is the set of mathematical texts or statements $\{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, ...\}$, where all elements in D_i contains a definition of a concept presented in a proof E_i. For example, the theorem in Figure 4.1 is connected to the mathematical text that defines what is a real function.

> Let $f : \mathbb{R} \to \mathbb{R}$ be the real function defined as: $f(x) = a^x$ where a^x denotes a to the power of x.

Figure 4.1: Example of a premise of type definition.

A **collection of premises of type propositions** is the set of propositions R_i that helps support the argument proposed in the mathematical text of a proof E_i. It is often used as an explanation for certain statements used for the construction of the proof.

Figure 4.2 presents part of the proof, where the name of the premises are highlighted. For example, the mathematical statement of Cauchy's Mean Theorem is a supporting proposition for the proof shown.

Let $x, y \in \mathbb{R}$.
Note that, from Power of Positive Real Number is Positive: Real Number:
$\forall t \in \mathbb{R} : a^t > 0$.
So:

$$
\begin{aligned}
a^{(x+y)/2} &= \sqrt{a^{x+y}} && \text{(Exponent Combination Laws)} \\
&= \sqrt{a^x a^y} && \text{(Exponent Combination Laws)} \\
&\leq \frac{a^x + a^y}{2} && \text{(Cauchy's Mean Theorem)}
\end{aligned}
$$

Figure 4.2: Example of part of a proof, where four premises are present.

The collection of **premises** P_i of a proof written in mathematical text E_i is composed of premise propositions R_i and the premise definitions D_i, i.e., $P_i = R_i \cup D_i$.

Finally, the **Natural Language Premise Selection task** can be defined as: Given a new conjecture c, that requires a mathematical proof, and a collection (or a knowledge base) of premises $P = \{p_1, p_2, \ldots, p_{N_p}\}$, with size N_p, the NLPS task aims to retrieve the premises that are most likely to be useful for proving c. Premises of accepted truth statements can also have a subset of premises $\tilde{P} \subseteq P$.

Figure 4.3 presents an example of a conjecture containing two premises. Both *Premise 1* and *Premise 2* can be used as part of the proof for this conjecture. If both premises are contained in our knowledge base, we would need to retrieve them for the NLPS task.

4.4 Dataset Construction: PS-ProofWiki

In this section, we present our dataset, PS-ProofWiki, and detail the steps taken for its construction. Our dataset is available as a set of JSON files in `http://github.com/debymf/nl-ps`, together with the code for parsing and generating PS-ProofWiki. A summary of the process is presented in Figure 4.4.

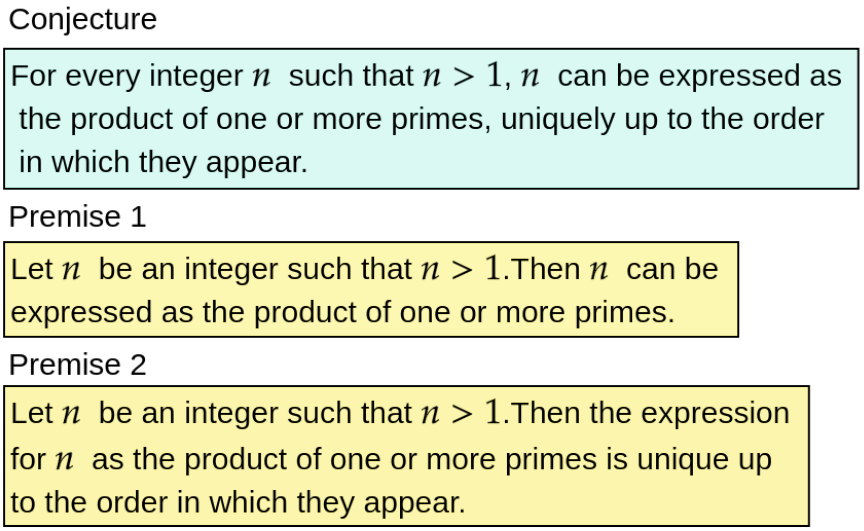


Figure 4.3: Example of a conjecture and its premises.

4.4.1 Parsing the corpus

The proposed dataset was extracted from the source code of ProofWiki. We include an example of the source code of a ProofWiki page in Appendix B.1. ProofWiki is an online compendium of mathematical proofs, with the goal to collect and classify mathematical proofs. ProofWiki contains links between theorems, definitions and axioms in the context of a mathematical proof, determining which dependencies are present. Different collaborators manually curate ProofWiki; therefore, there are different styles of mathematical text and many elements cannot be extracted automatically. Other Wikis have been used in the past to generate successful and widely used datasets for the NLP domain [96].

4.4.2 Cleaning wiki tags

ProofWiki uses Wikimedia tags, same as Wikipedia; however, it has also specific tags related to the mathematical domain (e.g., the tag {{qed}} used to represent the end of proofs in ProofWiki). Consequently, we cannot use default wiki extraction tools. A bespoke tool was developed to comply with ProofWiki's tagging scheme. There is also a particular tag for referring to another mathematical text, using passages from other texts in order to support a claim (Figure 4.5).

This stage does not remove any entry from the initial dataset. At this point, we have a total of **78,793** entries in the dataset.

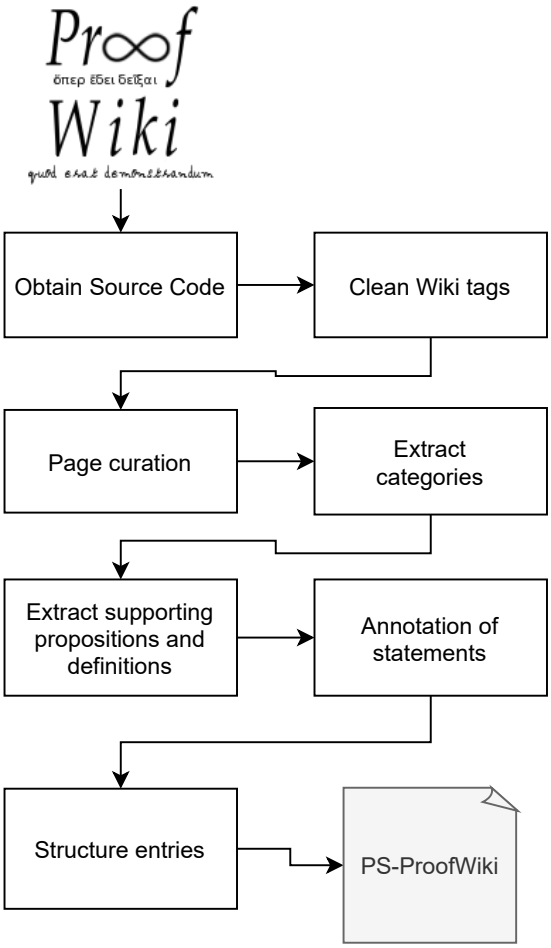


Figure 4.4: Pipeline used to build the PS-ProofWiki dataset.

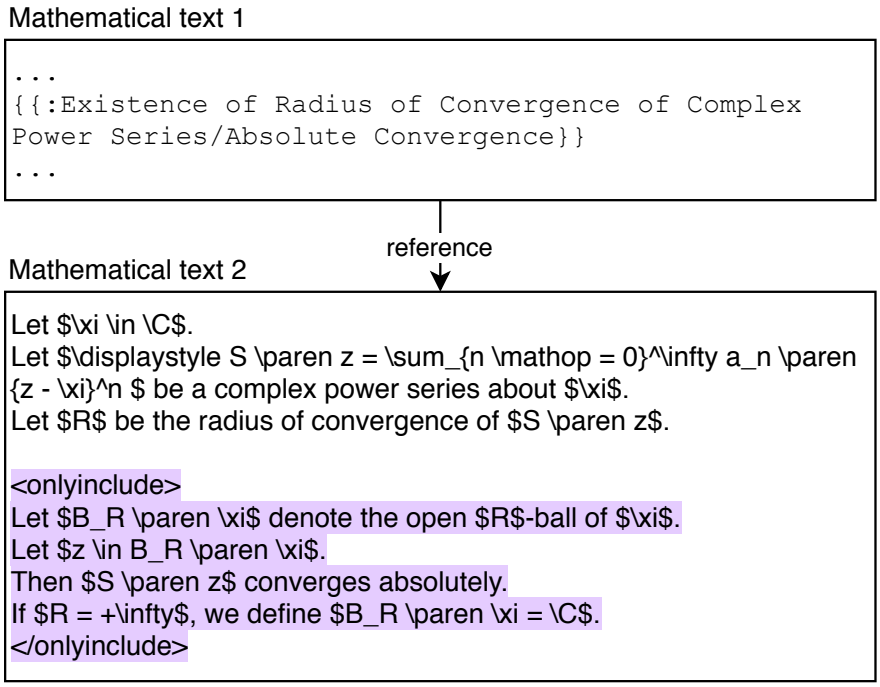


Figure 4.5: An example where Mathematical text 1 references a passage in Mathematical text 2 using the name of the passage to be referenced between curly brackets. Only the part highlighted is being referenced.

4.4.3 Curating pages

Several pages in ProofWiki are not directly related to mathematical propositions or definitions, such as users pages and help pages.. We manually analysed the pages types and removed the ones that are not definitions, lemmas, theorems or corollaries. The pages containing the following strings in the titles were removed from the dataset: User talk, Mathematician, User, File, Talk, Book, Definition talk, Euclid, Template, Symbols, Help, Greek Anthology Book XIV, Category talk, Axiom talk, ProofWiki, Template talk, MediaWiki, Help talk, Book talk, Mathematician talk, File talk, ProofWiki talk, Symbols talk.

Some pages also contained tags indicating that the page in question was only used to redirect to another page; those cases were removed from the dataset. At the end of this stage, a total of **43,786** entries remained in the dataset.

4.4.4 Extraction of categories

ProofWiki has associated categories for each page. However, the categories are not harmonised across definitions and propositions. We merged different categories that

belonged to the same mathematical branch and selected the categories that contained at least 100 different entries, since often one page would belong to more than one category.

The categories selected were: Analysis, Set Theory, Number Theory, Abstract Algebra, Topology, Algebra, Relation Theory, Mapping Theory, Real Analysis, Geometry, Metric Spaces, Linear Algebra, Complex Analysis, Applied Mathematics, Order Theory, Numbers, Physics, Group Theory, Ring Theory, Euclidean Geometry, Class Theory, Discrete Mathematics, Plane Geometry and Units of Measurement.

4.4.5 Extracting premises

The pages in ProofWiki are connected using hyperlinks. We leverage this structure to extract supporting propositions and supporting definitions, which is a central part of the argumentation structure behind a proof. From the mathematical text of proofs, we extract the hyperlinks to other propositions and definitions, considering these as premises to that conjecture. For example, Figure 4.6 presents a theorem and its respective proof. The proof contains links (highlighted) to other propositions; these are premises needed in order to support the construction of the proof. These can then be connected to its original content, as shown previously in Figure 4.3.

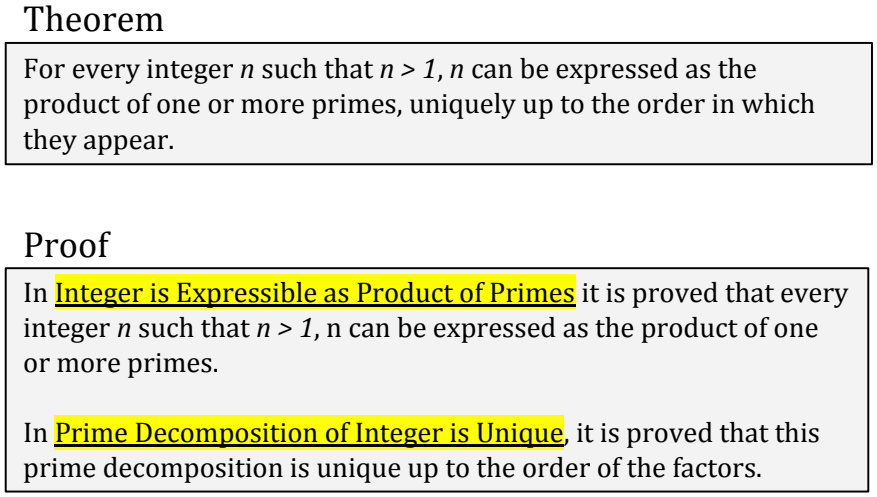


Figure 4.6: Example of premises found inside a proof.

4.4.6 Annotating mathematical text

The entries in ProofWiki are often divided into sections. For PS-ProofWiki, we are only interested in the sections that present a definition, a proposition or a proof. Proofs were curated (combining manual and automatic annotation) to contain only mathematical discourse, removing satellite discourse such as *Historical Notes*. Because some propositions can be proved in different ways, we included various proofs found on one page. After this process, some duplicated pages were removed together with pages that contained incorrect tags. This resulted in a total of **21,746** entries in the dataset.

4.4.7 Structuring the entries into Training, Development and Validation set.

We split the dataset into training, validation, and test to reflect the real-world scenario of proving newly seen conjectures at test time. In order to do this, we have to ensure that no entries in the test and dev set can also be found in the training set or the knowledge base.

The process for obtaining the splits was defined as follows:

1. Create a Knowledge Base containing all mathematical statements.

2. Create a collection of conjectures filtering the mathematical statements that have associate premises.

3. Split the collection of conjectures into Training, Validation and Test set, using sizes of 0.5/0.25/0.25 respectively.

4. Remove all the statements that are in the Test and Validation set from the Knowledge Base. Also, remove reference to those statements in the Training set.

Examples extracted from the Training set and the Knowledge base are included in Appendix B.2.

4.5 Dataset Analysis

PS-ProofWiki has a total of 21,746 different entries, composed of definitions, lemmas, corollaries and theorems, as shown in Table 4.1.

Statement type	Data Split				
	KB	**Train**	**Dev**	**Test**	**All (Unique)**
Definitions	7,077	0	0	0	7,077
Lemmas	252	134	70	69	252
Corollaries	161	113	57	57	275
Theorems	8,715	5,272	2,652	2,636	14,003
Total	16,205	5,519	2,778	2,763	21,746

Table 4.1: Types of mathematical statements present in PS-ProofWiki. The table shows the number divided by the data split. The last columns shows the total unique entries for each mathematical type.

Note that only the Knowledge Base contains definitions since, as previously stated, definitions do not contain proofs and, consequently, do not have premises. However, definitions are often used as premises playing a fundamental role in the NLPS task. There also exists an intersection between the KB and the training set. Accordingly, we include the last column to account for all unique entries in the dataset.

Figure 4.7 presents the distribution of different categories in the dataset. We can notice that the most populated category is the one containing statements about Numbers. This category include definitions of fundamental mathematical quantities, such as the definition of the number π.

Figure 4.8 presents a histogram with the frequency of the different number of premises. We can observe that the statements usually have a small number of premises, with $9,640$ (Around 87% of the entries in the Train/Dev/Test set) statements containing between one and five premises. The highest number of premises for one theorem is 72. The theorem in question claims: "There exists a minimal uncountable well-ordered set. That is, there exists an uncountable well-ordered set Ω with the property that every initial segment in Ω is countable". This particular theorem has the support of different proofs, explaining the large number of premises.

Similarly, the histogram in Figure 4.9 shows the frequency of the different number of dependencies, computing how many times each statement is used as a premise, and we observed that most of the statements are used as dependencies for only a small subset of premises. A total of $4,236$ statements has between one and three dependants. On average, statements contain a total of 289 symbols (characters and mathematical symbols). The specific number of tokens will depend on the type of tokenisation used for the mathematical symbols. For example, the expression $x+y$ can potentially be

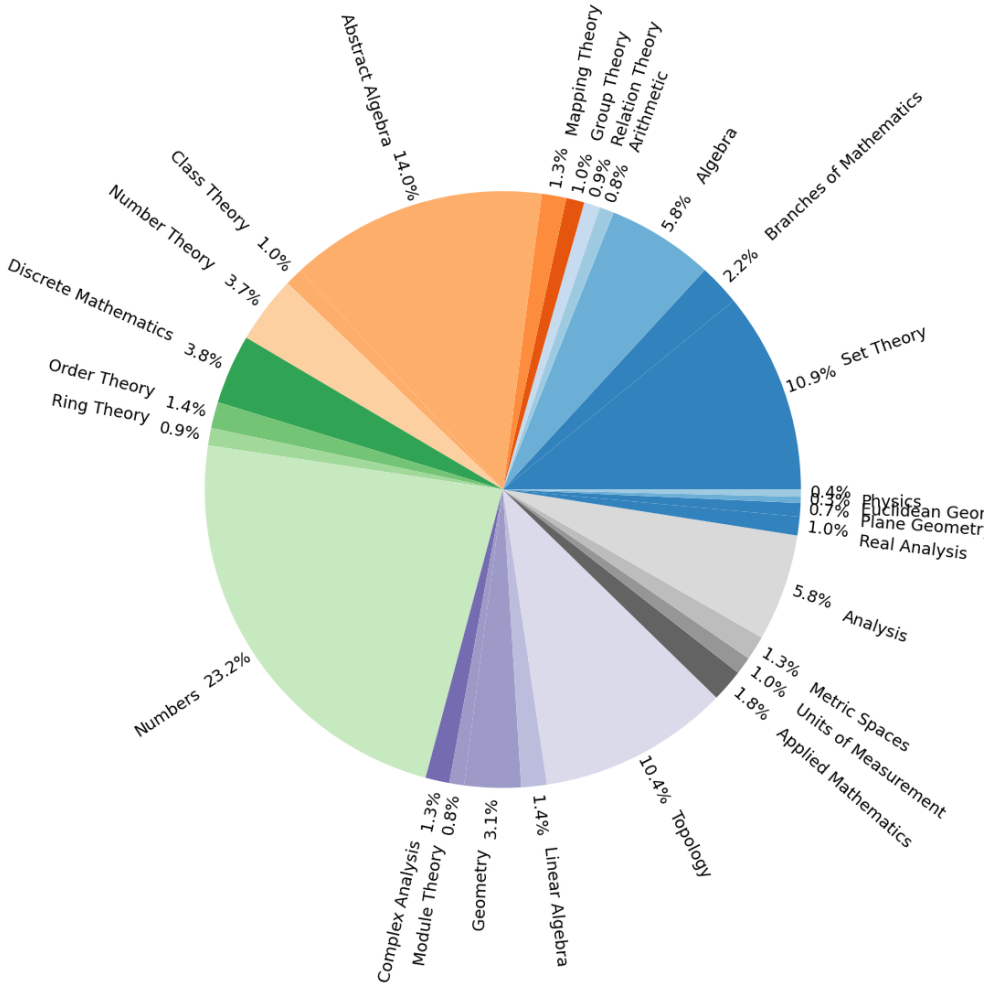


Figure 4.7: Distribution of documents per category in the dataset.

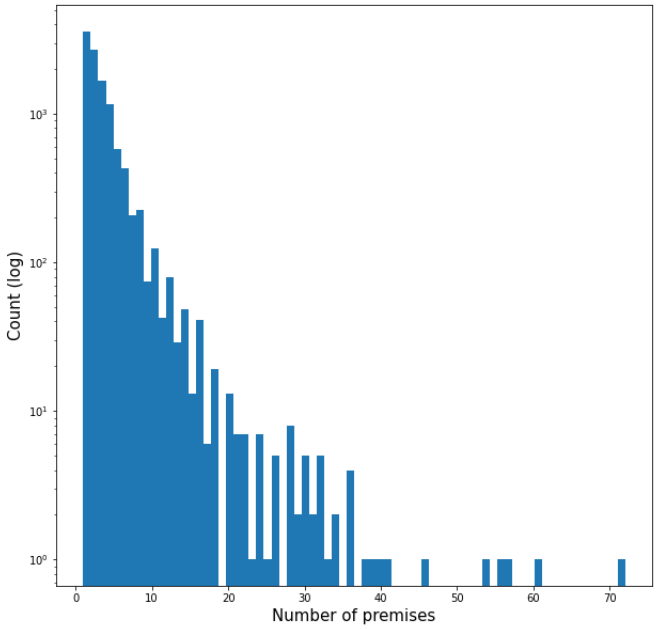


Figure 4.8: Distribution of the number of premises in the ProofWiki corpus. Log transformation is applied to facilitate visualisation for the y axis.

considered a single token or three different tokens x, $+$ and y.

The dataset provides a specific semantic modelling challenge for natural language processing as it requires specific tokenisation and the modelling of specific discourse structures tailored towards mathematical text, such as encoding mathematical elements harmonised with natural language, and encoding the relationship between conjecture-premise and even premise-premise.

