Exploration of Neural Machine Translation in Autoformalization of Mathematics in Mizar

Qingxiang Wang
Department of Computer Science
University of Innsbruck
Austria
shawn.wangqingxiang@gmail.com

Chad Brown
CIIRC
Czech Technical University in Prague
Czech Republic

Cezary Kaliszyk
Department of Computer Science
University of Innsbruck and University of Warsaw
Austria
cezary.kaliszyk@uibk.ac.at

Josef Urban
CIIRC
Czech Technical University in Prague
Czech Republic
josef.urban@gmail.com

Abstract

In this paper we share several experiments trying to automatically translate informal mathematics into formal mathematics. In our context informal mathematics refers to human-written mathematical sentences in the LaTeX format; and formal mathematics refers to statements in the Mizar language. We conducted our experiments against three established neural network-based machine translation models that are known to deliver competitive results on translating between natural languages. To train these models we also prepared four informal-to-formal datasets. We compare and analyze our results according to whether the model is supervised or unsupervised. In order to augment the data available for auto-formalization and improve the results, we develop a custom type-elaboration mechanism and integrate it in the supervised translation.

CCS Concepts • **Theory of computation → Logic and verification**; • **Computing methodologies → Neural networks**; • **Applied computing → Language translation**.

Keywords Automating Formalization, Proof Assistants, Neural Machine Translation, Mizar, Machine Learning

1 Introduction

Formalization of all existing mathematics from textbooks and literature has been a long-term dream for researchers in interactive theorem proving (ITP) [2]. For decades we have witnessed the growth of many ITP proof assistant libraries, some of which can now cover quite a proportion of mathematics. However, despite such growth, at the current stage formalizations are still done manually and by a small community of domain experts. This makes the speed of formalization much slower than the speed of increase of mathematics. In order to work toward the vision of formalizing all existing mathematics, we need to figure out ways to automate the formalization process or parts of it. We coin this research effort as *autoformalization*.

Autoformalization is generally considered as a very challenging (or even impossible) task. In addition to all the technical issues facing the ITP community, an autoformalization pipeline needs to capture the human thinking process that enables translation from informal mathematics texts into formal pieces of code in a proof assistant. Therefore, some people argue that implementing such an autoformalization pipeline would amount to implementing a program that is capable of achieving sophisticated natural language understanding and mathematical reasoning. We believe that the above holistic view can be simplified if we first approach the problem by just considering it as a language translation problem, i.e., given an informal mathematical sentence, we translate it into a formal sentence that has the potential to be further formalized into a proof assistant. If such a translation tool were available, we would be able to preprocess the vast mathematical materials and come up with their formal versions that can be used for later steps of formalization.

By simplifying and considering formalization as a language translation problem, we can employ state-of-the-art

machine translation tools and adapt them to our purpose. Currently, the best machine translation models are all based on neural networks [13]. As we are still in the exploration phase, in this paper we will not cover detailed engineering aspects of implementing neural network models but be satisfied with a high-level understanding of them, which we will introduce in Section 3. To simplify our engineering effort we will use three established neural machine translation models:

1) The neural translation model (**NMT**) by Luong et al. [31] is a well documented model developed at Google based on the Tensorflow [1] framework. It is an encoder-decoder architecture with a flexible range of configurable hyper-parameters. The default configuration is a two-layer recurrent neural network with LSTM cells [23]. Wang et al. [49] have conducted extensive experiments with this model in particular with hyper-parameter configurations suitable for autoformalization. In this paper we extend the NMT with a type elaboration mechanism.

2) The unsupervised translation models (**UNMT**) based on Lample et al. [30] and Artetxe et al. [3] attempt to find a mapping between two languages without any initial alignment data. Both models are implemented in the PyTorch [35] framework. As both models share the same idea of transforming an unsupervised learning problem into a series of supervised learning problems, we only perform experiments with the first one, training it with with non-aligned autoformalization data. UNMT can be configured to use both the NMT architecture as in 1) or the transformer architecture [47]. We will illustrate both ideas in Section 3.

3) Lample et al.'s [29] cross-lingual pretraining model (**XLM**) is a multi-task model. We only focus on its translation capability and treat it as an improved version of the UNMT model. In UNMT a technique called word2vec [33] is used to obtain vector representation of word tokens. This vector representation is kept fixed during the whole training and evaluation process. In XLM, a new technique by Devlin at al. [18] called *pretraining* is adapted to directly obtain initial vector representations at both sentence level and word token level. From these initial representations unsupervised learning can be performed to fine-tune the model. Pretraining has been shown by the natural language processing (NLP) community to provide better results in many experimental benchmarks. In this paper we will also see its improvement on our informal-to-formal datasets comparing to the original UNMT model.