4.6 Baselines

In order to identify the challenges of the task of natural language premise selection using PS-ProofWiki, we performed initial experiments using BM25, given that it is a well known Information Retrieval baseline. We use this method to as the supporting weighting scheme for all the mathematical texts in our Knowledge Base. Then, at evaluation time, we also obtain the vectors for each entry in the Dev and Test set to later compute the cosine similarity between each entry and the ones found in the Knowledge Base. The results are then ranked by proximity. We then compute the mean Average

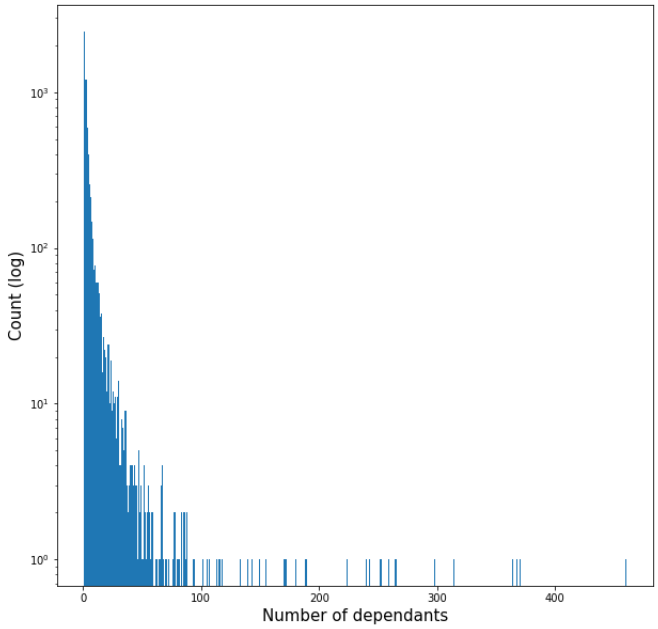


Figure 4.9: Number of times a statement is referred as a premise. Log transformation is applied to facilitate visualisation for the y axis.

Precision (mAP) for each baseline, ranking all possible premises, computed as:

$$\text{mAP} = \frac{\sum_{i=1}^{N} \text{AveP}(\tau_i)}{N}$$

where N is the total number of documents, m_i is the i-th mathematical text and AveP is the average precision.

Table 4.2 presents the initial results. We compare three different types of tokenisations for the mathematical elements. Initially, we treat the expressions and equations as single tokens (Expressions as tokens); for example, the expression "$x+y+z$" would be considered a single word. We also considered tokenised expressions, tokenising operations and operators, the expression "$x+y+z$" would be tokenised as ['x','+','y','+','z']. Finally, we tokenise the whole text as a sequence of characters.

From these initial results, we can conclude that the task is semantically non-trivial and cannot be solved with retrieval strategies based on lexical overlap. We can also notice that we obtain better results when we tokenise the expressions, hinting that the elements representing the expressions at a more granular level generated better

representations for determining the relevant premises.

Split	Tokenisation type	BM25
Dev	Expression as tokens	8.42
	Tokenised expressions	9.44
	Char-level encoding	4.13
Test	Expression as words	8.23
	Tokenised expressions	9.31
	Char-level encoding	3.69

Table 4.2: Result for mAP applying BM25 and comparing tokenisation of expressions. The values for mAP are multiplied by 100.

Table 4.3 presents the results for different pre-trained sentence embedding models. These models are based on Transformers architecture and are fine-tuned to improve retrieval tasks such as semantic search and paraphrasing. While these models are not largely exposed to mathematical text, there is still a considerable performance improvement compared with BM25. These models have their own tokenisers, it was not an option to decide how the mathematical elements would be split.

Model	Dev	Test
all-mpnet-base-v2	**12.65**	**12.34**
multi-qa-mpnet-base-dot-v1	11.94	11.71
paraphrase-multilingual-mpnet-base-v2	8.81	8.89
paraphrase-albert-small-v2	8.30	8.33

Table 4.3: Results for different Sentence-Embedding models. The results are shown for mAP $\times$ 100.

In Table 4.4, we compare the results for different sizes of the dataset. We consider the full dataset and three different categories of subsets. We can notice that for smaller subdatasets, both baselines perform better. This result was expected since with smaller datasets there are less possible premises, and elements from the same categories tend to be more uniform between themselves.

We can also consider the fact that the premises are transitive, i.e., if one mathematical text m_i has a premise p and a mathematical text m_j has m_i as a premise, then p should also be a premise of m_j. Figure 4.10 presents an example with Euclid's Lemma and together with its one-hop and two-hop premises.

	BM25	SBERT
Algebra (1,241)	19.03	33.26
Analysis (1,102)	21.04	35.09
Number Theory (741)	14.64	38.9

Table 4.4: Comparing mAP results for different categories (the number between parenthesis indicates the number of entries for that category). For SBERT, we use the best performing model.

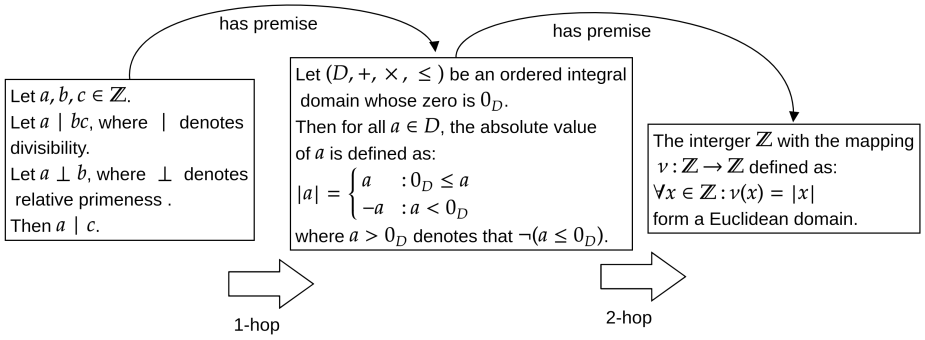


Figure 4.10: Example of premises in a multi-hop scenario.

In this case, the task becomes even more challenging, as we present in Table 4.5, where we consider the transitivity with two hops of distance, contrasting with the classic one-hop scenario.

	BM25	SBERT
2-hop premises	1.93	7.37
1-hop premises	9.31	12.34

Table 4.5: Comparing number of hops needed for obtaining premises.

From the results, we notice that the baselines perform worse when adding one more hop needed to obtain the premise. This suggests the need for the design of PS models which are robust in a multi-hop setting.

4.7 Discussion

In this chapter we proposed a new task for mathematical language processing: natural language premise selection. We also made a new dataset available for the evaluation of the task and we analysed how the dataset works with the task on different baselines.

From our experiments we identified that handling mathematical expressions is crucial for solving the proposed task, encoding the specific semantics of operators and variables at a more granular level can increase the performance of our baselines. This provides evidence on the need for specific embeddings and representation paradigms for mathematical formulas and discourse, which could most certainly improve the prediction of future work in the natural language premise selection task, reinforcing the need to explore **RQ2** (*Can we improve the performance of deep learning models for inference over mathematical text by generating specialised encodings for mathematical natural language elements?*) and **RQ3** (*How to encode the semantics of variables at sentence-level representations?*).

We also identify that the task becomes more challenging when we consider that the premises are transitive, suggesting that the task could benefit from methods that capture the relationships conjecture-premise and premise-premise, such as structural semantic similarity **RQ4** (*Can the dependencies between mathematical statements provide a simplified representation of mathematical arguments?*) and graph-based representations **RQ5** (*How to encode the inter-statement dependency structure to support inference for the natural language premise selection task?*).

The results presented in this Chapter provide the necessary evidence to answer **RQ1** (*How to support the evaluation of models for the interpretation and inference over mathematical text (regarding inference tasks, evaluation measures and supporting datasets)?*). While still performing at a baseline level, they comparatively point in the direction of using transformer-based models as a foundation for statement-level representation. When compared with IR baselines, there is an observable advantage in the use of dense representations (deep-learning encodings), especially when considering the subsets of PS-ProofWiki and more challenging tasks such as the multi-hop premise selection. This result provides evidence for the potential of such models, highlighting the need for more complex models adapted to this type of discourse.

In order to build this dataset and evaluation framework, some assumptions were made:

- ProofWiki is a data source that contains a representative set of mathematical statements:

  ProofWiki is a slice of the mathematical content that exists. While it is not practical to verify if this data source contains the same distribution of Mathematical topics as one would find in other data sources, we assume that ProofWiki is

representative enough to evaluate generalisability to different Mathematical Areas, given that it contains a large number of mathematical statements and a wide range of different categories. However, this data source is limited in terms of Mathematics applied to other sciences. Therefore, we do not assume that this work is generalisable to areas that apply Mathematics.

- The majority of mathematical statements and proofs in ProofWiki are valid and follows correct mathematical semantics and syntax:

  While we cannot manually verify all the mathematical statements in our dataset, we assume that ProofWiki contains mostly statements and proofs that are mathematically correct. This type of assumption has been made in the past, given that Wikis are known to work well as good data sources, for example, having been used to evaluate question-answering systems [96] and serving as a data source for training different models [25].

- The hyperlinks found inside the proof of a mathematical statement represents the external knowledge needed for proving that statement, reflecting the notion of relevance.

  Overall, we found the pattern that mathematical statements that are linked inside a proof in ProofWiki are helpful in proving that statement, and that is what we based our notion of relevance on. While we cannot assume that the links contain all the relevant premises, we assume that all the statements found are relevant. As previously mentioned, in mathematical language, often when writing a proof, some assumptions are made in terms of what knowledge needs to be explicitly defined or brought to the surface. Mathematicians frequently write statements assuming a certain level of knowledge from the reader. Therefore, it is impossible to provide a precise notion of completeness in terms of relevant premises since that might be relative for each person. However, we can define relevant in terms of what can be used and cannot be used as part of a proof. We assume that ProofWiki correctly maps this through the hyperlink structure found in the proofs.

4.8 Reproducibility Statement

In order to make this dataset construction reproducible together with the experiments performed here, the repository `http://github.com/debymf/nl-ps` contains all the

code executed for generating this dataset, together with the annotations and the implementation of baselines.

The ProofWiki source was obtained on Apr 18, 2021, from `https://proofwiki.org/xmldump/latest.xml`. We include in the repository a copy of the dump from that date.

Chapter 5

Retrieving Relevant Premises

The previous chapter has shown promising results with the use of pre-trained language models for the task of Natural Language Premise Selection, with deep learning based representations outperforming IR baselines. This chapter expands the investigation of the task by examining **RQ4** (*Can the dependencies between mathematical statements provide a simplified representation of mathematical arguments?*), comparing the performance of Sentence-Transformers models with a novel method capable of encoding similarity between premises at statement and structure levels, computing similarity in terms of conjecture-premise and premise-premise relationships. While Sentence-Transformers models are designed to retrieve sentences that have either similar meanings or an entailment relationship, this chapter empirically shows that these models can be improved for the NLPS task with the encoding of the relationship between premises. The results shows that the conjecture-premise relationship should not only be encoded at statement-level.

5.1 Motivation

Given two input mathematical statements, we want to automatically derive a relevance score indicating the likelihood of these statements with regard to a conjecture-premise relationship. The previous chapter shows that pre-trained language models can better encode this relationship, obtaining higher mAP scores than IR baselines, despite both types of models achieving low scores. Such models have a limitation in common: both can represent statements only at a local level (i.e., elements inside a single sentence or statement). As previously described, mathematics entail a network structure, where discoveries are built on top of previous ones. None of the models presented in the

last chapter can encode this structural relationship between statements. Consider the following theorems:

> Statement 1: Let F_n denote the n-th Fibonacci number. Let F_n be a prime number. Then, apart from $F_4 = 3$, n is a prime number.

> Statement 2: Let $p \in \mathbb{Z}_{>0}$ be a prime number. Let F_p be the p-th Fibonacci number. Then F_p is not itself necessarily prime.

Both theorems contain shared tokens, such as "prime number", "Fibonacci number", and even variables in common, such as F. An approach based on only statement-level similarity, where statements are represented closer in proximity in a semantic space based on the meaning of a sentence, would likely classify both statements as related. In the case of this example, this would imply that Statement 1 could potentially be retrieved as a premise to Statement 2, and vice-versa. Using this type of relevance score would lead to incorrect results since, even though both statements are related, they do not depend on each other for their proofs; both statements can be proven in a completely independent manner. We hypothesise here that the similarity obtained at a statement-level should not be the only guiding metric to retrieve relevant mathematical statements and can be improved with the addition of an component that encodes the similarity at structure-level.

Even though both statements from the above example do not encode a conjecture-premise relationship, they share common premises. Both statements are connected to the definition of a *prime number* and the definition of a *Fibonacci number*. In Mathematics, we often find that similar theorems are proven using shared mathematical statements. This chapter explores this property further and empirically verifies that this can be generalised to a larger dataset. Given a conjecture, we want to retrieve possible premises based on the statement level similarity and on the structural similarity, structure based on the of premises of similar statements.

This idea of retrieving elements based on structural similarity has been explored previously in retrieving explanations for science questions answering [84]. A given statement is considered explanatory in the science domain if it performs an *unification*, showing how a set of initially disconnected phenomena expresses the same regularity (e.g., statements that do not have a direct relationship but share the same regularity in terms of having the same premises). One example of unification is Newton's law

of universal gravitation, which explains phenomena for bodies with mass. In Mathematics, definitions will often act as unification statements since they are reused across different proofs and even other definitions. For example, the definition of the set of integer values $\mathbb{Z}$ is often used to define the scope of variables across different areas of mathematics, being reused several times. In the following section, we present how to incorporate unification score [84] into the NLPS retrieval pipeline.

5.2 Proposed Method

Given a new conjecture c and a knowledge base of mathematical statements KB, we want to retrieve the premises from KB that are relevant for finding proof for conjecture c. Our proposed method is based on the composition of statement similarity and the structural similarity relevance score. The following sections describe how these scores are computed and combined for an end-to-end retrieval method.

5.2.1 Statement Similarity Relevance Score (StaSim)

The Statement Similarity Relevance Score is the local score computed between conjecture and premise, considering the distance in a semantic space between vectors encoding both conjecture and premises. A fundamental design requirement for this function is scalability. The use of traditional ML schemes for the induction of this similarity function, such as classification or pair-wise regression, to compute the relevance score between pairs of statements leads to many possible pair combinations. Every new conjecture would require N_{KB} inference computations, resulting in an approach that does not scale.

In order to address this requirement, this approach focuses on mapping each statement to a vector space, such that semantically similar statements are represented closer to each other. From a vector representation, one can efficiently compute the similarity of statements using metrics such as cosine similarity or euclidean distance. We define an embedding model $\gamma : KB \cup c \mapsto \mathbb{R}^d$, where d is the dimension of the output vector. Each element in the KB is converted to a vector using γ. The same operation is performed with the conjecture c. This embedding model can be any dense semantic encoder such as Sentence-Transformers. In our experiment section, we show different encoders being used.

The Statement Similarity score between two mathematical statements m_i and m_j is

then computed using the cosine similarity as follows:

$$StaSim(m_i, m_j) = \frac{\mathbf{m_i} \cdot \mathbf{mi_j}}{\|\mathbf{m'_i}\| \|\mathbf{m'_j}\|} \tag{5.1}$$

The embedding model is fine-tuned for premise scoring using a Siamese neural network [45] approach, where a pair of statements are simultaneously fed into two networks with shared weights. The representation for each statement is obtained and compared using cosine similarity, where the expected score is 1 if both pairs represent a correct conjecture-premise pair or 0 otherwise.

In order to fine-tune such a model, we need both negative and positive pairs of conjecture-premise. We want to approximate $P(m_i|c)$ for $m_i \in KB$ during training. For every positive pair, we generate $K-1$ random negative examples, summing up to a total of K pairs for each conjecture. The goal is then to minimise the approximated mean negative log probability of the data. For a single batch, with conjectures c_i and mathematical statements m_i, this loss is computed as follows [34]:

$$\mathcal{J}(m, c, \gamma) = -\frac{1}{k} \sum_{i=1}^{K} [StaSim(m_i, c_i) - log \sum_{j=1}^{K} e^{StaSim(m_i, c_i)}] \tag{5.2}$$

5.2.2 Structural Similarity Relevance Score (StructSim)

The similarity between conjectures and premises can also be considered at a structural level. We hypothesise here that when statements are similar, they often have some premises in common (i.e., statements with higher statement-level similarity will also have a higher structural similarity). The Unification Score [84] is able to improve the retrieval considering this structural similarity. This score was initially proposed as an explanation retrieval mechanism, where it was found that similar questions will have explanations (that could lead to an answer) in common. Given the similarity of the previous application scenario, we apply the unification score in this context to define our Structural Similarity Score:

$$StructSim(m_i, c) = \sum_{p_k \in kNN(c)}^{k} StaSim(\gamma(m_i), \gamma(c)) \cdot \mathbb{1}(m_i, m_k) \tag{5.3}$$

$$\mathbb{1}(m_i, m_k) = \begin{cases} 1 & \text{if } m_i \text{ is premise of } m_k \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

where $kNN(c) = \{m_1, \ldots, m_K\}$ represents a list of mathematical statements retrieved

according to the semantic similarity between the embeddings $\gamma(m_i)$ and $\gamma(m_k)$, and $\mathbb{1}(\cdot)$ is the indicator function verifying whether m_i is a premise for m_k. In this case, mathematical statements that are used as premises across several semantically similar conjectures are going to have higher scores. Where k is the number of neighbours we consider when investigating the structure of c to compare with the neighbourhood of m_i. In the experiment section, we will test different values for k to decide how large this neighbourhood should be for this task.

5.3 Conjecture Premise Relevance Score

Finally, the Conjecture-Premise Relevance Score (CPRS) between a conjecture c and a mathematical statement m_i is defined as a linear combination of both scores:

$$CPRS(c,m_i) = \lambda * StaSim(c,m_i) + (1-\lambda)StructSim(c,m_i) \tag{5.5}$$

where λ is a temperature that defines how much each component impacts the final results. This is an hyperparameter that is fine-tuned in the experiment section.

Using this relevance score, we can retrieve the top-n most relevant premises with the following algorithm:

Algorithm 1: Obtaining top n premises for a conjecture.

Data: Conjecture c, KB of premises, positive integer n
Result: Top n relevant premises
Initialise *all_scores* as an empty list;
for $p \in KB$ **do**
    $score \longleftarrow CPRS(c,p)$;
    Concatenate *score* to *all_scores*;
end
all_scores $\longleftarrow$ descending sort *all_scores*
return first n elements of *all_scores*

5.4 Experiments

This section describes the experiments performed using the conjecture-premise relevance score. Table 5.1 presents the results of the experiments performed using PS-ProofWiki. To further explore the capabilities of this model and contrast with other approaches, we report here the metrics in terms of Mean Average Precision and Recall

at 10 and 100, which are standard for Information Retrieval approaches. As a baseline, we include again the results for BM25 and different Sentence-Transformers models without any fine-tuning.

Model	Dev			Test		
	mAP	**R@10**	**R@100**	**mAP**	**R@10**	**R@100**
BM25	10.22	12.29	29.98	10.32	12.72	29.54
No extra finetuning						
all-mpnet-base-v2	12.65	15.76	37.41	12.34	16.11	37.12
multi-qa-mpnet-base-dot-v1	11.94	14.43	34.20	11.71	14.56	34.11
scibert-nli	8.64	9.74	23.70	8.40	9.68	22.39
roberta-base-nli-mean-tokens	6.52	7.18	15.89	6.04	7.25	16.65
nli-bert-base	6.75	7.73	18.67	6.67	8.13	18.53
FineTuning						
all-mpnet-base-v2	16.16	20.43	52.11	16.09	21.12	50.99
MathBERT	11.43	14.15	42.79	10.86	14.28	41.60
bert-base-uncased	13.32	17.24	47.40	13.09	17.52	45.88
CPRS	**31.88**	**39.64**	**63.32**	**31.10**	**38.70**	**61.92**

Table 5.1: Results for premise selection retrieval. Here we compare our approach with pre-trained models and BM25.

We also include the results for the fine-tuned models for the Statement Similarity (StaSim) only (Fine-Tuning section of the table). For the CPRS, we chose the fine-tuned model with the highest score (all-mpnet-base-v2) as our embedding model. We also tried different values for λ and k and found that the best results are obtained using $\lambda = 0.6$ and $k = 10$.

The fine-tuned sentence-embedding model shows a significant improvement compared to the non-fine-tuned results. The model `all-mpnet-base-v2` achieves a considerable improvement in Recall@100 while also achieving higher results for the other metrics. Such results support the idea that deep learning models, fine-tuned for this task, can improve the retrieval results. In this case, these models start as a statement-level similarity retrieval model and are fine-tuned towards the specialised task of premise retrieval. However, are still unable to encode the structure that follows each statement.