The above models require training and testing data in the form of collection of sentences, i.e., a text corpus. For the translation task we need a source corpus and a target corpus (in our case an *informal mathematics* corpus and a *formal mathematics* corpus). Depending on whether the sentences in both corpora are aligned or not, we can employ the supervised model as in 1) or the unsupervised models as in 2) or 3). To further clarify our research objective, in this paper we regard informal mathematics corpus as sentences

in the LaTeX format, and formal mathematics corpus as statements in the Mizar language. We prepared four datasets to test the above models. The first three are LaTeX — Mizar corpora with the informal part in increasing corpus size and resemblance to real natural language, while the last one corresponds to an ATP translation:

1. Synthetic LaTeX — Mizar dataset;
2. ProofWiki — Mizar topology dataset;
3. ProofWiki — Mizar full dataset;
4. Mizar — TPTP dataset.

In Section 2 we explain our decision to use Mizar as the formal corpus and share with the reader its ensuing strengths and weaknesses. We also give a detailed account of the four datasets, explaining how they are prepared and giving a few examples.

In Section 3 we first briefly explain the background of neural machine translation and cover works that are related to the three models we used. Then we give a high-level introduction to the three models to the point where we can apply them to our datasets.

In Section 4 we show our preliminary experimental results for training the translation models with our datasets. We evaluate the translation quality by several metrics: the BLEU rate and perplexity provided by the models, the distribution of edit distances, as well as visual comparison. Based on these we try to analyze their performance and point out the limitations of our current experiments.

In Section 5 a separate experiment is performed that incorporates a type elaboration mechanism with the supervised NMT model. We introduce our motivation for this experiment and then describe the type elaboration algorithm on pattern formulas.

There are few related works to autoformalization of mathematics. We discuss them in Section 6 and conclude our paper in Section 7.

2 Informal-To-Formal Datasets

The choice of LaTeX sentences as informal corpus is justified by the fact that most of the mathematics literature nowadays are written in this format. The choice of Mizar statements as formal corpus needs more explanation.

First of all, as our objective is to leverage data-driven machine learning models, our chosen proof assistant must already have a large database of mathematical knowledge from which we can prepare a formal corpus. Typical natural language corpus consists of millions of lines of sentences. By comparison, none of the existing proof assistants can provide a corpus of this size. Among all proof assistants, Mizar has more than 55 thousand theorem statements and more than 3 million lines of code in its Mizar Mathematical Library (MML) [6]. This is the largest formal library that the ITP community can provide that has a comparable size to a natural language corpus. The only contender is the

Isabelle library and its Archive of Formal Proofs [9], which has more formalized computer science knowledge, but less mathematics that can be overlaid with LaTeX texts. It is mainly because of the library size that we chose Mizar as our formal language.

Secondly, although it is still early to discuss foundational issues, we believe that the formal mathematics we are considering needs to be built on a formal theory that is capable to conveniently describe all contemporary mathematics. Indeed there are areas of mathematics (e.g., certain branches of set theory) where one can always formulate a statement that cannot be described in a fixed semantics, but for the purpose of formalizing mainstream mathematics the underlying formal theory needs to be at least able to describe large categorical notions since they are now pervasive in many areas of mathematics. This rules out several HOL-based systems, as from a denotational semantics point-of-view all non-polymorphic types are interpreted as members of a single fixed universe whose existence can be proved from ZFC, so the theory of HOL is weaker than the theory of ZFC [21], and ZFC is not enough for describing large categories without introducing the notion of class. On the other hand, Mizar is based on Tarski-Grothendieck set theory. The existence of universes is guaranteed by the Tarski A axiom [42], therefore there are no major issues in describing large categories.

There are also minor reasons that contributed to our decision to use Mizar: We want our formal language to be human readable so it can be easier for us to evaluate translation quality; We want to minimize unicode processing and the MML is ASCII-based; We also want the proof assistant library not to have a drastic syntactic difference between the theory language and its meta-language so that we can extract the maximum amount of information to formulate our corpus.

The above reasons for choosing Mizar do not mean that we think Mizar is better than other proof assistants. What we want to claim is that we think Mizar is currently the most *expedient* for us to test the effectiveness of neural machine translation models. We also do not claim that Mizar will be the formal language in a potential auto-formalization pipeline. Except the size of MML, all other reasons for choosing Mizar are simple and speculative. In fact, by choosing a proof assistant not based on type theory, we lose the opportunity to easily use automation later on to possibly further improve the quality of translation, which is why in Section 5 we need to use a different Mizar — TPTP dataset in order to test our type elaboration mechanism for data augmentation in the supervised learning setting.