We also fine-tuned the traditional BERT model (bert-base-uncased), and MathBERT. Even though MathBERT is pretrained over mathematical corpora, it performs worse than BERT, suggesting that feeding more mathematical content to a model is not enough to improve its performance over mathematical text.

The best result was achieved with our proposed method. The conjecture-premise relevance score can achieve almost double of mAP than the traditional semantic retrieval, together with a significant increase in Recall@10. In a real scenario, where a mathematician is interested in retrieving premises, Recall@10 is the most relevant of all considered metrics, since the higher the rank of the useful premises, less time is needed looking through the retrieved results.

Given that two different scores are combined in our approach, we also perform ablation studies to understand how much both scores influence in the final results. Table 5.2 presents the ablation studies performed. The first row presents the results for the StaSim score only, and the second row presents the values for StructSim. The StaSim scores are the same as the ones seen for the fine-tuned sentence embedding model in Table 5.1.

Model	Dev			Test		
	mAP	**R@10**	**R@100**	**mAP**	**R@10**	**R@100**
1. StaSim only($\lambda = 1$)	16.52	21.14	51.22	16.47	21.11	51.22
2. StructSim only ($\lambda = 0$)	30.77	38.31	61.96	30.48	37.93	61.51
3. BM25 encoder	27.32	34.40	54.54	25.57	32.55	52.75
CPRS	**31.88**	**39.64**	**63.32**	**31.10**	**38.70**	**61.92**

Table 5.2: Ablation studies, comparing the different components that are part of the Conjecture-Premise relevance score.

The third row presents the results with BM25 as an encoder instead of the Sentence-Transformer model. It is clear that the achieved results are not as strong as those obtained with the fine-tuned model; however, there is a substantial improvement compared to the use of BM25 for statement similarity only. This highlights the benefit of using our approach across different encoders. Those results also highlight the major role that the structural similarity has in the NLPS task. All the results perform better considering StructSim in contract with the StaSim.

Figure 5.1 and Figure 5.2 shows how mAP changes when trying different values of λ and k using BM25 and SBERT as encoders, respectively. We can notice that in both scenarios, selecting a smaller set of connected premises (smaller k) is beneficial for the task of NLPS. This behaviour is different from what has been seen in the domain of science question answering [84], which usually takes a value of k of at least 100.

As seen in Chapter 4, most of the statements are connected to a small number of other statements, having a large value for k leads to more noise added to the retrieval leading to a degradation in performance.

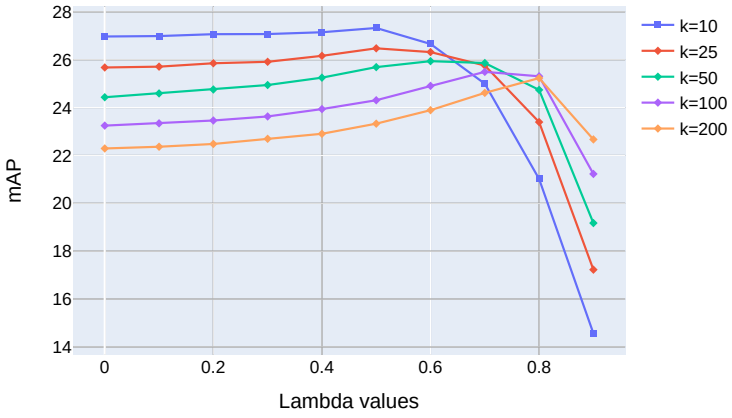


Figure 5.1: Different λ values and k for BM25.

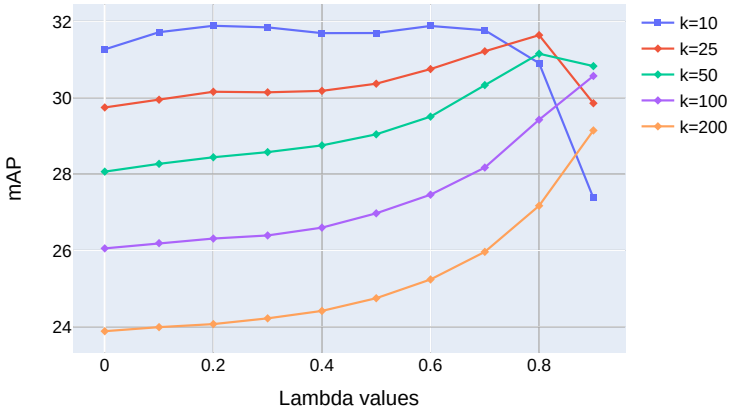


Figure 5.2: Different λ values and k for SBERT.

5.5 Discussion

This chapter has proposed the Conjecture-Premise Relevance Score, which is computed using statement and structure level representations. This method improves on top of Sentence-Transformers, which is only able to encode statement-level representations. This chapter has also shown that the statement-level representations have a weak performance in the NLPS task, even fine-tuned models are still not able to achieve robust performance. This reinforces the idea that NLPS is a challenging new task in the

field of MathLP.

The results support the idea that in order to achieve strong results in Mathematical Language Processing tasks, we need to encode particular properties of the mathematical text. We show that the addition of a structural similarity component is able to consistently increase the retrieval score for the NLPS task across different models (both Transformer-based models and BoW-based methods). These results help answer **RQ4** (*Can the dependencies between mathematical statements provide a simplified representation of mathematical arguments?*), suggesting that for the NLPS task, we need to consider not only the similarity at statement-level, but also the argument-structure that connects mathematical statements. We empirically have shown that similar statements have a similar structure considering connected premises. This structure can then be leveraged to improve the retrieval performance for NLPS. The structure plays a major role in this retrieval, showing a large increase in retrieval scores when applied across different models. In Chapter 7, we further explore this idea, showing that these structures can also be encoded latently via Graph Neural Networks.

We also presented how a technique designed for general school-level science questions can also be applied to a more complex mathematical domain, suggesting a a regularity across different sciences. We leave the exploration of this behaviour across other fields of Science for future work.

While the results obtained have substantially outperformed the baselines, the method still has some limitations, namely.

- The output results relies heavily on the quality of the embedding function being used.

  The Statement Similarity and the Structural Similarity rely on an external pre-trained model that converts each statement to a semantic vector. The encoding generated by the proposed method will depend on the quality of the obtained representations from this original model. This hints that we could potentially obtain better results with pre-trained models designed for mathematical discourse.

- Our method is only suitable for a large dataset containing the annotated relationship between statements.

  Since our method is based on Structure Similarity, one of the limitations is that we require this structure (in our case, the relationship between statements) to be explicitly annotated in the dataset, so this can be leveraged for the retrieval.

Our method is not suitable for a scenario with only a few examples or for an unsupervised setting.

5.6 Reproducibility Statement

The implementation of all retrieval models, including the baselines, can be found in the repository `https://github.com/debymf/premise_selection_retrieval`. The dataset used is PS-ProofWiki, which was presented in the previous chapter.

Each one of the fine-tuned models were trained using 4 Tesla 16GB V100 GPUs for 5 epochs in total with batch size 32 and seed 42. The hyperparameters adopted are as follows:

- gradient accumulation steps = 1;
- learning rate = 5e-5;
- weight decay = 0.0;
- adam epsilon = 1e-8;
- warmup steps = 0;
- max grad norm = 1.0;

Chapter 6

Linguistic modalities in the mathematical text

Mathematical statements have a particular discourse structure that makes it challenging to use traditional NLP techniques. Some of its distinctive features are: (1) Entangled dual lexical spaces for the mathematical elements (ME) and natural language (NL); (2) Distinct syntactic phenomena between ME and NL. This chapter proposes a new method for representing these different types of linguistic modalities present in the mathematical text, answering **RQ2** (*Can we improve the performance of deep learning models for inference over mathematical text by generating specialised encodings for mathematical natural language elements?*).

6.1 Motivation

Given the linguistically hybrid nature of the mathematical discourse, where two different linguistic modalities co-exist in the same text, sentence-level representation models are not able to capture the different semantics for each modality [31]. For example, in the mathematical domain, variables are represented using recurring symbols; this lexical dimension is unable to capture the semantics of the variables, i.e. the context surrounding the variables is more important than the symbol itself. When interpreting mathematical discourse, such particulars need to be taken into account.

In this chapter, we propose STAR, a cross-modal representation for mathematical statements for addressing the task of premise selection. In order to interpret the different modalities in the mathematical discourse (natural language and equational), STAR uses two different self-attention layers, one focused on the mathematical elements,

such as expressions and variables, while the other attends to natural language features. STAR is taught to see disentangle these two symbolic modalities as parts of different languages, the mathematical language and the English Language, in analogy to cognition [15]: even though the brain interprets mathematics as a language, it processes it with the support of different structures [7]. Using different attention layers, STAR models that dual linguistic aspect, encoding separately these two modalities.

The approach presented in this chapter is based on the hypothesis that the use of cross-modal attention-based mechanisms provides a better encoding of the semantic content of mathematical statements for the task of premise selection.

6.2 Natural Language Premise Selection as Inference Task

Similar to previous methods for premise selection [41], we reformulate the NLPS as a pairwise relevance classification problem. Given a pair (c, p_i), containing a conjecture c and a premise p_i, we classify if p_i can be used for proving c. Considering the large existent number of possible premises (as mentioned before, in PS-ProofWiki, there are more than 16k entries in the knowledge base), it is not scalable to perform classification over all the different possible pairs of conjectures and premises.

We design a new subtask of NLPS which samples a smaller subset of this problem but still reflects its semantic diversity, allowing us to test in a more feasible manner which approaches suit better mathematical text. The proposed setting is similar to the classic Natural Language Inference/Entailment scenario, where given a pair of premise and hypothesis, a model needs to decide if the premise entails the hypothesis. In this new NLPS scenario, given a pair of premise p_i and conjecture c, we need to design a classifier f, such that, $f(c, p_i) = 1$ if the premise can be used as part of the proof for the conjecture or $f(c, p_i) = 0$ otherwise. A *positive pair* is one where the premise can be used, and with a *negative pair*, we have the opposite. As we will see in the rest of this chapter, we can design different methodologies for selecting the negative pairs and raising the task's difficulty by adding more negative pairs. One example of a positive and a negative pair is shown in Figure 6.1.

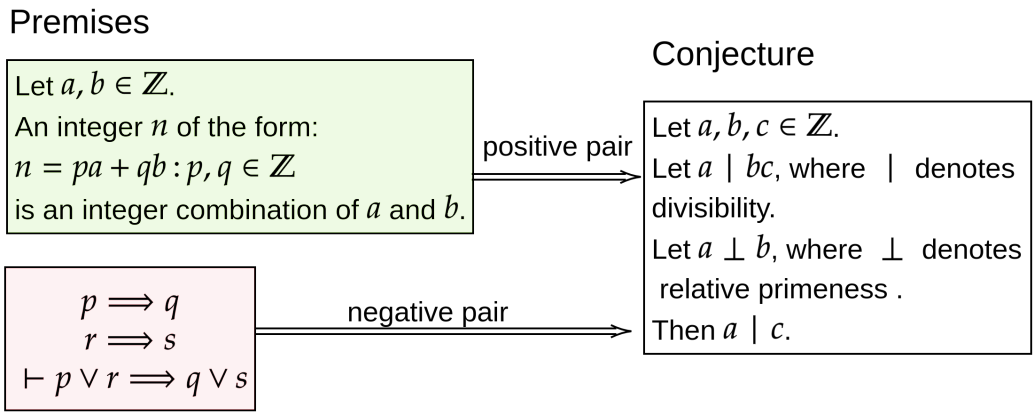


Figure 6.1: Example of a positive and a negative pair.

6.3 Cross-modal STAtement Representation (STAR)

Given the set of mathematical statements $\mathcal{M}$ and a statement $m \in \mathcal{M}$, m is defined as a sequence of elements $m = \{s_1, s_2, \ldots, s_n\}$, where $s_i \in \mathcal{W}$, the set of words, or $s_i \in \mathcal{E}$, the set of mathematical elements present in $\mathcal{M}$. These components are situated in different lexical spaces; therefore, a function to generate a representation for m should take this into account.

We define an embedding model $\gamma : \mathcal{M} \mapsto \mathbb{R}^d$, where d is the dimension of the output vector. The complete architecture is presented in Figure 6.2, where part of a statement is shown as an input example. Each layer is described in detail below.

6.3.1 Token embedding layer

The input to the embedding model is a mathematical statement. This embedding layer is a $W_E \in \mathbb{R}^{k \times v}$ where k is the dimension of the word embeddings, and v is given by $|\mathcal{W}| + |\mathcal{E}|$.

6.3.2 Word/Expression-specific Self-Attention Layer

Inspired by the behaviour of the human brain, Two layers of self-attention [85] are introduced, one for each language modality, encoding distinctively these two linguistic behaviours. One layer captures specific natural language linguistic features, while the other represents particular mathematical formalism features. Given a matrix of queries

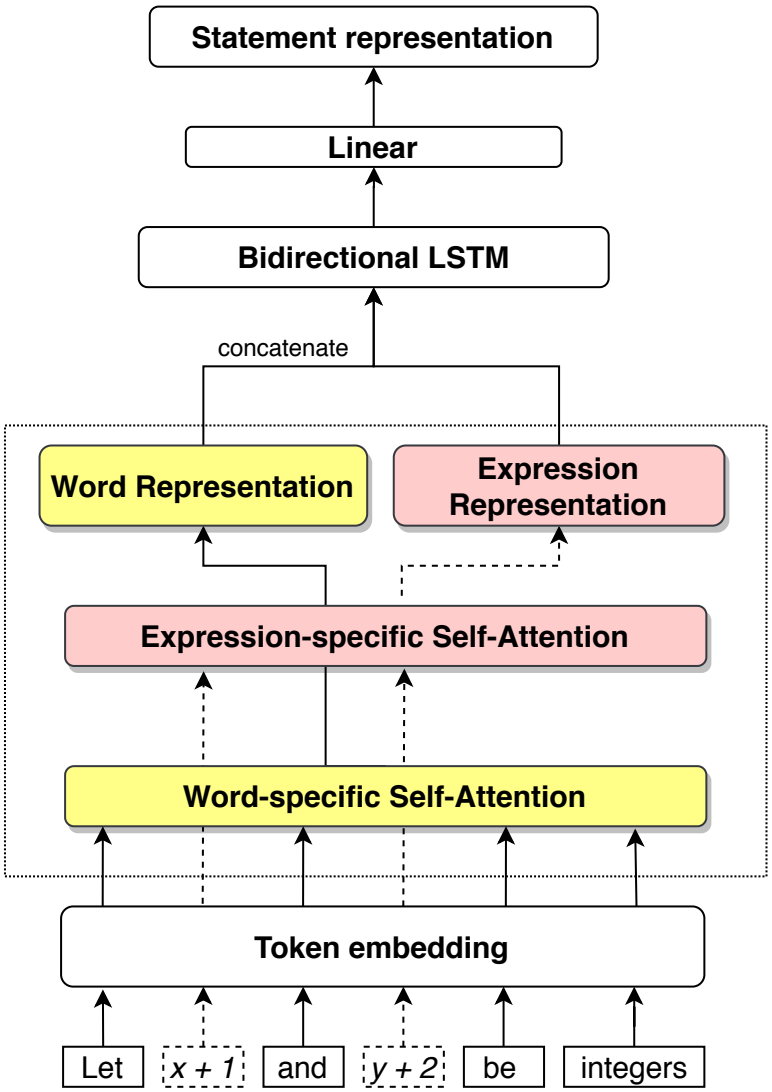


Figure 6.2: This figure presents the model used to generate a representation for each statement, where we combine two self-attention layers, one for each token modality.

Q and matrices of keys and values K and V. The attention head is defined as:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{6.1}$$

where d_k is the dimension of the keys.

These attention heads compose a multi-head attention mechanism, defined as:

$$\begin{aligned} &MultiHead(Q,K,V) = \\ &Concat(head_1,\ldots,head_h)W^O \end{aligned} \tag{6.2}$$

where:

$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$

and W_i^Q W_i^K and W_i^V are parameter matrices. In order to apply self-attention, we consider Q, K and V as the same values, obtained using a linear layer on top of the output of the embedding layer. Words and expressions tokens have a very distinct nature, and we hypothesise that these two layers allow learning and representing these differences.

6.3.3 Long Short-Term Memory Layer

LSTM networks [38] are a complex activation unit, based on a chain structure explicitly designed to capture long-term sequence dependencies. LSTM is a suitable technique for treating sequential data such as mathematical statements. This layer was described in detail in Chapter 2.

6.3.4 Training objective

Finally, in order to obtain the score between conjectures and premises, a siamese neural network setting is used (Figure 6.3), where a pair of statements are simultaneously fed into two networks, with shared weights. This allows the model to learn the representation of each statement individually, while still being aware that the statements belong to the same semantic space.

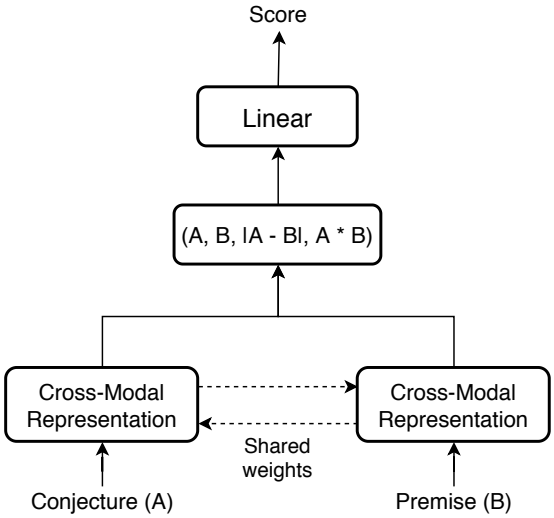


Figure 6.3: Siamese Network used for classifying the conjecture-premise pair, based on the representations obtained.

The representation for each statement is obtained and combined, where the expected score is 1 if B is a premise to A, or 0 otherwise.

The used training objective function is the Cross Entropy Loss, defined as:

$$\mathcal{L} = -\frac{1}{N}\sum_{n=1}^{N}\left[Y_n \log \hat{Y}_n + (1-Y_n)\log\left(1-\hat{Y}_n\right)\right] \tag{6.3}$$

where Y is the predicted classification and $\hat{Y}_i$ is the expected classification.

6.4 Experiments

This section presents the experiments performed to test our hypotheses. We use the dataset PS-ProofWiki for these experiments. In contrast with previous chapters, where we were working with an Information Retrieval task, this chapter solves the related inference problem. Therefore, we use appropriate inference metrics for evaluation, such as F1-score, Precision, and Recall. We adapt the dataset, framing the problem as a pair classification task. For each positive pair, where the statement is a premise to the conjecture, there can be n number of negative pairs. For testing the robustness to noise in the proposed model, we use $n \in \{1, 2, 5, 10\}$. The number of entries for Train, Validation and Test for each value of n is shown in Table 6.1.

n	**Train**	**Val**	**Test**
1	32,758	10,798	10,112
2	49,137	16,197	15,168
5	98,274	32,394	30,336
10	180,169	59,389	55,616

Table 6.1: Number of entries for Training, Validation and Test for different values of n.

The negative pairs are obtained using two different methods. The first collects random examples of statements that are not premises to form a new pair (**negative examples**). In the second technique, we use BM-25 to retrieve statements that are lexically similar to the premises, but that are not part of positive pairs (**similar examples**). For these experiments, we used 512 as a the size of the hidden units layer in the LSTM, embedding size and output statement vector in the embedding architecture. We used 50 epochs for each training round. As shown in Figure 6.4, with this number of epochs

we achieve convergence for all values of n. For each epoch, the validation set was evaluated, and the best model was chosen for testing.

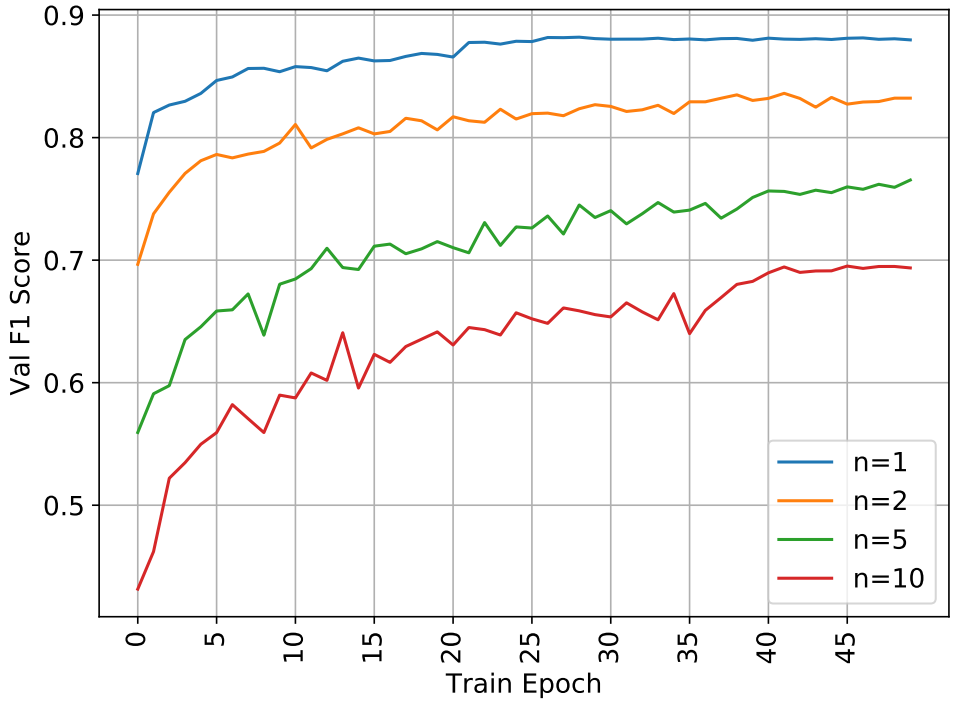


Figure 6.4: Number of training epochs and the obtained validation F1-score.

6.4.1 Quantitative Analysis

In order to verify the proposed hypothesis, we compare the proposed model, i.e., using different self-attention layers for each modality (mathematical elements and natural language) with a modified model, using only one self-attention layer for all parts of the text. This modified model is obtained by replacing the layers inside the dotted rectangle from Figure 6.2 with a single self-attention layer. This modified model is referred here as **Self-attention + BiLSTM**.

Results

Table 6.2 presents the results for the premise selection task using the random examples.

The aggregate scores obtained using STAR is consistently higher than the baseline. With the addition of more negative examples, a degradation is expected, given the

		Val			Test		
	n	**F1**	**P**	**R**	**F1**	**P**	**R**
STAR	1	**.885**	.854	.917	**.882**	.865	.899
	2	**.836**	.803	.871	**.829**	.793	.870
	5	**.765**	.693	.853	**.765**	.706	.835
	10	**.695**	.614	.799	**.684**	.603	.791
Self-att + LSTM	1	.651	.550	.796	.631	.573	.703
	2	.514	.406	.702	.514	.420	.663
	5	.493	.372	.728	.461	.344	.700
	10	.408	.283	.731	.406	.276	.766

Table 6.2: Comparison of STAR with a model containing a single self-attention layer.

introduction of extra noise in the input. Even though there is an expected degradation, STAR still outperforms the baseline in all cases, demonstrating robustness to noise. These results support our hypothesis that different modalities inside the mathematical text should be represented in different linguistic spaces.

Similarly, we re-run both models, but this time using the similar examples. The results can be found in Table 6.3.

		Val			Test		
	n	**F1**	**P**	**R**	**F1**	**P**	**R**
STAR	1	**.798**	.725	.886	**.793**	.723	.879
	2	**.716**	.624	.840	**.707**	.593	.875
	5	**.620**	.485	.857	**.626**	.493	.854
	10	**.546**	.412	.809	**.528**	.387	.834
Self-att + BiLSTM	1	.648	.561	.767	.538	.699	.437
	2	.537	.444	.679	.540	.448	.678
	5	.389	.261	.760	.379	.251	.773
	10	.289	.179	.759	.286	.174	.799

Table 6.3: Results for STAR and baseline for different number of negative examples (n) using similar examples.

We can notice that STAR precision decreases when compared with the results obtained using the random examples. However, once more, STAR outperforms the baseline for all values of n. The results of the baseline model do not change significantly from the previous result improving it in some cases. We hypothesise that this is due to the fact that the use of lexical similarity for the generation of similar examples does

not provide reliable discriminators (due to the limited intrinsic semantics of variables). Variables can have the same lexical form across mathematical statements, without sharing the same meaning.

Transferring Knowledge across mathematical domains

Another targeted hypothesis is that STAR performs better than the baseline in the task of transferring knowledge between different mathematical domains. In order to verify this hypothesis, we train the baseline and our model using one topic and test it in a different one. The topics used are Abstract Algebra (AA), Topology (TP) and Set Theory (ST). Table 6.4 presents the number of statements for Train/Val/Test for each topic.

Topic	Train	Val	Test
AA	2,246	633	580
ST	1,897	618	590
TP	2,539	810	788

Table 6.4: Distribution of dataset with different topics.

Table 6.5 shows the experimental results for the different mathematical topics. Initially, we expected that training using the largest dataset would allow both models to obtain the best performance. However, training using the Topology dataset topic did not achieve the highest results. This is likely because of the distinctive nature of its symbolic space, more focused on the properties of geometric objects. On the other hand, the best performing training and test dataset, Abstract Algebra, is heavily based on the algebraic notation that our model is capable of capture using cross-modal attention.

In terms of transferable knowledge, Set Theory is the tested dataset with the highest score, confirming the expectation that Set Theory is an important component of both Abstract Algebra and Topology, being an intrinsic part of the mathematical argumentation on these topics. Therefore, such knowledge is more natural to transport. Our proposed model outperforms the baseline in all cases. However, both models see substantial performance degradation when trying to transfer the knowledge from one topic to another, indicating both the need for better abstractive mathematical models and an intrinsic domain-specificity mathematical inference.

Topic		STAR			Self-att + BiLSTM		
Train	**Test**	**F1**	**P**	**R**	**F1**	**P**	**R**
AA	AA	**.862**	.823	.906	.629	.581	.684
TP	TP	**.752**	.692	.825	.722	.680	.769
ST	ST	**.787**	.763	.813	.613	.654	.578
AA	ST	**.662**	.595	.747	.627	.595	.664
AA	TP	**.595**	.520	.693	.570	.539	.605
TP	AA	**.654**	.536	.836	.649	.602	.704
TP	ST	**.673**	.588	.787	.628	.561	.714
ST	TP	**.535**	.539	.531	.578	.535	.627
ST	AA	**.644**	.598	.697	.625	.591	.663

Table 6.5: Testing the transferability across different mathematical areas. For these experiments, we use random examples with $n = 1$.

Other baselines

In order to verify the model performance, we test our model against two state-of-the-art models. The first baseline is a Transformer-based model, BERT. We fine-tune BERT [85] using the same configuration as the one used for Natural Language Inference [42] since this task carries similarities with the premise selection task. The other baseline is MathSum [99]: an encoder-decoder model used to represent mathematical content found in online forums. We use only the encoder part of this model, together with the same siamese network as STAR and the same parameter configuration. The results can be found in Table 6.6.

	Val			Test		
	F1	**P**	**R**	**F1**	**P**	**R**
BERT	**.886**	.871	.901	.877	.925	.834
MathSum	.644	.512	.869	.459	.562	.388
Self-attention + BiLSTM	.651	.550	.796	.631	.573	.703
STAR	.885	.854	.917	**.882**	.865	.899

Table 6.6: Comparison of our model with other baselines, using n=1 and random examples.

Considering the F1-Score obtained, BERT was placed second in the test set evaluation. Even though BERT is not explicitly trained for the Mathematical domain, it

Conjecture	Premise	Predicted	Label
Let $T = (S, \tau)$ be a topological space. Let A, B be subsets of S. Then: $\partial(A \cap B) \subseteq \partial A \cup \partial B$ where ∂A denotes the boundary of A.	Let S, T_1, T_2 be sets such that T_1, T_2 are both subsets of S. Then, using the notation of the relative complement: $ST_1 \cap T_2 = ST_1 \cup ST_2$	1	1
$\int \frac{\mathrm{X}}{x(x^2-a^2)} = \frac{1}{2a^2}, \ln \frac{x^2-a^2}{x^2} + C$ for $x^2 > a^2$.	$\int \frac{dx}{x} = \ln x + C$ for $x \neq 0$.	1	1
Let $T = S, \tau$ be a compact space. Then T is countably compact.	Let $T = (S, \tau_{a,b})$ be a modified Fort space. Then T is not a T_3 space, T_4 space or T_5 space.	1	0

Table 6.7: Some of the premises existing in the dataset, together with the predictions from STAR.

presents an excellent performance for the premise selection task. BERT is a large-scale model that was also trained on sources containing mathematical notation, including latex notation, therefore it partially encodes mathematical notation. Our model outperforms BERT for the test set, even though it employs a significantly smaller set of parameters (5x less parameters) and is not pre-trained on a large corpus as BERT is.

6.4.2 Qualitative analysis

We present examples of predicted pairs in Table 6.7. When analysing the obtained classified pairs, we found that STAR not only can represent heavily equational statements, such as the second pair from the table, but it can also handle statements that contain a high level of entanglement between mathematical and natural language terms, such as the first pair.

However, we found that STAR, for some examples, incorrectly matches statements based on variable names. For example, in pair 3, the variable T co-occurs across the statements. The model infers that this implies that there is a relation between both statements. The relationship exists since both statements refer to the concept *spaces*; however, this does not define a dependency relationship. This result provides evidence for the need of an architecture which better represents variable semantics.

Figure 6.5 presents a comparison of our model with the single attention model. This graph shows the percentage of mathematical elements in the statement versus the percentage of the statements in the dataset that the model was able to predict correctly.

STAR has a consistent performance throughout different distributions of mathematical and natural language terms. Such results demonstrate the need of an attention layer for each term modality. On the other hand, we can observe that the baseline obtain lower number of predictions when predicting statements that are mostly mathematical (right-end of the graph), finding it easier to predict the statements which have the prevalence of natural language terms (left-end of the graph). The results show that

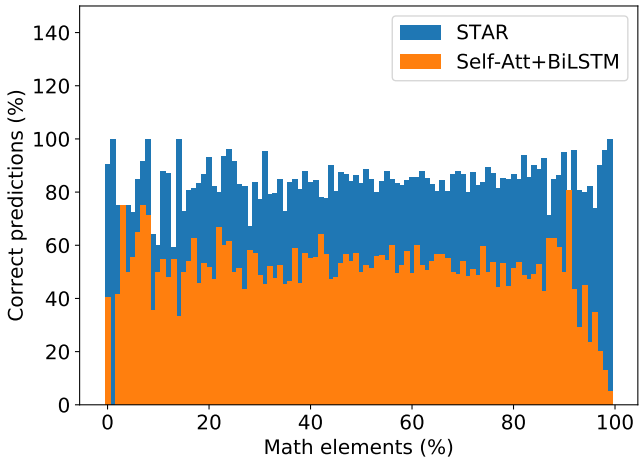


Figure 6.5: Comparison of our model and the baseline on the capability of predicting correctly statements with different levels of entanglement.

our model is better suitable for dealing with this type of entangled text.

6.5 Discussion

In this chapter, we introduced STAR, a model to represent mathematical statements for the task Natural Language Premise Selection. In this model, we used two layers of self-attention, one for each language modality present in the mathematical text.

In order to test STAR's ability to encode different linguistic aspects of each modality, verifying if it can interpret that expressions and words belong to different lexical spaces, we compared our performance with other baselines. Considering **RQ2** (*Can we improve the performance of deep learning models for inference over mathematical text by generating specialised encodings for mathematical natural language elements?*), we found that having one layer for each modality significantly increases the performance for premise selection. We also compared our approach with state-of-the-art models and found that STAR achieves the highest results for the Test set. STAR was also tested for transfer learning, revealing that cross-modal attention improves the transportability between different mathematical areas.

This chapter also presents an extensive ablation study, comparing how the multimodal components are crucial for the model's performance. Encoding the two modalities into two different embedding spaces allows the model to obtain higher scores, perform better in generalisation tasks and learn how to handle heavily entangled text

better.

However, we discovered that STAR still has some limitations:

- This method is still limited regarding the representation of variable semantics.

  There is still a gap in handling variable typing in latent models, considering its co-referent types and properties instead of its lexical representation. This was highlighted during our qualitative analysis, where the inference is sometimes influenced by variables that have the same, despite having different semantics.

- The inference process only considers encoding at statement-level.

  Unlike the previous method proposed in Chapter 5, the encoding performed using STAR is only able to consider the information at a statement-level since there is no mechanism to capture the structure of the relationship between statements. While the previous chapter dealt with a retrieval problem, while this one deals with an inference problem, the ideas of structural similarity are explored in the following chapter to encode information at both statement and structure level for the premise inference task.

6.6 Reproducibility Statement

All experiments and data can be found in the Github repository `http://github.com/ai-systems/crossmodal_embedding`. The hyperparameters used for STAR were:

- LSTM Hidden Size = 512;
- LSTM output size = 512;
- Number of attention heads = 4;
- Weight decay = 0.01;
- Number of epochs = 50;
- Batch Size = 32;
- Learning rate = 2e-4;
- Embedding layer size = 100;

Chapter 7

Connecting mathematical statements

Considering our findings from Chapter 5, where we analysed the structural similarities between different statements, in this chapter we implement a model based on that proposed concept. This chapter describes a new model for solving the NLPS inference task using a Deep Graph Convolutional Neural Network to build a network of related mathematical statements. In this chapter we are able to further explore **RQ4** (*Can the dependencies between mathematical statements provide a simplified representation of mathematical arguments?*) and provide insights into **RQ5** (*How to encode the inter-statement dependency structure to support inference for the natural language premise selection task?*).

7.1 Motivation

We propose a method to solve the natural premise selection task, representing all conjectures and premises as nodes and the dependencies between the corresponding statements as edges, formulating the problem as a *link prediction* problem. We hypothesise that graph-based embeddings are suitable structures for representing and detecting the dependencies between different mathematical statements. We then use Deep Convolutional Graph Neural Networks [103] over a structural and content-based encoding of proofs in order to obtain the set of relevant premises for proving a statement.

In order to evaluate this task, we use the dataset PS-ProofWiki, in the same setting as the previous chapter. The performance of the proposed model is compared to a set fine-tuned pre-trained BERT model for the task, formulating the problem as a binary classification between conjectures and premises, in the same setting as the previous chapter.

Contrasted to the previous chapter, the model proposed on this chapter targets capturing both content (local) and structural dependencies (global) across natural language mathematical statements and its evaluation on the natural language premise selection problem.

In the next sections, we describe the proposed model for addressing the premise selection task. The proposed model uses a Deep Graph Convolutional Neural Network (DGCNN) for solving the premise selection task as a link prediction task [102]. The proposed model aims to encode the natural language and the formulae terms as well as the dependencies and graph-structural patterns of the mathematical text.

7.2 Encoding mathematical propositions and premises

7.2.1 Graph construction

In Mathematics, theorems are always built on top of previous mathematical knowledge, such as lemmas, corollaries, definitions and other theorems. Thus, Mathematics as a discourse intrinsically entails a network structure. With this hierarchy and interlinking of concepts in mind, we developed a graph representation to represent all mathematical statements present in the corpus and their associated dependencies.

The extracted dependency graph is a directed graph $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of vertices, composed by mathematical statements and $\mathcal{E}$ is a set of ordered pairs of vertices (edges), in this case the relationship between mathematical statements. If $m_1, m_2 \in \mathcal{V}$ and $(m_1, m_2) \in \mathcal{E}$ that means the statement m_1 is a premise to the statement m_2.

7.2.2 Subgraph extraction

From the set of graphs containing all asserted dependency relations, an enclosing subgraph (with a fixed hop h size of $1 \leq h \leq 2$) is extracted by selecting a pair of nodes as the target. These pair of nodes will be used to define the link prediction classification context, in which a binary class is assigned, P when $(m_1, m_2) \in \mathcal{E}$ and NP (not a premise) otherwise (Figure 7.1).

As we predict the link between different statements, we are also predicting the dependencies between different statements, therefore, addressing the natural premise selection problem.

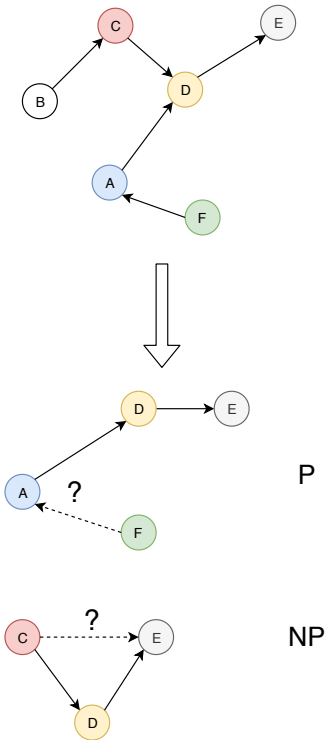


Figure 7.1: Sub-graph extraction for link prediction.

7.2.3 Node features

Every node $m_i \in \mathcal{V}$ is composed of two parts: (1) a label based on a function which encodes its neighbourhood, (2) an embedding of its textual content.

The framework generates labels for the nodes using the *Double-Radius Node Labelling* (DRNL) [102] mapping.

The embedding of the textual content is an embedding of the mathematical statements. A mathematical statement is composed of a hybrid setting of mathematical notation and natural language statements. Paragraph Vector Distributed Memory (PV-DM/Doc2Vec) [48] was used to encode a statement-level representation of the constituent statements of the proof (where each statement is a 'paragraph'). The expressions and equations are encoded as a tree, by representing every sub-expression as a token. For example, the expression '$(x+y)*c$' is represented as the sequence of tokens ['x','y','$(x+y)$','$(x+y)*c$'], capturing the syntactic structure of the mathematical expression. The same model captures both the natural language and the mathematical elements tokens. Figure 7.2 depicts how the structural and content aspects are represented.

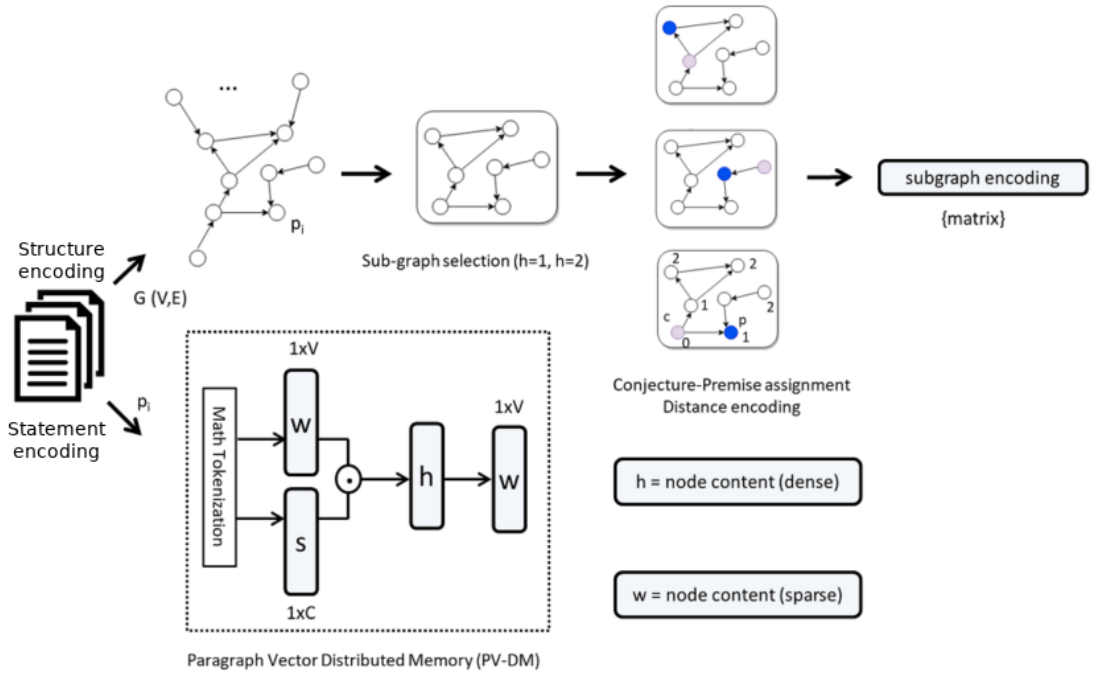


Figure 7.2: Pre-processing workflow of the proof corpus.

7.3 Proposed Model: Premise Selection based on DGC-NNs

7.3.1 Design Principles

A Deep Graph Convolutional Neural Network (DGCNN) architecture [103] was used as the default GNN engine of the premise selection. The architecture was selected due to its ability to encode network features with a consistent performance across different graph network (GN) evaluation scenarios. Moreover, we use the graph encoding proposed in [102], which aims for learning subgraph structural patterns using DGC-NNs. This method embeds the learning of a problem-specific graph heuristic function (which is formalised as the γ-decaying heuristic theory). This can be contrasted with the use of pre-defined methods from a single heuristic framework (such as Katz index, PageRank and SimRank [102]), by using a graph-specific approximation instead.