2.1 The Synthetic LaTeX — Mizar Dataset

By using existing tools it is possible to extract logical information from the MML. Urban [43, 46] proposed a toolchain that can translate the MML into first-order proof obligations in the TPTP format [39]. These proof obligations can then be used by resolution-based first-order automated theorem

provers (ATP). The Mizar language is not purely first-order, its ability to form schemes of theorems as well as its soft-typing mechanism in formulation of statements make it possible to extract higher-order information from the MML. This was verified by Brown et al. [12] and the translated higher-order TPTP proof obligations were subsequently tested by a resolution-based higher-order ATP [11].

However, to prepare a LaTeX-to-Mizar dataset, techniques and experience gained from previous efforts cannot be directly applied. Although there are abundant mathematics literature in the LaTeX format that are freely available on the internet, parsing natural language to obtain logical information is known to be difficult. In the NLP community, preparing aligned data usually requires intensive manual labor, iterative data cleansing and huge computing resources. As a result, at the initial phase of our exploration it would be too costly for us if we went along this direction of directly facing the massive data.

To temporarily circumvent this problem, a possible approach is to generate informal sentences directly from the formal sentences we already have. Such an *informalization* approach has been adopted by Kaliszyk et al. [26, 27] to obtain aligned informal-to-formal corpora from the formal proof of the Kepler conjecture [22] in HOL Light and the MML. In those works informal corpora were made by merging overloaded symbols, deleting symbol prefixes, removing necessary parentheses, etc. However, this approach can only generate *ambiguated* informal sentences from formal sentences and they are not in the LaTeX format.

Table 1. The synthetic LaTeX — Mizar dataset

Total sentence pairs	1056478
Unique sentence pairs	605232
Shortest Mizar sentence	3 tokens
Shortest LaTeX sentence	3 tokens
Longest Mizar sentence	1526 tokens
Longest LaTeX sentence	19012 tokens

Fortunately, over the last decades the Mizar group has developed a tool that can be adapted to our purpose: To facilitate the publishing of Mizar's journal *Formalized Mathematics*, Bancerek et al. [5, 7] developed a tool that can translate theorems and top-level proof statements from a Mizar article into artificial LaTeX sentences. This tool first specifies translation patterns for all the Mizar constructors (e.g. functors, predicates, attributes, modes and structures) that are involved in that Mizar article. Then it uses a combination of algorithms to handle the use of parenthesis, the regrouping of conjunctive formulas, grammatical correctness as well as other typesetting issues. Finally, necessary style macros are added to polish the output rendering.

Comparing to other ITP-journals such as Isabelle's *Archive of Formal Proofs*, Mizar's *Formalized Mathematics* blurs the distinction between side remarks and formal code. The artificial LaTeX sentences are quite readable and the whole translated article has an outlook that is similar to a human-written article. We find these features appealing and spot the significance of Bancerek's tool for generating informal-to-formal corpora.

Bancerek's original tool could only translate Mizar abstracts, i.e., top-level theorem statements, definitions and registrations. In 2018, based on the early work of Urban [44], the Mizar group extended Bancerek's translation tool and applied it to all the Mizar articles [8]. To generate aligned LaTeX — Mizar sentences, this new tool keeps track of the offsets of Mizar statements throughout the intermediate steps, so the final LaTeX output can be traced back to its corresponding Mizar statement. We use this new tool to generate an aligned LaTeX — Mizar corpus of one million sentence pairs. Since the LaTeX corpus is artificially generated from Mizar corpus instead of collected from human-written LaTeX sentences, we call this dataset the *synthetic* dataset. Some statistics of this dataset are presented in Table 1 and example sentences are presented in Table 2.

As stated previously, obtaining aligned informal corpus from formal corpus algorithmically is only a temporary workaround to the data collection problem. Since all the translation techniques involved are deterministic, although the generated LaTeX sentences are grammatically correct, they usually look artificial and vocabulary used is smaller than in human-written sentences. This weakens the generalization capability of a translation model trained by only the synthetic dataset.

2.2 The Proof Wiki — Mizar Datasets

Translation models based on supervised learning require high-quality aligned corpora between source and target languages. Unsupervised learning claims to get rid of aligned corpora, so that only monolingual corpora are needed for training. However, at least for the two unsupervised models we are considering, a smaller aligned corpus for model validation and testing is still required. In addition, to maximize the translation quality, both languages from the aligned corpus need to come from the same topic realm where the monolingual corpora are prepared.