The underlying assumption behind the selection of the base architecture is that the premise selection problem requires the encoding of both the statement content and of the graph-dependency patterns.

The final problem of premise selection is rephrased as a problem of link prediction, and the final classification layer has a binary classifier. Figure 7.3 depicts the main

components of an end-to-end architecture.

7.3.2 Detailed Model

A denotes the adjacency matrix of a graph, n the number of vertices where each vertex has a c-dimensional feature vector, denoted as $X \in R^{n \times c}$. For a vertex v, we use $\Gamma(v)$ to denote the set of v's neighbouring nodes. DGCNN uses the graph convolution function:

$$Z = f(\tilde{D}^{-1}\tilde{A}XW) \tag{7.1}$$

where $W \in \mathbb{R}^{\times}$ is a weight matrix of graph convolution parameters, $\tilde{A} = A + I$ based on the adjacency matrix A, $\tilde{D}$ is a diagonal degree matrix [102] and f is a non-linear activation function. $\tilde{D}^{-1}\tilde{A}$ is a propagation matrix.

The graph aggregation layer builds for each node a graph-level feature vector based the individual node states, which is defined by:

$$Z_i = f(\frac{1}{|\Gamma(i)|+1}[X_i W + \sum_{j \in \Gamma(i)} X_j W]) \tag{7.2}$$

The graph convolution aggregates node patterns, extracting local subgraph patterns. The last graph convolution layer output can be used to sort the graph vertices in an order which reflects the vertices structural roles [102].

After the aggregation, the DGCNN uses a *sort pooling layer*, which sorts the final node states based on to the last graph convolution layer's output [102]. The sorting criteria are based on a topological-based ordering. For example [57] provide a labelling scheme for vertexes based on topological patterns. This topological ordering is consistent across graphs: vertices in two different graphs will be assigned similar relative positions if they have similar structural roles [103].

The ordering operation is followed by a *max-k pooling operation* which creates a representation for the different graphs with uniform dimensions (truncating or extending into k dimensions). This allows the application of a *1-D CNN layer* on the node sequence. A *final dense layer* connected to a *softmax layer* performs the binary classification of the target vertices into the *premise/non-premise case*.

A standard DGCNN configuration is used [103], containing four graph convolution layers, a sort pooling layer with a k assignment 0.60 (graph coverage), two 1-D convolution layers and a dense layer with 128 neurons.

The proposed model has a locality assumption expressed at the statement encoding

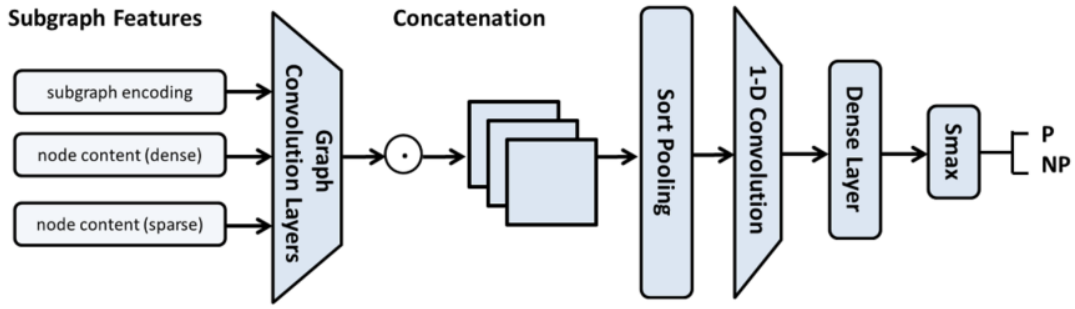


Figure 7.3: Depiction of the DGCNN architecture used in the premise selection task.

level, which limits the proof neighbourhood to two hops. This follows the intuition that the premise selection model aims to reflect the mentioned structure of proofs (expanding, however an additional hop) privileging the classification of closer and more specific conjecture-premise relations. More exploratory types of proofs may require the expansion of the hops to cope with longer distance relations.

7.4 Experiments

This section evaluates the performance of the proposed model using PS-ProofWiki. These are further expanded using a transformer-based architecture (BERT), due to its state-of-art results for the encoding of sentence-level embeddings and their use in tasks such as natural language inference.

For the experiments using BERT and the proposed method, we split the dataset using a 50/20/30 (train/dev/test) split. We run all experiments ten times, evaluating on the test set, and report the average Precision, Recall and F1-score, given that, similar to the previous chapter, we are dealing with an inference task. All evaluation data, as well as the experimental pipeline, can be found online[1] for reproducibility purposes.

7.4.1 Baseline: BERT

In order to use BERT, we reformulate this problem as a pairwise relevance classification problem, as done previously in the context of ATP systems. We have a set of mathematical statements S, a set of conjectures C and a set of premises P, where

[1] https://github.com/ai-systems/premise_selection_graph

$C \subseteq P$, $C \subseteq S$ and $P \subseteq S$. Considering a conjecture $c \in C$ and a premise $p \in P$, a function $f(c,p)$ is defined, where $f(c,p) = 1$ if p is a part of the proof of c and $f(c,p) = 0$ otherwise.

For this experiment, we used the pre-trained BERT model *bert-base-uncased*, fine-tuning it for the target task with a sequence classifier, adding a linear layer on top of the transformer embeddings.

7.4.2 Quantitative analysis

The dataset is imbalanced by the nature of the natural premise selection inference problem. In order to solve the natural premise selection task, any method would have to be able to handle a large number of negative examples. There are 10k different possible premises, and some conjectures are only connected to one premise, creating a large number of negative pairs in our dataset, requiring the definition of a cap for the number of negative samples. In order to provide a more constrained setting, we define a subset of the PS-ProofWiki, named PS-ProofWiki$_{TRIG}$ targeting trigonometric functions.

Table 7.1: Precision (P), recall (R), and F1-score (F1) for the BERT baseline and the proposed method, with 30 negative examples for each positive case (values are multiplied by 100).

	BERT			Proposed Model		
	P	R	F1	P	R	F1
PS-ProofWiki$_{TRIG}$	**39.9**	22.9	29.1	34.0	**50.0**	40.5 (+ 39%)
PS-ProofWiki	**47.1**	26.7	34.1	**48.5**	47.7	48.1 (+ 41%)

The proposed method outperforms the BERT-based model by 41% in terms of F1-score, as shown in Table 7.1. We hypothesise that the encoding of the structural patterns of the dependency relations in addition to the content-based similarity better captures the semantic nature of the proof (fundamental to interpret a proof by its neighbourhood).

7.4.3 Scalability & Imbalance Robustness analysis

In order to evaluate the robustness of the proposed method and the baseline with regard to an increase in imbalance (reflecting a notion of *scalability* of the quality of the

inference within the KB), we compare how the F1-score changes as we add more (random) negative examples to the dataset.

Figure 7.4a and Figure 7.4b presents a comparison between BERT and our method for the PS-ProofWiki$_{TRIG}$ and the PS-ProofWiki datasets, respectively.

The results indicate that the BERT-based classifier performance degrades faster as we increase the number of negative samples in the dataset. For $n = 30$, the F1-score reaches a value of almost zero. In contrast, the proposed model presents a significantly slower decline (25%), showing better scalability properties in the context of the premise selection problem.

Finally we experiment on how BERT and the proposed model compares when we consider transitivity between premises (n-hop relations), using PS-ProofWiki$_{TRIG}$ and 10 negative examples for each positive example. We report the results in Table 7.2, where we can see that the proposed model obtains better overall performance as the number of hops is increased. These results reinforce the architectural design supported by graph-based models.

Table 7.2: Comparison of BERT and the proposed model for different levels of transitivity between premises (values are multiplied by 100).

	BERT			Proposed Model		
	P	R	F1	P	R	F1
2-hop	47.5	**78.9**	59.3	54.8	**68.7**	61.0 (+ 3%)
3-hop	41.0	45.1	**49.2**	58.8	**63.3**	61.2 (+ 24%)

7.4.4 Qualitative analysis

From the results obtained from our model we observed that the model struggles to encode statements which are centered around pure equational (formulae) content. Embeddings for mathematical symbols should take into consideration more specific semantics of operators: such semantics is not obtained using PV-DM (Doc2Vec) or BERT. This provides evidence on the need for more principled structural embeddings for mathematical formulas, which could most certainly improve the prediction of future work in the natural premise selection task.

Even though BERT is not trained in a mathematical corpus, it still obtains relevant results, hinting that training BERT on a mathematical corpus could achieve better results. However, this task is outside the scope of this work and will be left for future work.

The proposed DGCNN-based model is capable of finding structural patterns between the statements and to reinforce content-based semantic evidence. We observed that statements that are similar in content, commonly have a significant intersection of premises, as a result of the graph embedding, the DGCNN-model is able to better discriminate more fine-grained semantic cues better.

7.5 Discussion

In this work, we introduced a method for natural language premise selection (finding relevant theorems, axioms and definitions) in large natural language mathematical texts. The proposed method, which uses Deep Graph Convolutional Neural Networks (DGCNNs), combines both structural and content elements of mathematical statements for addressing the premise selection problem as a link prediction classification problem. Results show that the method outperforms a BERT-based baseline by 41% in F1-score. Moreover, the proposed model shows significantly lower F1-score degradation concerning class imbalance, a fundamental desirable scalability property for the problem of premise selection.

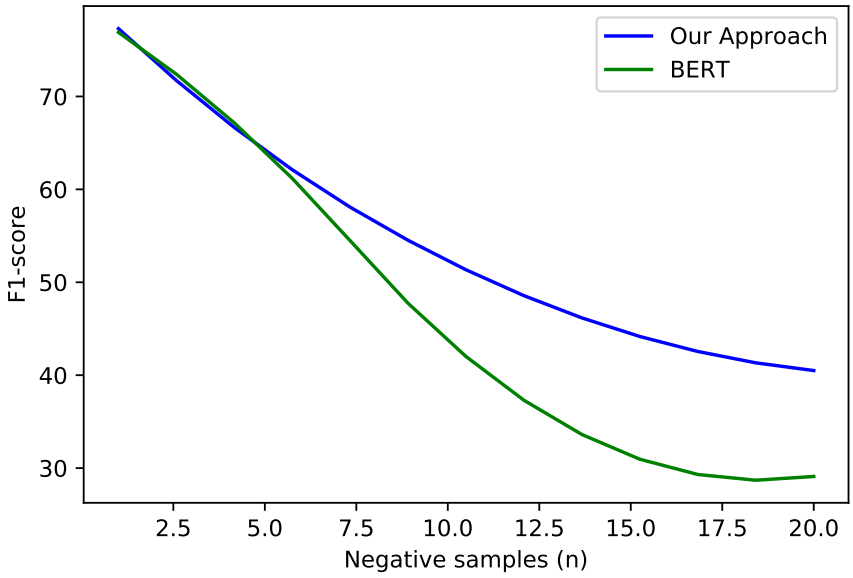
Going back to **RQ4** (*Can the dependencies between mathematical statements provide a simplified representation of mathematical arguments?*), we found that the results reported in this chapter support the idea previously introduced in Chapter 5, where the structural similarity and dependencies is a fundamental component of encoding inference in the context of a NLPS task. We also found that by leveraging Deep Learning architecture applied to Graph structures, we are able to obtain better performance in both a single hop and multi-hop scenario, providing answers to **RQ5** (*How to encode the inter-statement dependency structure to support inference for the natural language premise selection task?*). The qualitative analysis indicates that there is a demand to design principled embeddings for better capturing the semantics of proofs that are denser in mathematical formulae, reinforcing the results obtained in Chapter 6. In the next chapter, we explore how to explicitly design a model that can better encode variables.

Our proposed method has one major limitation: it does not scale well to larger and imbalanced datasets. This model requires the encoding of entire sections of the graph simultaneously in memory. Due to the large size of this graph, when adding several negative premises to the graph, the problem becomes infeasible since representing all premises would lead to impractically considerable memory complexity. For cases when scalability is crucial, methodologies as the one proposed in Chapter 5 are more

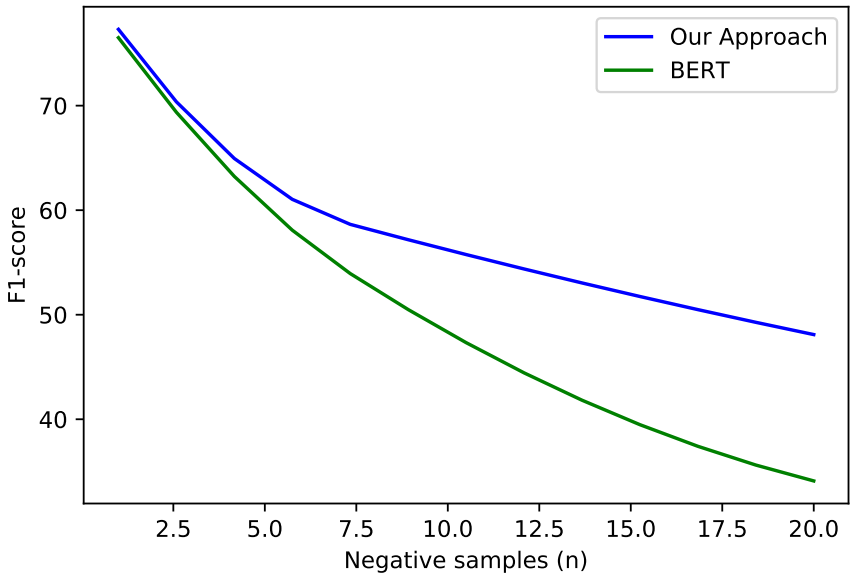
suitable.

7.5.1 Reproducibility Statement

The repository containing all experiments from this chapter are found in `https://github.com/ai-systems/premise_selection_graph`. For the DGCNN, we used the implementation avaiable in `https://github.com/muhanzhang/DGCNN`. The `doc2vec` encodings were obtained from the Python library gensim.



(a) Evaluating on PS-ProofWiki$_{TRIG}$



(b) Evaluating on PS-ProofWiki

Figure 7.4: Comparison of the proposed model and BERT, showing how both models perform (in terms of F1-score) when adding more negative examples to the training and test set.

Chapter 8

Improving Variable Understanding

The articulation of mathematical arguments is a fundamental part of scientific reasoning and communication. One of the particular linguistic elements used for such argumentation is *variables*: they are essential for expressing complex mathematical ideas, allowing scientists to refer to a set of values compactly and rigorously. Given the essential nature of variables, models that perform inference over scientific and mathematical text should be able to meaningfully represent and leverage such elements to understand mathematical language and improve downstream inference performance. In this Chapter, we investigate the question proposed in **RQ3** (*How to encode the semantics of variables at sentence-level representations?*), which concerns the representation of variables in pre-trained language models.

8.1 Motivation

While there is still debate on a universally accepted definition for variables, their functional aspects within mathematical text are well established. A variable is usually defined as a symbol standing as a referent for a set consisting of at least two elements [61]. Given their nature of abstract and dynamic referents, two aspects make the representation of variables particularly challenging in the field of NLP: (i) The meaning of a variable is exclusively determined by its context [73], a variable carries no meaning when considered in isolation; (ii) The same variable symbol (e.g., the variable x) can be reused in an unlimited number of sentences and expressions, possibly assuming different meanings and referring to different sets of values while keeping the same name. While a similar problem can also be found for the representation of words (i.e., word sense disambiguation), the scale of ambiguity is more marked, due to the

fact that variable names, unlike words, are not grounded a priori to any particular set of concepts.

In order to understand mathematical text, the meaning of variables (i.e., their type) needs to be implicitly or explicitly inferred at some point from the text. Identifying and qualifying the binding between variables and their types, therefore, is crucial for reasoning with mathematical text, given that any form of inference on a variable is fundamentally constrained by the possible values it can take, and those values are uniquely determined by its type. For example, we can only correctly predict the entailment between two sentences containing variables if we infer their types beforehand.

As a benchmark for variable comprehension in mathematical text, the variable typing task [80] requires finding the connections between variables and their respective types in a sentence. Despite its importance for scientific inference, the task is still widely unexplored. Most of the existing work focusing on the representation of mathematical elements, in fact, has been carried out in a non-textual setting, where the mathematical elements are represented independently from any textual content [1]. While some expressions and formulas are universal enough to be encoded without context (e.g., Pythagoras theorem), variables have no meaning in isolation and different symbols can be arbitrarily chosen to indicate variables with the same meaning. Therefore, an important principle for designing and evaluating a variable representation is that such a representation should be agnostic to the specific symbols adopted as referents in the text – i.e., the variable names. This is because the variable names contained in a generic passage can be opportunely renamed without altering the sentences' meaning. For instance, a robust variable representation should be able to encode the sentences "Let x be an integer" and "Let y be an integer" in exactly the same way, if they are inserted in the same context.

However, while this characteristic seems to be intrinsic in the nature of variables, it has been largely ignored by current evaluation frameworks, where there is no agnostic way to verify the quality of the generated variable representation independently of their surface form. To move a step forward towards more robust and generalisable representations, this work proposes a new testing and modelling framework based on the property of *generalisation to variable renaming*. Specifically, we define a model to be generalisable to variable renaming if the model's performance does not decrease when renaming variables in the test set with variable names never seen during training. Through testing such a property, since the renaming of variables does not alter the context or meaning of the sentences, we are verifying whether the context is correctly

moulding the variable encoding and, at the same time, whether the performance is not due to overfitting to the surface form of the text.

To address and study generalisation to variable renaming, this work proposes **VarSlot** (Variable Slot), a model for variable typing that represents variables in a surface agnostic way. To achieve a generalisable representation, VarSlot initialises variables as blank slots and employs a customised Slot Attention mechanism [53] to iteratively specialise the representation. Specifically, VarSlot leverages self-attention to conform the representation of each variable to its context (i.e., its surrounding words and other variables), where each surrounding element has a different influence on the final representation. Experiments adopting VarSlot to extend sentence embeddings based on Transformers [25] demonstrate not only that the proposed framework is able to achieve state-of-the-art results on the variable typing task, but also, in contrast with previous work, that VarSlot allows for better generalisation to variable renaming. To the best of our knowledge, this is the first work that focuses on generalisation and robustness for variable representation in mathematical text, providing, at the same time, a critical analysis of large language models (LLMs) targeting the scientific domain.

8.2 Variable Typing problem

This work follows the same definition of a variable used in [80], considering as variables *simple expressions* composed of a single, possibly scripted, base identifier. The *variable typing task* requires the assignment between variables in the sentence and its respective types. We use the same setting as [80] for this task: given a sentence s with a pre-identified set of variables V and types T, the task is defined as a binary classification of all edges $V \times T$, where a positive edge means that the variable is assigned that type and negative otherwise. Figure 8.1 introduces an example of a mathematical statement, containing one variable b with type *persistence length*. The edge linking to this type is a positive one, while the one linking to type *chains* is a negative edge.

While the task carries some similarity to two other well-known tasks in NLP, *Coreference Resolution* and *Relation Extraction*, there are some fundamental differences. Variables have a particular behaviour, where there is a disconnection between the variable name and its meaning. Also, the variable typing task mainly involves intra-sentence dependencies and does not rely on a predicate-argument relation.

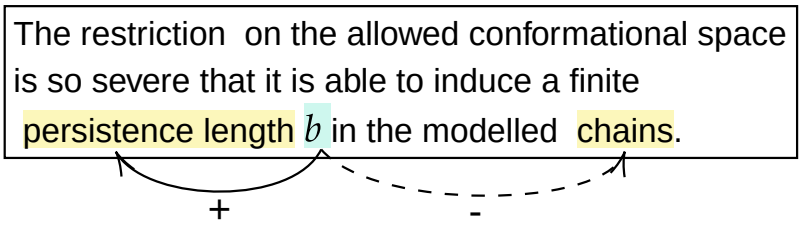


Figure 8.1: Example of a mathematical statement containing one variable. This example shows two different edges (variable→type), with a positive and a negative binding.

8.3 Proposed Method: Variable Slots (VarSlot)

Following the recent literature [80], this work frames the variable typing problem as binary classification, where each variable is tested against all possible types. Given a sentence s from a mathematical text containing known variables $V = \{v_1, v_2, \ldots, v_{n_v}\}$ and types $T = \{t_1, t_2, \ldots, t_{n_t}\}$, VarSlot aims at finding a function such that $f(v_i, t_j) = 1$ if v_i has type t_j and $f(v_i, t_j) = 0$ otherwise. In this section we detail our method, **VarSlot**, illustrated in Figure 8.2.

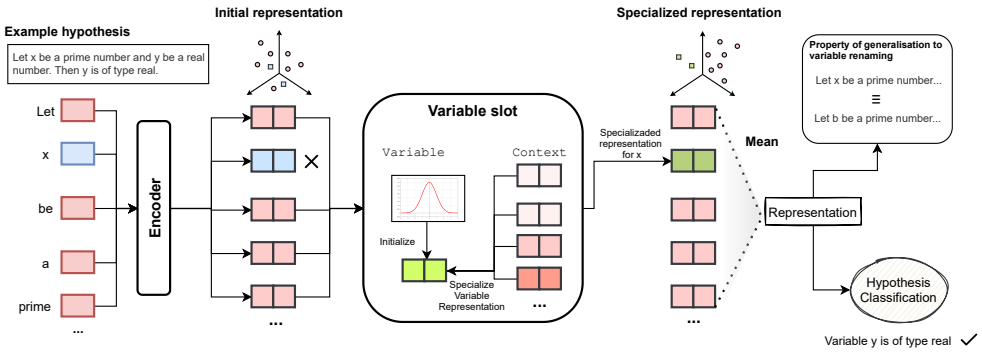


Figure 8.2: Our model takes as input a statement containing words and variables and obtain the classification of the hypothesis.

8.3.1 From types and expressions to hypotheses

A novel framing for the variable typing problem is proposed in this work. Instead of treating sentences, types and variables as different features of the problem [80], each sentence s_i is converted to a set of hypotheses $H_{s_i} = \{h_{(v_1,t_1)}, h_{(v_1,t_2)}, \ldots, h_{(v_{n_v},t_{n_t})}\}$ of size $n_v \times n_t$ by adding to the end of the sentence, the following phrase: "Then [variable] is of type [type]" where *variable* and *type* are replaced by the ones being evaluated at

that instance. For example, if one wants to test if the variable x is of type *integer* in the sentence "Let x be an integer and y be a real", it can be converted to a hypothesis by adding "Then x is of type *integer*" after the initial sentence. This modification will allow our method to leverage pre-trained sentence encoders, obtaining enriched representations.

8.3.2 Encoding the sentences

Previous research has shown that LLMs struggle to understand concepts such as numeracy [79] and solving math word problems [62], but there is still no research on the representation of variables inside the mathematical text for such models, despite their higher performance for different inference tasks [67].

We hypothesise that it is possible to leverage pre-trained sentence representations to obtain an initial encoding for each hypothesis. By using an encoder, each hypothesis h of size N is mapped to a representation $E \in \mathbb{R}^{N \times D}$, where each token is assigned a vector of size D, regardless of being a variable or a word. At this point, the representation is unable to distinguish between both modalities of elements.

8.3.3 Representing Variables with Slots

Variables represent a set of possible values, acting as a place-holder element for any possible value inside that set [73]. This set of values is attached to the type corresponding to that variable; for example, if a variable is typed as an integer, we expect it to take values from that set only. A variable with the symbol x can refer to completely different sets depending on its context. Therefore, when representing a variable, its *surface form* (or symbol) should not interfere with its representation; the semantics of the variable should be guided exclusively by the defined type. We aim to approximate this behaviour with Slot Attention [53].

Slots use a common representational format, where each slot can store any object from the input, rendering it a suitable candidate for obtaining a latent representation of variables since the same variable name can take many different contexts and possible types. Slot Attention was initially proposed for mapping a set of objects contained in an image to a set of output vectors referred to as slots. Slot Attention was not previously adapted to the field of NLP. In this work, instead of using a single slot to learn the representation of all needed elements, each variable instance has its own slot, learning an independent surface form agnostic representation.

Given a hypothesis $h_{(v_i,t_j)}$ containing the variables $V = \{v_1, \ldots, v_{n_v}\}$, a pre-trained sentence embedding is used to obtain a representation E for this hypothesis. In order to obtain a representation that can better distinguish from symbol and abstraction of the variables, we generate a new representation $E_{(v_i-)} \in \mathbb{R}^{N \times D}$ for each variable in the sentence, where the vectors representing the variables are all replaced by zeroes. For each variable, the representation obtained from the encoder is dropped, preserving only the other tokens. This step allows our model to learn the representation of the variables based on their context and abstraction, diminishing the weight from its surface features.

A representation $e_{(v_i)} \in \mathbb{R}^D$ is obtained for each variable through iterative Scaled Dot-Product Attention. First, in order to obtain an initial representation, we initialise $e_{(v_i)}$ by sampling from a Gaussian distribution $\mathcal{N}(\mu, \sigma)$, with learnable parameters $\mu \in \mathbb{R}^D$ and $\sigma \in \mathbb{R}^D$. This will generate an empty slot, which will be iteratively conformed into a representation for the variable v_i.

The results from Chapter 6 has shown that having a separated representation for mathematical elements and words can be beneficial for performing inference over mathematical text. We hypothesise that allowing our model to learn a new representation for variables will naturally separate it from other elements.

We apply the linear transformation $\boldsymbol{k}$ and $\boldsymbol{v}$ over $E_{(v_i-)}$ and $\boldsymbol{q}$ over $e_{(v_i)}$ to compute the Scaled Dot-Product Attention [85]:

$$\texttt{Att} = \texttt{softmax}(\frac{\boldsymbol{q}(e_{(v_i)})\boldsymbol{k}(E_{(v_i-)})^T}{\sqrt{D}})\boldsymbol{v}(E_{(v_i-)}) \tag{8.1}$$

The obtained attention is applied to a Gated Recurrent Unit [17] with hidden size D and transformed with a multi-layer perceptron (MLP) with ReLU activation in order to obtain the new value for $e_{(v_i)}$. This process is repeated for T iterations, for each variable in the sentence.

8.3.4 Classifying hypotheses

After obtaining the representation for each variable $e_{(v_i)}$, we need to match it again with the original representation. This is achieved by mapping the obtained variable representations to their original positions in $E_{(v_i-)}$. The final matrix is encoded by a BiLSTM layer, obtaining a final enriched representation $\mathcal{E}$ for the sentence, as shown in Algorithm 2.

Finally, we obtain a representation for the hypothesis as a single vector of dimension D by computing the mean over the rows of $\mathcal{E}$. In order to obtain the classification

Algorithm 2: Slot Attention for learning the representation of mathematical statements with variables.

Input: Encoded representation of the hypothesis E, variables in the hypothesis $v_1, v_2, ..., v_{n_v}$, and positions of each variable in the hypothesis $P_{v_1}, P_{v_2}, ...P_{v_{n_v}}$.
Output: New hypothesis representation $\mathcal{E}$.

$E_{(v_i-)} \leftarrow E$
for $j \leftarrow P_{v_0}$ ***to*** P_{v_n} `/* Discard variable representations */`
do
  $E_{(v_i-)_j} \leftarrow [0\,0\,\ldots\,0\,0]$
end
for $i \leftarrow 0$ ***to*** n_v `/* For all variables */`
do
  $e_{(v_i)} \sim \mathcal{N}(\mu, \sigma)$; `/* Initialise new representation */`
  for $t \leftarrow 0$ ***to*** T `/* Iterate over representation T times */`
  do
    $e_{(v_i)_{t-1}} \leftarrow e_{(v_i)}$
    $e_{(v_i)} \leftarrow \texttt{LayerNorm}(e_{(v_i)})$
    $K \leftarrow \boldsymbol{k}(E_{(v_i-)})$
    $Q \leftarrow \boldsymbol{q}(e_{(v_i)})$
    $V \leftarrow \boldsymbol{v}(E_{(v_i-)})$
    $A \leftarrow \texttt{Att}(Q, K, V)$
    $e_{(v_i)} \leftarrow \texttt{GRU}(e_{(v_i)_{t-1}}, A)$
    $e_{(v_i)} \leftarrow e_{(v_i)} + \texttt{MLP}(\texttt{LayerNorm}(e_{(v_i)}))$
  end
  for $j \leftarrow P_{v_{i0}}$ ***to*** $P_{v_{in}}$ `/* Replace with new representation */`
  do
    $E_{(v_i-)} \leftarrow e_{(v_i)}$
  end
end
$\mathcal{E} \leftarrow \texttt{BiLSTM}(E_{(v_i-)})$
return $\mathcal{E}$

for each hypothesis, we use a final linear layer, with as training objective function the *Binary Cross-Entropy Loss*.

8.4 Experiments

This section presents the experiments performed to evaluate the performance of VarSlot for the variable typing task. As the encoder for the model, we use the Sentence Transformers [65] version of SciBERT [11] pre-trained for NLI tasks (SciBERT-NLI)[1]. We compare our method with previous research and standard pre-trained language models. The evaluation is conducted in two different settings:

[1]The model `gsarti/scibert-nli` from Hugging Face is used.

1. **Classic Variable Typing**: In this setting, the variables in the Train/Dev/Test have the same surface form, i.e., the variable symbols are consistent across the different splits of the datasets. This setting represents the canonical evaluation of variable typing task, as it was initially described in [80].

2. **Variable Typing with Renaming**: In order to evaluate the model's ability to abstract from the surface form of the variables and generalise to variable renaming, we replace all the variables in the test set with new symbols, unseen in the Train and Dev set. For instance, if the Train/Dev only contains letters as variables, we apply substitution of all the variables by $x_1, x_2, ..., x_n$ in the Test set without modifying the sentence's meaning.

8.4.1 Dataset

The dataset used in this work to evaluate the performance of the proposed model is the Variable Typing Dataset [80]. This dataset was manually curated and annotated from mathematical statements contained in scientific papers.

While the original dataset only contained generated ids for the text's variable names (ex: using an id `<m:12>` instead of the original x term), we modified the dataset by replacing such tokens with common variable names. For the **Classic** setting, the Train, Dev, and Test split only contain variable names with single letters (i.e., $b, c, d, \ldots, z$). In this process, we were mindful not to include the letters a and i, considering those can be mistaken for English words. We also do not consider uppercase variable names, since some of the LLMs only are available in an uncased implementation.

For the **Renaming** setting, the Train and Dev set is the same as the previous setting, but the Test set contains only variable names in the format $x_1, x_2, x_3, ..., x_n$. The Test set in both settings has the same hypotheses with identical semantic meaning but different variable names. For example, the fragment "Let b be ..." becomes "Let x_1 be ..." in the Renaming setting. For reproducibility purposes, we make this expanded dataset available in our repository.

8.4.2 Baselines

We compare our method to the following models:

- **Stathopoulos et al. [80]**: This architecture is based on a Bidirectional LSTM.

The model uses one string feature, which is referred to as *supertype*. If the token is a type, then this feature is the string key of the embedding vector of its supertype or *NONE* otherwise. These features are mapped to a separate embedding space and then concatenated with the word embedding to form a single task-specific word representation.

- **BERT [25], RoBERTa [51], SciBERT [11] and MathBERT [60]**: To assess the understanding of variables in pre-trained language models, we use the mentioned models as baselines for the Variable Typing task. For all models we use the Base and uncased version. The models are fine-tuned as a classic NLI model, using a `[SEP]` token to separate the original sentence with our new typing sentence. All models are fine-tuned with batch size of 32, for 10 epochs.

8.4.3 Quantitative results

Table 8.1 presents the results for the Variable Typing task in both the classic setting and the renaming setting. We include here our method with $T = 3$ and $T = 2$. We will start our discussion with the results from the classic setting and then move to the renaming setting.

Model	Dev (Classic)			Test (Classic)			Test (Renaming)			
	P	R	F1	P	R	F1	P	R	F1	Decrease
Stathopoulos et al. [80]				83.10	74.70	78.90				
BERT	84.47	80.96	82.68	77.8	82.98	80.31	54.00	78.23	63.89	20.44%
RoBERTa	83.95	78.99	81.39	82.32	80.13	81.21	55.43	68.34	61.21	24.62%
MathBERT	83.33	70.02	76.09	81.85	73.76	77.59	54.37	44.29	48.82	37.07%
SciBERT	83.84	84.02	**83.93**	77.26	87.54	82.08	30.25	93.15	45.67	44.35%
VarSlot (T=3)	84.94	78.99	81.85	83.18	81.36	**82.26**	60.86	73.47	66.58	19.06%
VarSlot (T=2)	86.82	77.89	82.12	83.95	79.08	81.44	69.80	79.04	**71.86**	**11.76%**

Table 8.1: Comparison of our method with different baselines for Variable Typing for both the Classic and Renaming Setting.

Classic Setting

Considering first the classic setting, we can observe that both BERT and RoBERTa achieve good results, outperforming the method proposed by [80], which was designed explicitly for variable typing and uses specific typing embedding. Such performance does not come as a surprise since, as discussed previously, variable typing carries some

similarity to tasks that these models have excelled in the past. However, such performance does not imply that these models have a good understanding of the meaning of variables; most likely, they are leveraging the syntactic knowledge they possess to connect types and variables.

MathBERT and SciBERT are models specialised in scientific and mathematical knowledge. While SciBERT is pre-trained using corpora from several scientific disciplines, MathBERT was exclusively trained on mathematical text. We initally expected both models to excel on this task since they have been previously exposed to mathematical notation. However, as seen from our results, MathBERT was the worst-performing model from our baselines, while SciBERT was the best performing one. While we cannot establish the reasons for MathBERT's lower performance, since this is outside the scope of this work, we hypothesise that training a model with a large amount of mathematical notation without explicitly designing an encoding which reflects its abstract semantics can be damaging to a model's performance. For example, many variable names are reused across different mathematical disciplines, and as we will see in the renaming setting, most models cannot abstract variable meaning from their surface form. SciBERT has been exposed to a smaller mathematical corpus, and usually inside non exclusively mathematical contexts (e.g., computer science papers). Therefore, it is able to achieve higher performance. Given the results obtained from SciBERT, we decided to use the Sentence-Transformers version of this model as our encoder.

We can see that for $T = 3$, VarSlot can outperform all of the models for the classic setting. Even though VarSlot uses SciBERT as part of its model, we can still see an improvement when combining it with Slot Attention. For $T = 2$ we still obtain competitive results, being outperformed only by SciBERT.

Renaming setting

In this setting, we have the same sentences as in the previous setting, but the variables in the Test split have now different names. Considering the semantic of variables, the previous results should not change, considering the sentence's meaning remains untouched; only the surface of the variables have been altered. However, the obtained results prove otherwise. For all the models, there is a decrease relative to the results obtained in the classic setting. Such results hint at the fact that the models still do not encode the expected variable behaviour.

Looking at the obtained results, we can notice that the results obtained using SciBERT suffers from a more remarkable decrease, with a 44.35% (82.08→45.67) decrease

in performance for this new setting, even though it was the best performing model for the previous setting. These results hint that SciBERT is overfitting to the names of the variables instead of abstracting from its symbols and adapting from context. A significant decrease can also be seen for MathBERT. The best performing baseline for this setting is BERT, which is likely, as previously mentioned, using syntactical cues to obtain the correct types.

We can see that VarSlot with $T = 2$ is more robust to this transition, having a less prominent decrease when compared to the other results (81.44→71.86), while we see a slight larger decrease for $T = 3$. The results suggest that our model better understands the difference in behaviour between variables and words and the surface agnostic aspect of variables, not over-fitting as much as the baselines for names of the variables. The results also hints that our model can correctly obtain the meaning of the variables from the context.

8.4.4 Robustness to substitution

In a more practical scenario, a model should harmonise a combination of the classic and renaming settings. During inference time, a mixture of seen and unseen variables is likely to be present. In this section, we evaluate the robustness of the model for the duplication of hypotheses with substituted variable names.

We add n repetitions of the same hypothesis in the test set for this experiment but with different variable names. For $n = 0$, the Test set is the same as the classic setting. For $n > 0$, we follow the same procedure as the renaming setting; however, we change the letter being used when adding more repetitions. For instance, for $n = 1$ the variables have format $x_{[\text{variable\_index}]}$, while for $n = 2$ we use the format $y_{[\text{variable\_ index}]}$. For $n = 10$, we will have the same hypothesis appear ten times, but all with different variable names.

This test allows us to compare the robustness of each model to the addition of different variables names. Figure 8.3 presents the performance of the different models in this task, considering the F1 score and number of added substitutions n. For $n = 0$ the results are the same as the ones present in Table 8.1.

We can observe that all the models suffer degradation in the performance as we add more duplicated modified hypotheses. While the models start at a close point, we can notice that as we increase the value of n, the gap between the proposed method and the others models increases. Again, we note a big decrease in SciBERT's performance. The performance obtained for VarSlot shows how using the slot attention for

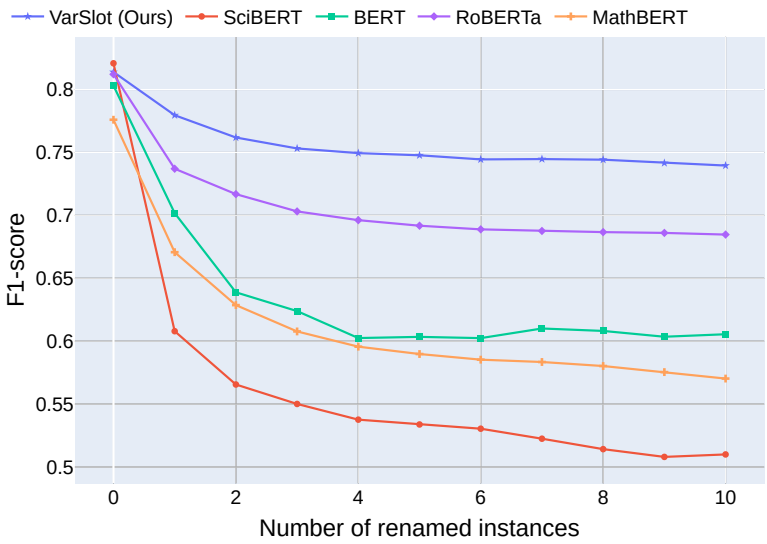


Figure 8.3: F1 score obtained for different models when adding the same hypotheses with different variable name substitutions.

representation of variables can increase the robustness of the model for variable name substitution through the test set.

8.4.5 Ablation Studies

In order to establish what components are impacting on VarSlot comprehension of variables, we perform different ablation studies. Table 8.2 presents the F1 score for different tests. The first row shows the results for considering only the encoder of our method (SciBERT-NLI), removing the enriched variable representations and fine-tuning for the classification task. We can notice that our method improves on top of the sentence representation, with significantly better results on the Renaming setting, showing that the Slot Attention plays a crucial role on that task.

Ablation	**Classic**	**Renaming**
1. Encoder only (SciBERT-NLI)	80.92	58.88
2. Encoder only (BERT-NLI)	80.67	66.63
3. Using BERT-NLI as encoder	81.24	69.19
4. VarSlot ($T = 1$)	79.52	68.03
5. VarSlot ($T = 2$)	81.44	**71.86**
6. VarSlot ($T = 3$)	**82.26**	66.58
7. VarSlot ($T = 4$)	80.42	65.88

Table 8.2: Comparison of our method for the different settings.

We also tested the generalisability of our method to different encoders. Row 2 presents the results for the hypothesis classification only using BERT-NLI[2], and Row 3 presents our method combined with BERT-NLI. We found that by using BERT-NLI instead of SciBERT-NLI for encoding our hypothesis, we could still observe an increase in performance for both classic and renaming settings. The proposed model consistently increases the understanding of variables for different sentence representation models.

The number of iterations T can also have an impact on the generalisation abilities of VarSlot. Rows 4-7 presents how its performance changes as the number of iterations increases. We can notice that we can obtain best results for the renaming setting with $T = 2$, while the best performance for the classic setting comes for $T = 3$. Similarly to previous work on Slot Attention [53], our model requires a few iterations to conform the variable representation into the correct shape. When adding more iterations, VarSlot is more likely to overfit, leading to a degradation in performance with increasing number of iterations.