In UNMT, the authors used English and French news articles in WMT'14 dataset as monolingual corpora, and parallel English-to-French news translations in WMT'17 dataset for model evaluation. Since both are news articles, the linguistic variety and the vocabulary used are at the same range. To adapt this model for our purpose, we also need to have sizable LaTeX and Mizar corpora with a subset of the two corpora aligned. A simple solution to this requirement is to just use the previous synthetic dataset, keeping the alignment of a subset of it and forgetting the alignment of the

rest. However, in order for us to cope with the limitations of using the synthetic dataset and to explore the capability of the unsupervised models, we also manually prepared other datasets.

The new ProofWiki — Mizar datasets were prepared by collecting sentences from MML and *proofwiki.org*. ProofWiki is a website that contains more than 26K theorems with detailed human-written proofs. These proofs have been collectively contributed by many users over the years and are written in MathJax, a browser-adapted LaTeX format. During 2015–2016, Grzegorz Bancerek embarked on an individual project to align ProofWiki theorems with Mizar theorems. Bancerek manually aligned 470 theorems, more specifically theorem statements that appear in both MML and ProofWiki. All the aligned theorems were in point-set topology and lattice theory.

Thanks to this effort we are able to prepare a dataset similar to the English-to-French dataset used in UNMT. This ProofWiki — Mizar dataset contains two monolingual corpora, one in the Mizar language and the other in ProofWiki's LaTeX. These two monolingual corpora need not be aligned. The ProofWiki — Mizar dataset also contains a smaller aligned corpus consisting of theorem statements extracted from Bancerek's manual alignment. Since Bancerek's alignment only focuses on topology and lattice theory, we also picked sentences from the monolingual corpora that are from topology and lattice theory and form a dataset with the same aligned corpus but two smaller monolingual corpora. Some statistics of these datasets are presented in Table 3 and sentence examples are given in Table 5.

2.3 The Mizar — TPTP Dataset

As mentioned earlier, a serious disadvantage of using the Mizar language as our formal language is that we lose the ability to further improve the translation quality by harnessing Mizar's own toolchain. Adding to the fact that the Mizar engine is closed-source, Mizar texts are notoriously hard to compile without a proper environment declaration, let alone a single Mizar statement without any external reference. A correct environment declaration requires a conceptual understanding of the internals of the Mizar engine together with a lot of domain expertise. This would be too challenging for us at the current stage if we want to automate the lookup of the environment declaratives.

To bypass this difficulty while fully exploring the capability of neural machine translation models, we design a different experiment in which the supervised NMT model takes Mizar statements as input and outputs TPTP formulas. These TPTP formulas, after preprocessing, can be fed into a custom type elaboration tool to determine the correctness of a Mizar statement. The aligned Mizar — TPTP dataset can be generated with the MPTP toolchain [46]. The size of the dataset is presented in Table 4.

Table 2. Mizar vs. Synthetic LaTeX

Mizar	`cluster reflexive -> complete for 1 -element RelStr ;`
LaTeX	One can verify that every 1-element relational structure which is reflexive is also complete .
LaTeX	One can verify that every 1-element relational structure which is reflexive is also complete .
Mizar	`let T be RelStr ;`
LaTeX	Let $ T $ be a relational structure .
LaTeX	Let T be a relational structure .
Mizar	`mode type of T is Element of T ;`
LaTeX	{ A type of $ T $ } is an element of $ T $.
LaTeX	A type of T is an element of T .
Mizar	`attr T is Noetherian means : Def1 : the InternalRel of T is co-well_founded ;`
LaTeX	We say that { $ T $ is Noetherian } if and only if (Def . 1) the internal relation of $ T $ is reversely well founded .
LaTeX	We say that T is Noetherian if and only if (Def . 1) the internal relation of T is reversely well founded .
Mizar	`a ast ( b ast t ) <= b ast t ;`
LaTeX	$ a \ast (b \ast t) \leq b \ast t $.
LaTeX	$a * (b * t) \leq b * t$.
Mizar	`'not' Ex ( Ex ( a , A , G ) , B , G ) = All ( 'not' Ex ( a , B , G ) , A , G ) ;`
LaTeX	$ \neg { \exists _ { \exists _ { a , A } G , B } } G = { \forall _ { \neg { \exists _ { a , B } } G , A } } G $.
LaTeX	$\neg \exists_{\exists_{a,A}G,B} G = \forall_{\neg \exists_{a,B}G,A} G$.