8.4.6 Qualitative Analysis

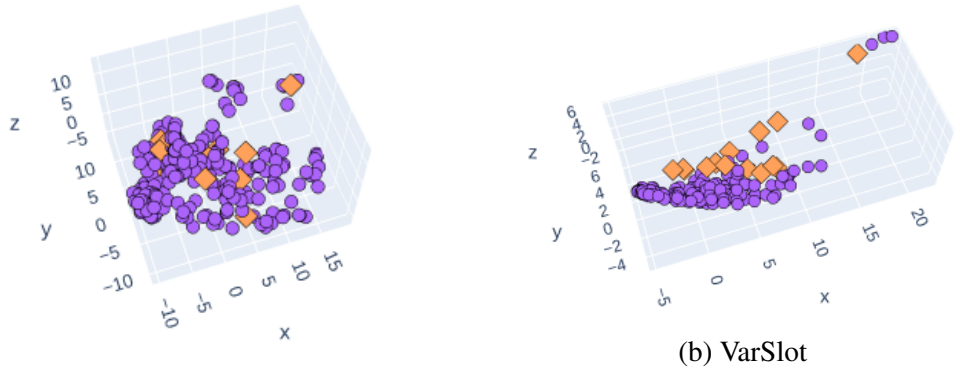
Our experiments from Chapter 6 have found that having separate embedding spaces for mathematical elements and natural language when representing mathematical text can be beneficial for performance in other mathematical language processing tasks. We initially hypothesised that by allowing our model to learn the representation of variables, such separation would naturally happen when starting from a Gaussian initialised slot.

In order to verify if the model can discriminate between variables and natural language, we compare the embedding of words obtained from SciBERT-NLI (fine-tuned for the Variable Typing task) and our method. We selected ten random sentences from the test set, obtained their embedding using both and reduced them to three dimensions using Principal component analysis[3]. The results are presented in Figure 8.4.

The embeddings for the variables are represented here using the yellow diamond. From the figure, we can immediately observe that our model easily distinguishes between words and variables, creating a clear separation. The same is not observed for SciBERT-NLI: there is no sharp separation between words and variables, suggesting that the model might not recognise the distinct semantic behaviour.

[2]Model obtained from Hugging Face: `bert-base-nli-mean-tokens`

[3]We use the PCA implementation from `Scikit-learn` library with the default parameters.



(a) SciBERT-NLI

(b) VarSlot

Figure 8.4: Embedding of the words and variables obtained from the test set for SciBERT-NLI and VarSlot.

We also designed a probing test to quantify which model generates a more separable representation for variables. We obtained the representations for every token present in the test set, generating a representation for each word and variable using VarSlot and SciBERT-NLI. Then, given each representation, we trained a Linear Model to classify these representations into variables or non-variables. For the Linear Model, we use the standard configurations from the Probe-Ably[4] [28] probing framework. The results can be observed in Figure 8.5. In terms of accuracy, we can notice that the representation generated by our model allows for an easier disentanglement between variables and words, with a Linear model achieving maximum accuracy for different norm values.

The selectivity score, namely the difference in accuracy between the representational probe and a control probing task with randomised labels, serves as an indicator that the probe architectures used are not expressive enough to "memorise" unstructured labels, as discussed in Hewitt and Liang [37]. Ensuring that there is no drop-off in selectivity increases the confidence that we are not falsely attributing strong accuracy scores to the representational structure where they could have been explained by over-parametrised probes.

Such finding reinforces the idea that variables requires a specialised representation. The slots for variables could have potentially relearned to replicate the original representations obtained from the encoder, however, a natural separation has been obtained between variables and words without explicitly being trained for that. This hints to the

[4]The parameters used for the Linear Model and splits are found in the Probe-Ably repository `https://github.com/ai-systems/Probe-Ably`

fact that such separation is beneficial for a better performance on tasks involving mathematical knowledge and models dealing with mathematical knowledge could benefit from explicitly defining such property.

8.5 Discussion

In this chapter, we introduced VarSlot, a model for solving the task of variable typing generating a more generalisable representation of variables. Answering **RQ3** (*How to encode the semantics of variables at sentence-level representations?*), our model leverages a type of attention mechanism called Slot Attention that generates empty slots that can be re-encoded according to its input. In order to evaluate the proposed encoding, with a particular emphasis on variable semantics, we propose a new setting for the variable typing task, where during inference type, the model is only exposed to new variable names.

We observed that in this setting, there is a decrease in performance for all tested models; however, VarSlot was the most robust model. As future work, we leave the application of these specialised embeddings for different downstream tasks. We expected that generating better representations for variables would enable an increase in performance across different mathematical language inference tasks.

While the results here show that VarSlot is a robust methodology, there are some clear limitations in this approach:

- We assume that variables are declared in the same sentence.

  Our methodology assumes that most of the variables are declared inside a single sentence (i.e., the variable is declared and used in the same sentence). A statistical analysis of mathematical corpora has shown that most of the variables are declared inside the same sentence, where it is first mentioned [32]. The dataset used for this chapter is also based on the same assumption. At the same time, we know that our methodology will fail for the outlier cases.

- We only present the results for the Variable typing task.

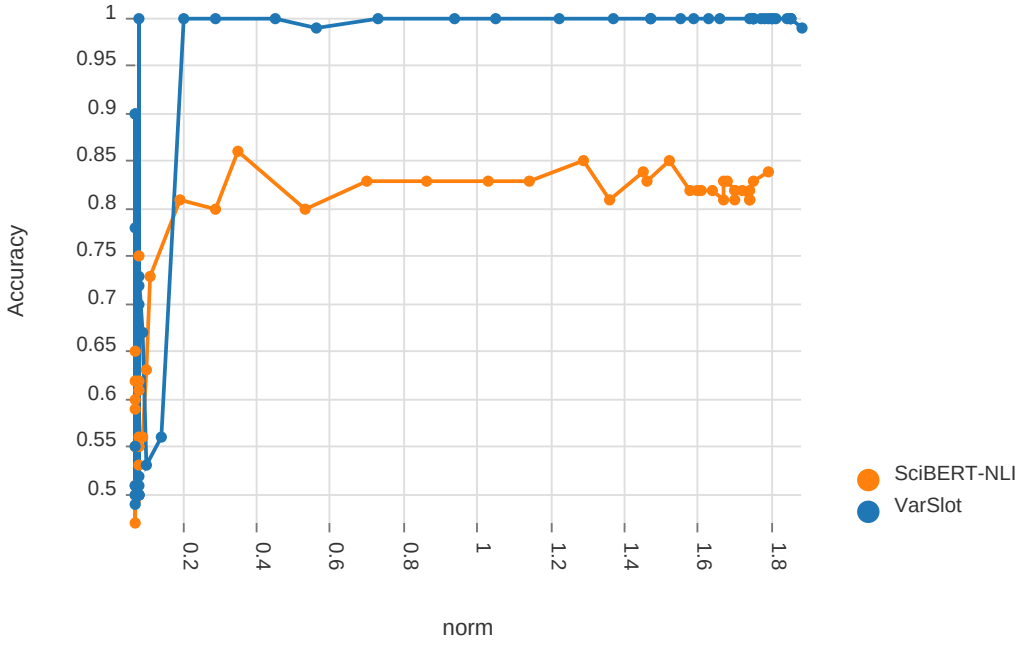
  While we expect this methodology to be applicable to other mathematical inference tasks, we scope this chapter to only the variable typing task. As future work, we leave the experiments of this approach for other MathLP tasks, such as the NLPS.

8.6 Reproducibility Statement

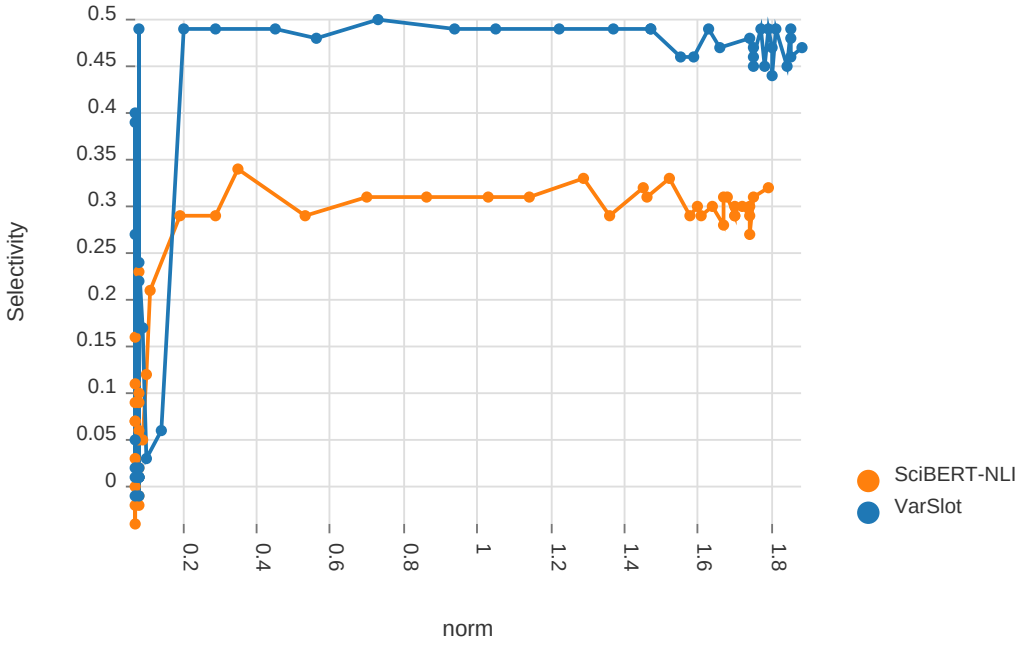
The models and datasets used are made available in our repository `https://github.com/debymf/varslot`. The probing framework used was Probe-Ably (`https://github.com/ai-systems/Probe-Ably`). The list of used hyper-parameters, which are shared across baselines and our method are as follows:

- Seed: 42
- Batch size: 32
- Train Epochs: 10
- Learning Rate: 1e-5 (5e-5 for MathBERT)
- Gradient Accumulation Steps: 1
- Weight Decay: 0.0
- Adam Epsilon: 1e-8
- Warmup Steps: 0
- Max Grad Norm: 1.0

For fine-tuning all the models, we used 4 Tesla 16GB V100 GPUs.



(a) Probing accuracy



(b) Probing Selectivity

Figure 8.5: Results for the probing task, where we compare the representations generated from SciBERT-NLI (yellow) and VarSlot (blue).

Chapter 9

Conclusion and Future Work

9.1 Summary and Conclusions

In this chapter we revisit the research questions stated in Chapter 1 and answered across the thesis.

RQ1: *How to support the evaluation of models for the interpretation and inference over mathematical text (regarding inference tasks, evaluation measures and supporting datasets)?*

In Chapter 4 we answer this question by introducing a novel dataset, PS-ProofWiki, and a new MathLP task, Natural Language Premise Selection. This dataset was proposed to address a research gap existing in the field of MathLP: the need for datasets which express complex interpretation and inference phenomena, allowing the evaluation of encoding methods at a sentence-level (encoding mathematical elements together with natural language) and at argument-level (encoding the relationship between mathematical statements). This dataset also provides different scenarios for evaluating the NLPS task: a multi-hop premise selection task across different mathematical domains. The dataset is built over a wiki-based platform, reflecting the stylistic diversity of different authors, it reflects the balance between the two linguistic modalities and targets well-established mathematical results.

Moreover, we built a set of baselines based on canonical IR weighting schemes and pre-trained transformer-based language models. We find that despite the fact that pre-trained LMs are not designed for mathematical text, they are still able to achieve better performance than the IR-based baselines, indicating that there is potential for using these types of models for supporting the interpretation of mathematical text.

Scoping & Limitations: The developed dataset focused on a relatively lower complexity subset of mathematical discourse, where the corpora reflects textbook-level mathematical proofs (within a broad range of the academic spectrum). Other types of mathematical text were not covered in the proposed evaluation, for example, scientific papers in Mathematics or to more applied domains such as Physics. While the composition of mathematical expressions, sentences and argumentation structures are expected to be representative of other texts, this was not empirically validated in this work.

Statement of contributions:

1. Definition of a new MathLP task: Natural Language Premise Selection (NLPS).
2. Construction of a new supporting dataset: PS-ProofWiki.
3. Developed a set of IR-based and transformer-based baselines.
4. Resources: Dataset, dataset construction workflow, evaluation scripts are available as open source resources.

RQ2: *Can we improve the performance of deep learning models for inference over mathematical text by generating specialised encodings for mathematical natural language elements?*

In Chapter 6 we proposed a new model for sentence-level representation of mathematical text which maps natural language and mathematical expressions into different embedding spaces. This is performed via a multi-modality self-attention mechanism, which attends distinctively over natural language and mathematical elements and then harmonises both inputs via an LSTM layer. The performance increases consistently across statements with different distributions of natural language and mathematical language elements. This chapter also shows that this method outperforms large language models. This suggests that deep learning models for MathLP can benefit from first learning how to separately represent each token modality and then learning how to merge them into a single encoding. The task was empirically validated under the NLPS setting.

Scoping & Limitations: The empirical analysis concentrated on the use of an extrinsic evaluation setting (the NLPS task) to assess the model. Sensible intrinsic evaluation methods and measures which would be applicable in a language model setting were not covered in the scope of this work. Moreover, the work focuses on a quantitative

measurement argument, using the canonical evaluation setting which is current practice in the field, not covering a more detailed qualitative characterisation of the behaviour of the model.

Statement of contributions:

1. Definition of a new conceptual contribution: the separate encoding of the two linguistic modalities in mathematical text.
2. Construction of a new supporting architecture: specialisation of self-attention mechanisms to support a sentence-level encoding.
3. Model construction and extrinsic empirical evaluation under a NLPS setting.
4. Resources: Source code for the model and evaluation scripts available as open source resources.

RQ3: *How to encode the semantics of variables at sentence-level representations?*

This question was addressed in a pre-trained language model scenario, where the original representation was enriched representation model specialised on the encoding of variables. In order to assess the robustness of the generated representations, we designed a novel testing method based on systematic variable renaming. We found that by using attention slots to represent variables, where the slots learn to represent the variables based on their context, we could improve the performance for the variable typing task and also increase the robustness to variable renaming. We were able to generate representations less bound to surface lexical variability and better connected to its context.

Scoping & Limitations: The empirical analysis focused solely on the Variable Typing task. While the VarSlot model can be easily extended (i.e., the variables slots can be applied regardless of the task) for application across other tasks, the exploration of the behaviour of this architectural element for other tasks has been left as future work. Another important assumption, that is inherited from previous work is that we perform inference over a single statement, assuming that the variables are declared and used inside the same mathematical statement.

Statement of contributions:

1. Definition of a new conceptual contribution: Specialised sentence-level embeddings for the semantic behaviour of variables.

2. Construction of a new supporting architecture: specialisation of slot attention mechanisms to support sentence-level variable representation.

3. Proposal of a new evaluation task: Systematic variable renaming for the evaluation of variable representation robustness.

4. Model construction and extrinsic empirical evaluation under a Variable Typing setting.

5. Resources: Source code for the model and evaluation scripts available as open source resources.

RQ4: *Can the dependencies between mathematical statements provide a simplified representation of mathematical arguments?*

We establish empirically that models based on lexical and semantic overlaps (e.g. BM25 and Sentence-Transformers) for sentence-level premise selection achieve low performance, especially on mean average precision, when taking into account statement-level representations at statement level, even after fine-tuning large models. We found that the use of argumentation-level structural similarity can provide substantial improvements for premise selection, by encoding the general structure of mathematical arguments. This improvement is observable for both IR-based and transformer-based models. The underlying modelling assumption is that the problem of premise selection requires the balancing of both localised evidence (which is confined to the evidence provided at the statement-level) to broader, global dependencies (i.e., statements that have several premises in common). This adapts the notion of an unification score into the MathLP setting.

Scoping & Limitations:

The performance of the proposed Conjecture-Premise retrieval score relies on the quality of an external pre-trained embedding model, given that the proposed method is not trainable to generate encodings for statements. When comparing different pre-trained Sentence-Transformers, we show how the results of similar models (e.g., models based on MPNet) can have distinct scores. In order to apply the same technique to other datasets, there should be, first, an analysis of the suitability of different encoders, given that the same encoder that works for premise selection, might not be suitable for another similar task.

Statement of contributions:

1. Definition of a new conceptual contribution: Specialised structural-level embeddings for NLPS.

2. Construction of a new supporting architecture: design of a retrieval-based model which integrates statement-level and structural-level measures.

3. Model construction and extrinsic empirical evaluation under an NLPS retrieval setting.

4. Resources: Source code for the model and evaluation scripts available as open source resources.

RQ5: *How to encode the inter-statement dependency structure to support inference for the natural language premise selection task?*

This work represented the graph structure entailed the premise co-reference structure in the corpus to support inference in the NLPS setting. A graph neural network-based model with particular emphasis on encoding the network-level features (DGCNN) was used as a support for the generation of the embeddings. The underlying conceptual assumption is that the generated embeddings will capture the a latent measure of unification (which can be translated as a network centrality concept). The proposed model demonstrated to deliver better premise selection and robustness over a multi-hop NLPS setting.

Scoping & Limitations: Overall, graph-based neural architectures are known not to scale well [6]. Our proposed DGCNN model requires encoding large parts of the graph simultaneously in memory. We operate under scalability constraints, sampling for a limited set of negative premises to avoid an increase in memory and processing complexity.

Statement of contributions:

1. Definition of a new conceptual contribution: Specialised graph-based embeddings capturing the mathematical argument structure in the mathematical text based on the notion of unification.

2. Construction of a new supporting architecture: design of a Deep Convolution Graph Neural Network for predicting the link between mathematical statements.

3. Comparison with baselines across different settings: comparison in a subset of NLPS and a multi-hop scenario.

4. Delivery of a model which is more robust towards n-hop settings.

5. Model construction and extrinsic empirical evaluation under an NLPS inference setting.

6. Resources: Source code for the model and evaluation scripts available as open source resources.

The investigations of the above research questions lead to a conclusion of the high-level research question proposed in this work:

RQ: *Can Deep Learning-based representation models support understanding and inference over mathematical text corpora?*

We have evaluated deep learning-based representations for two tasks: the natural language premise selection (for both inference and retrieval) and the variable typing task. Encoding mathematical statements using deep learning models has consistently improved performance compared to previously proposed methods and IR baselines. While the results confirm that these models can support understanding and inference over mathematical discourse, this work has also presented that the performance of such models can be improved by encoding particular aspects of mathematical discourse, such as the structural similarity between statements and the context-dependent meaning of variables.

9.2 Opportunities for Further Research

During the course of this work, opportunities for new research programmes on MathLP were identified. These opportunities dialogue with both limitations of this work and broader research trends:

- *Complex, end-to-end architectures for MathLP*

  This work introduced different representational mechanisms under the Deep Learning architectural framework to support the interpretation and inference of mathematical text. This work fell short on providing an integration of these representational devices, evaluating the integrated ensemble of the proposed contributions and assessing the cross-interaction between these components.

- *Encoding of complex mathematical objects*

While the representation of variables was explored, there is a need for a better understanding of the representational requirements of more complex mathematical objects (e.g. functions and operators) and complex formulae. Mathematical text communicates with the methodological benefit of expressing statements over objects with well-defined semantic properties, allowing for the opportunity of systematically exploring long-distance semantic relations.

- *From language models to abstract use of language*

  Mathematical text provides a better contained universe of discourse to explore the dialogue between the opportunistic space provided by language models (with an emphasis on paradigmatic and syntagmatic relations), to the need for encodings which require better abstraction capabilities. This work took the contemporary language model perspective to start a movement in the direction of encoding and evaluating more abstract uses of language.

- *Transferability across tasks and subdomains*

  The application and suitability analysis of the methods presented here for other MathLP tasks such as Math Summarization and Mathematical Information Retrieval. Given that the techniques presented here provided consistent improvements for Variable Typing and NLPS, we expect that they are also generalisable for other tasks, given the uniform vocabulary, sentence and argumentation structure of mathematical text.

- *Proof representation and discourse relations*

  While PS-ProofWiki is extracted from the structure of proofs, we do not use the content of those proofs, only the external knowledge required by them. It is not part of the scope of this work to explore the internal structure of the proofs. We leave it as future work to further explore how the arguments are structured inside a proof and how the methods introduced in this work can be improved and leveraged for challenging tasks such as automated natural language proofs.

Glossary

ATP Automated Theorem Proving

BOW Bag-of-Words

MathLP Mathematical Language Processing

MDC Mathematical Document Categorisation

MLP Multilayer Perceptron

NLP Natural Language Processing

NLPS Natural Language Premise Selection

Bibliography

[1] Akiko Aizawa and Michael Kohlhase. “Mathematical Information Retrieval”. In: *Evaluating Information Retrieval and Access Tasks*. Springer, Singapore, 2021, pp. 169–185.

[2] Akiko Aizawa et al. “NTCIR-11 Math-2 Task Overview.” In: *NTCIR*. Vol. 11. 2014, pp. 88–98.

[3] Jesse Alama et al. “Premise Selection for Mathematics by Corpus Analysis and Kernel Methods”. In: *Journal of Automated Reasoning* 52.2 (Feb. 2014), pp. 191–213. ISSN: 1573-0670. DOI: 10.1007/s10817-013-9286-5. URL: https://doi.org/10.1007/s10817-013-9286-5.

[4] Maria Alexeeva et al. “Mathalign: Linking formula identifiers to their contextual natural language descriptions”. In: *Proceedings of The 12th Language Resources and Evaluation Conference*. 2020, pp. 2204–2212.

[5] Daniel Alibert and Michael Thomas. “Research on mathematical proof”. In: *Advanced mathematical thinking*. Springer, 2002, pp. 215–230.

[6] Uri Alon and Eran Yahav. “On the bottleneck of graph neural networks and its practical implications”. In: *arXiv preprint arXiv:2006.05205* (2020).

[7] Marie Amalric and Stanislas Dehaene. “Origins of the brain networks for advanced mathematics in expert mathematicians”. In: *Proceedings of the National Academy of Sciences* 113.18 (2016), pp. 4909–4917.

[8] Robert L Baber. *The language of mathematics: utilizing math in practice*. John Wiley & Sons, 2011.

[9] Grzegorz Bancerek et al. “The Role of the Mizar Mathematical Library for Interactive Proof Development in Mizar”. In: *J. Autom. Reason.* 61.1-4 (2018), pp. 9–32. DOI: 10.1007/s10817-017-9440-6. URL: https://doi.org/10.1007/s10817-017-9440-6.

[10] Barbara Beeton and Richard Palais. “Communication of Mathematics with TEX.” In: *Visible Language* 50.2 (2016).

[11] Iz Beltagy, Kyle Lo, and Arman Cohan. “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)*. 2019, pp. 3615–3620.

[12] Julius Berner, Markus Dablander, and Philipp Grohs. “Numerically solving parametric families of high-dimensional Kolmogorov partial differential equations via deep learning”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 16615–16627.

[13] Jasmin Christian Blanchette et al. “Hammering towards QED”. In: *Journal of Formalized Reasoning* 9.1 (2016), pp. 101–148.

[14] Nicholas Bourbaki. “The architecture of mathematics”. In: *The American Mathematical Monthly* 57.4 (1950), pp. 221–232.

[15] Brian Butterworth. “Mathematics and the Brain”. In: *Opening address to the Mathematical Association, Reading* (2002).

[16] Ovidiu Calin. “Abstract Neurons”. In: *Deep Learning Architectures: A Mathematical Approach*. Cham: Springer International Publishing, 2020, pp. 133–165. ISBN: 978-3-030-36721-3. DOI: 10.1007/978-3-030-36721-3_5. URL: https://doi.org/10.1007/978-3-030-36721-3_5.

[17] Kyunghyun Cho et al. “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)*. 2014, pp. 1724–1734.

[18] Maximin Coavoux and Shay B Cohen. “Learning to Match Mathematical Statements with Proofs”. In: *arXiv preprint arXiv:2102.02110* (2021).

[19] Kevin Bretonnel Cohen and Dina Demner-Fushman. *Biomedical natural language processing*. Vol. 11. John Benjamins Publishing Company, 2014.

[20] Antonella Cupillari. *The Nuts and Bolts of Proofs: An Introduction to Mathematical Proofs*. Academic Press, 2005.

[21] Pankaj Dadure, Partha Pakray, and Sivaji Bandyopadhyay. "Embedding and generalization of formula with context in the retrieval of mathematical information". In: *Journal of King Saud University-Computer and Information Sciences* (2021).

[22] Pankaj Dadure, Partha Pakray, and Sivaji Bandyopadhyay. "Mathematical Information Retrieval Trends and Techniques". In: *Deep Natural Language Processing and AI Applications for Industry 5.0*. IGI Global, 2021, pp. 74–92.

[23] Li Deng and Yang Liu. *Deep learning in natural language processing*. Springer, 2018.

[24] Jacob Devlin et al. "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)*. 2019, pp. 4171–4186.

[25] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *NAACL-HLT (1)*. 2019.

[26] Sergey Edunov et al. "Understanding Back-Translation at Scale". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 489–500.

[27] Michael Färber and Cezary Kaliszyk. "Random Forests for Premise Selection". In: *Frontiers of Combining Systems*. Ed. by Carsten Lutz and Silvio Ranise. Cham: Springer International Publishing, 2015, pp. 325–340. ISBN: 978-3-319-24246-0.

[28] Deborah Ferreira et al. "Does My Representation Capture X? Probe-Ably". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Aug. 2021, pp. 194–201. DOI: 10.18653/v1/2021.acl-demo.23. URL: https://aclanthology.org/2021.acl-demo.23.

[29] Thibault Gauthier and Cezary Kaliszyk. "Premise Selection and External Provers for HOL4". In: *Proceedings of the 2015 Conference on Certified Programs and*

Proofs. CPP '15. Mumbai, India: ACM, 2015, pp. 49–57. ISBN: 978-1-4503-3296-5. DOI: 10.1145/2676724.2693173. URL: http://doi.acm.org/10.1145/2676724.2693173.

[30] Alex Graves. "Long short-term memory". In: *Supervised sequence labelling with recurrent neural networks*. Springer, 2012, pp. 37–45.

[31] André Greiner-Petter et al. "Why Machines Cannot Learn Mathematics, Yet". In: *4th BIRNDL workshop at 42nd SIGIR* (2019). URL: http://arxiv.org/abs/1905.08359.

[32] Mihai Grigore, Magdalena Wolska, and Michael Kohlhase. "Towards context-based disambiguation of mathematical expressions". In: *The joint conference of ASCM*. 2009, pp. 262–271.

[33] Kevin Gurney. *An introduction to neural networks*. CRC press, 2018.

[34] Matthew Henderson et al. "Efficient natural language response suggestion for smart reply". In: *arXiv preprint arXiv:1705.00652* (2017).

[35] Dan Hendrycks et al. "Measuring mathematical problem solving with the math dataset". In: *arXiv preprint arXiv:2103.03874* (2021).

[36] Tom Henighan et al. "Scaling laws for autoregressive generative modeling". In: *arXiv preprint arXiv:2010.14701* (2020).

[37] John Hewitt and Percy Liang. "Designing and Interpreting Probes with Control Tasks". 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 2733–2743. DOI: 10.18653/v1/D19-1275. URL: https://aclanthology.org/D19-1275.

[38] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[39] Zhenya Huang et al. "Neural mathematical solver with enhanced formula structure". In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2020, pp. 1729–1732.

[40] Bat-Sheva Ilany, Bruria Margolin, et al. "Language and mathematics: Bridging between natural language and mathematical language in solving problems in mathematics". In: *Creative Education* 1.03 (2010), p. 138.

[41] Geoffrey Irving et al. “DeepMath-deep sequence models for premise selection”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 2235–2243.

[42] Nanjiang Jiang and Marie-Catherine de Marneffe. “Evaluating BERT for natural language inference: A case study on the CommitmentBank”. 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)*. 2019, pp. 6088–6093.

[43] D Jurafsky and JE Martin. *Speech & Language Processing, an Introduction to NL Processing, Computational Linguistics & Speech Recognition*. 2000.

[44] Samuel Kim et al. “Integration of neural network-based symbolic regression in deep learning for scientific discovery”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.9 (2020), pp. 4166–4177.

[45] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. “Siamese neural networks for one-shot image recognition”. In: *ICML deep learning workshop*. Vol. 2. Lille. 2015, p. 0.

[46] Giovanni Yoko Kristianto, Goran Topić, and Akiko Aizawa. “Utilizing dependency relationships between math expressions in math IR”. In: *Information Retrieval Journal* 20.2 (2017), pp. 132–167.

[47] Zhenzhong Lan et al. “ALBERT: A Lite BERT for Self-supervised Learning of Language Representations”. In: *International Conference on Learning Representations*. 2019.

[48] Quoc Le and Tomas Mikolov. “Distributed representations of sentences and documents”. In: *International conference on machine learning*. 2014, pp. 1188–1196.

[49] Kevin P Lee. “A guide to writing mathematics”. In: (2010).

[50] Jason Lin et al. “Prediction of mathematical expression declarations based on spatial, semantic, and syntactic analysis”. In: *Proceedings of the ACM Symposium on Document Engineering 2019*. 2019, pp. 1–10.

[51] Yinhan Liu et al. “Roberta: A robustly optimized bert pretraining approach”. In: *arXiv preprint arXiv:1907.11692* (2019).

[52] Zhiyuan Liu, Yankai Lin, and Maosong Sun. "Representation Learning and NLP". In: *Representation Learning for Natural Language Processing*. Singapore: Springer Singapore, 2020, pp. 1–11. ISBN: 978-981-15-5573-2. DOI: 10.1007/978-981-15-5573-2_1. URL: https://doi.org/10.1007/978-981-15-5573-2_1.

[53] Francesco Locatello et al. "Object-Centric Learning with Slot Attention". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 11525–11538. URL: https://proceedings.neurips.cc/paper/2020/file/8511df98c02ab60aea1b2356c013bc0f-Paper.pdf.

[54] Norman Meuschke et al. "Analyzing mathematical content to detect academic plagiarism". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017, pp. 2211–2214.

[55] Norman Meuschke et al. "Improving academic plagiarism detection for STEM documents by analyzing mathematical content and citations". In: *2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. IEEE. 2019, pp. 120–129.

[56] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[57] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. "Learning Convolutional Neural Networks for Graphs". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 2014–2023. URL: http://dl.acm.org/citation.cfm?id=3045390.3045603.

[58] Kay O'Halloran. *Mathematical discourse: Language, symbolism and visual images*. A&C Black, 2008.

[59] Christopher Olah. "Understanding lstm networks". In: (2015).

[60] Shuai Peng et al. "MathBERT: A Pre-Trained Model for Mathematical Formula Understanding". In: *arXiv preprint arXiv:2105.00377* (2021).

[61] Randolph A Philipp. "The Many Uses of Algebraic Variables". In: *The Mathematics Teacher* 85.7 (1992), pp. 557–561.

[62] Piotr Piekos, Henryk Michalewski, and Mateusz Malinowski. "Measuring and Improving BERT's Mathematical Abilities by Predicting the Order of Reasoning". In: *arXiv preprint arXiv:2106.03921* (2021).

[63] Valentina Postelnicu and Florin Postelnicu. "College students' understanding of parameters in algebra". In: *CERME 9-Ninth Congress of the European Society for Research in Mathematics Education*. 2015, pp. 453–459.

[64] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational physics* 378 (2019), pp. 686–707.

[65] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 3982–3992.

[66] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

[67] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. "A primer in bertology: What we know about how bert works". In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 842–866.

[68] Maria Ryskina and Kevin Knight. "Learning Mathematical Properties of Integers". In: *arXiv preprint arXiv:2109.07230* (2021).

[69] Walter Warwick Sawyer. *Prelude to mathematics*. Courier Corporation, 1982.

[70] David Saxton et al. "Analysing Mathematical Reasoning Abilities of Neural Models". In: *International Conference on Learning Representations*. 2018.

[71] Philipp Scharpf et al. "Classification and Clustering of arXiv Documents, Sections, and Abstracts, Comparing Encodings of Natural and Mathematical Language". In: *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*. 2020, pp. 137–146.

[72] Imanol Schlag et al. "Enhancing the transformer with explicit relational encoding for math problem solving". In: *arXiv preprint arXiv:1910.06611* (2019).

[73] Alan H Schoenfeld and Abraham Arcavi. "On the meaning of variable". In: *The mathematics teacher* 81.6 (1988), pp. 420–427.

[74] Moritz Schubotz et al. “Challenges of mathematical information retrievalin the ntcir-11 math wikipedia task”. In: *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 2015, pp. 951–954.

[75] Moritz Schubotz et al. “Evaluating and improving the extraction of mathematical identifier definitions”. In: *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer. 2017, pp. 82–94.

[76] Mike Schuster and Kuldip K Paliwal. “Bidirectional recurrent neural networks”. In: *IEEE transactions on Signal Processing* 45.11 (1997), pp. 2673–2681.

[77] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*. Vol. 39. Cambridge University Press Cambridge, 2008.

[78] Kaitao Song et al. “Mpnet: Masked and permuted pre-training for language understanding”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 16857–16867.

[79] Georgios Spithourakis and Sebastian Riedel. “Numeracy for Language Models: Evaluating and Improving their Ability to Predict Numbers”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 2104–2115.

[80] Yiannos A Stathopoulos et al. “Variable Typing: Assigning Meaning to Variables in Mathematical Text”. In: *Proceedings of NAACL-HLT*. 2018, pp. 303–312.

[81] Pontus Stenetorp et al. “BioNLP shared task 2011: Supporting resources”. In: *Proceedings of BioNLP shared task 2011 workshop*. 2011, pp. 112–120.

[82] Tokinori Suzuki and Atsushi Fujii. “Mathematical document categorization with structure of mathematical expressions”. In: *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. IEEE. 2017, pp. 1–10.

[83] Avijit Thawani et al. “Representing Numbers in NLP: a Survey and a Vision”. In: *arXiv preprint arXiv:2103.13136* (2021).

[84] Marco Valentino, Mokanarangan Thayaparan, and André Freitas. “Unification-based Reconstruction of Multi-hop Explanations for Science Questions”. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 2021, pp. 200–211.

[85] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[86] Shlomo Vinner. “What Is Mathematics?” In: *Mathematics, Education, and Other Endangered Species: From Intuition to Inhibition*. Cham: Springer International Publishing, 2018, pp. 63–68. ISBN: 978-3-319-90035-3. DOI: 10.1007/978-3-319-90035-3_8. URL: https://doi.org/10.1007/978-3-319-90035-3_8.

[87] Eric Wallace et al. “Do NLP Models Know Numbers? Probing Numeracy in Embeddings”. 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)*. 2019, pp. 5307–5315.

[88] Alex Wang et al. “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding”. In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. 2018, pp. 353–355.

[89] Qingxiang Wang, Cezary Kaliszyk, and Josef Urban. “First experiments with neural translation of informal to formal mathematics”. In: *International Conference on Intelligent Computer Mathematics*. Springer. 2018, pp. 255–270.

[90] Xinyu Wang et al. “Automated Concatenation of Embeddings for Structured Prediction”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2021, pp. 2643–2660.

[91] Yan Wang, Xiaojiang Liu, and Shuming Shi. “Deep neural solver for math word problems”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017, pp. 845–854.

[92] Sebastian Wankerl, Gerhard Götz, and Andreas Hotho. “f2tag—Can Tags be Predicted Using Formulas?” In: *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2020, pp. 565–571.

[93] Seth Warner. *Modern algebra*. Courier Corporation, 1990.

[94] Sean Welleck et al. “NaturalProofs: Mathematical Theorem Proving in Natural Language”. In: *arXiv preprint arXiv:2104.01112* (2021).

[95] Yonghui Wu et al. “Google’s neural machine translation system: Bridging the gap between human and machine translation”. In: *arXiv preprint arXiv:1609.08144* (2016).

[96] Zhilin Yang et al. “HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 2369–2380.

[97] Michihiro Yasunaga and John D Lafferty. “Topiceq: A joint topic and mathematical equation model for scientific texts”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 7394–7401.

[98] Ke Yuan et al. “Automatic Description Construction for Math Expression via Topic Relation Graph”. In: *arXiv preprint arXiv:2104.11890* (2021).

[99] Ke Yuan et al. “Automatic generation of headlines for online math questions”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. 2020, pp. 9490–9497.

[100] Richard Zanibbi et al. “NTCIR-12 MathIR Task Overview.” In: *NTCIR*. 2016.

[101] Dongxiang Zhang et al. “The gap of semantic parsing: A survey on automatic math word problem solvers”. In: *IEEE transactions on pattern analysis and machine intelligence* 42.9 (2019), pp. 2287–2305.

[102] Muhan Zhang and Yixin Chen. “Link prediction based on graph neural networks”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 5165–5175.

[103] Muhan Zhang et al. “An end-to-end deep learning architecture for graph classification”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[104] Claus Zinn. “A Computational Framework For Understanding Mathematical Discoursexy”. In: *Logic Journal of IGPL* 11.4 (2003), pp. 457–484.

Appendix A

LaTeX vs MathML

Consider the following representation for the quadratic formula $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$ in LaTeX:

```
x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}
```

and the equivalent in MathML:

```
<math>
  <mi>x</mi>
  <mo>=</mo>
  <mfrac>
    <mrow>
      <mo>&#x2212;<!-- - --></mo>
      <mi>b</mi>
      <mo>&#x00B1;<!-- ± --></mo>
      <msqrt>
        <msup>
          <mi>b</mi>
          <mn>2</mn>
        </msup>
        <mo>&#x2212;<!-- - --></mo>
        <mn>4</mn>
        <mi>a</mi>
        <mi>c</mi>
      </msqrt>
    </mrow>
    <mrow>
      <mn>2</mn>
      <mi>a</mi>
```

```
    </mrow>
  </mfrac>
</math>
```

The following is the result of the tokenisation using BERT tokeniser for the Latex form of the equation (the commas separates each token):

```
['x', '=','ra','##c','{','-','b','\\','pm','\\',
 'sq','##rt','{','b','^','2','-','4a','##c','}','}','{','2a','}']
```

The same tokeniser is used for the MathML version of the equation, which outputs the following:

```
['<','math','>','<','mi','>','x','<','/','mi','>','<','mo',
'>','=','<','/','mo','>','<','m','##fra','##c','>','<','mr',
'##ow','>','<','mo','>','&','#','x','##22','##12',';','<',
'!','-','-','-','-','-','>','<','/','mo','>','<','mi','>',
'b','<','/','mi','>','<','mo','>','&','#','x','##00',
'##b','##1',';','<','!','-','-','±','-','-','>','<',
'/','mo','>','<','ms','##q','##rt','>','<','ms','##up',
'>','<','mi','>','b','<','/','mi','>','<','mn','>','2',
'<','/','mn','>','<','/','ms','##up','>','<','mo','>',
'&','#','x','##22','##12',';','<','!','-','-','-','-',
'-','>','<','/','mo','>','<','mn','>','4','<','/','mn',
'>','<','mi','>','a','<','/','mi','>','<','mi','>','c',
'<','/','mi','>','<','/','ms','##q','##rt','>','<',
'/','mr','##ow','>','<','mr','##ow','>','<','mn','>'
,'2','<','/','mn','>','<','mi','>','a','<','/','mi'
,'>','<','/','mr','##ow','>','<','/','m','##fra',
'##c','>','1','<','/','math','>']
```

Appendix B

PS-ProofWiki generation

B.1 ProofWiki Source Code Example

Example of the source code of a page in ProofWiki:

```
{{Previous POTW|29 December 2008|19 January 2009}}

== Theorem ==
<onlyinclude>
Let $a, b, c \in \Z$.

Let $a \divides b c$, where $\divides$ denotes
[[Definition:Divisor of Integer|divisibility]].

Let $a \perp b$, where $\perp$ denotes
[[Definition:Coprime Integers|relative primeness]].

Then $a \divides c$.
</onlyinclude>

== [[Euclid's Lemma/Proof 1|Proof 1]] ==
{{:Euclid's Lemma/Proof 1}}

== [[Euclid's Lemma/Proof 2|Proof 2]] ==
{{:Euclid's Lemma/Proof 2}}
{{Namedfor|Euclid|cat = Euclid}}
```

```
== Also see ==

* [[Euclid's Lemma for Prime Divisors]]

[[Category:Euclid's Lemma]]
[[Category:Divisors]]
[[Category:Coprime Integers]]
```

B.2 Example from PS-ProofWiki

Example of an entry from the Training set of PS-ProofWiki:

```
"91ebf385-9f25-49c3-b182-12d8e3411086": {
    "text": "Let $\\RR$ be a relation on a set $S$. Let $
       \\RR^\\ne$, be the reflexive reduction of $\\RR$.
       Then $\\RR^\\ne$ is antireflexive.",
    "premises": [
      "1dc7f036-2316-45c8-b7f6-eca878043854",
      "1e5e391a-4e94-4ee3-8957-416ca0669733",
      "eb9ef16e-1a77-43bf-a8ac-1dfb884c1fba"
    ],
    "category": "Set Theory",
    "type": "theorem"
  },
```

Example of an entry from the Knowledge base of PS-ProofWiki:

```
"398f7103-78dc-448e-96e6-5fba829b5d08": {
    "text": "The '''Zermelo-Fraenkel axioms''' are the
       most well-known basis for axiomatic set theory.
       There is no standard numbering for them, and their
        exact formulation varies. Certain of these axioms
        can in fact be derived from other axioms, so
       their status as "axioms" can be questioned. The
       axioms are as follows:",
    "type": "definitions",
```

```
4     "category": "Set Theory"
5   }
```