Table 3. The ProofWiki — Mizar datasets

ProofWiki — Mizar full dataset	
Total sentences in monolingual Mizar	1056478
Unique sentences in monolingual Mizar	1035511
Total sentences in monolingual ProofWiki	198801
Unique sentences in monolingual ProofWiki	198225
Total sentences in pairs	330

ProofWiki — Mizar topology dataset	
Total sentences in monolingual Mizar	49408
Unique sentences in monolingual Mizar	48774
Total sentences in monolingual ProofWiki	29763
Unique sentences in monolingual ProofWiki	29618
Sentences in pairs	330

Table 4. The Mizar — TPTP dataset

Total sentence pairs	53994
Unique sentence pairs	53994
Shortest Mizar sentence	3 tokens
Shortest TPTP sentence	2 tokens
Longest Mizar sentence	986 tokens
Longest TPTP sentence	786 tokens

3 Neural Machine Translation Models

3.1 Brief Overview

Theoretical foundations demonstrating the expressiveness of neural networks date back to late 1980s. In 1989, Cybenko [17] and Hornik at al. [25] showed that any measurable function between Euclidean spaces can be uniformly approximated by a composition of an affine mapping together with a non-linear mapping satisfying certain mild restrictions. Hornik (1991) [24] further generalized the underlying spaces and relaxed the restrictions on the non-linear mapping. Due to the limitation of computing resources as well as the scarcity of large datasets, we did not see competitive applications based on neural network until the seminal

Table 5. Sample aligned ProofWiki — Mizar theorem statements

Mizar	`for T being non empty TopSpace for A being Subset of T st A is countable holds A ^0 = { }`
LaTeX	Let T = \left ({ S , \tau } \right) be a topological space . Let A be a subset of S . Then if A is countable , then A ^ 0 = \varnothing .
Mizar	`for T being non empty TopSpace for A , B being Subset of T holds ( A \/ B ) ^0 = ( A ^0 ) \/ ( B ^0 )`
LaTeX	Let T = \left ({ S , \tau } \right) be a topological space . Let A , B be subsets of S . Then \left ({ A \cup B } \right) ^ 0 = A ^ 0 \cup B ^ 0

Table 6. Sample Mizar — TPTP statements

Mizar (Source)	`for A holds A is doubleLoopStr & not A is empty implies for B holds B is Scalar of A implies B is being_a_square iff ex C st C is Scalar of A & B = C ^2`
Prefix (Target)	`c! b0 c=>__2 c&__2 cnl6_algstr_0__1 b0 c~__1 cnv2_struct_0__1 b0 c! b1 c=>__2 cnm4_vectsp_1__2 b1 b0 c<=>__2 cnv1_o_ring_1__1 b1 c? b2 c&__2 cnm4_vectsp_1__2 b2 b0 cnr1_hidden__2 b1 cnk1_o_ring_1__1 b2`
TPTP-FOF	`fof(a1,elaborate,(! [A0] : ((nl6_algstr_0(A0) & (~ nv2_struct_0(A0))) => (! [A1] : (nm4_vectsp_1(A1,A0) => (nv1_o_ring_1(A1) <=> (? [A2] : (nm4_vectsp_1(A2,A0) & nr1_hidden(A1,nk1_o_ring_1(A2)))))))))).`
TPTP-THF	`thf(a1,elaborate, (! [A:$i]:(((nl6_algstr_0 @ A) & ~ ((nv2_struct_0 @ A))) => (! [B:$i] : ((nm4_vectsp_1 @ B @ A) => ((nv1_o_ring_1 @ B) <=> (? [C:$i] : ((nm4_vectsp_1 @ C @ A) & (nr1_hidden @ B @ (nk1_o_ring_1 @ C)))))))))).`
Elaborated-FOF	`fof(a1,conjecture,(! [X0] : ((~(l6_algstr_0(X0) => (~(~v2_struct_0(X0)))))) => (! [X1] : (m1_subset_1(X1,u1_struct_0(X0)) => (v1_o_ring_1(X1,X0) <=> (~(! [X2] : (~(~(m1_subset_1(X2,u1_struct_0(X0)) => (~r1_hidden(X1,k1_o_ring_1(X0,X2))))))))))))).`

work of AlexNet in 2012 [28] which harnessed the computational power of GPU and the ImageNet dataset. Since then, machine learning based on deep neural networks has gained tremendous popularity in a wide range of areas from computer vision [28] to natural language translation [14] and even to mastering the game of Go [38]. During these years we have also witnessed progress in learning theory: Telgarsky (2015) [41] and Safran et al. (2016) [37] showed that the increase of depth in a deep neural network can exponentially reduce the width of the affine maps at each layer and Du (2018) [19] discovered the global convergence of gradient descent in an over-parameterized neural network.

In neural machine translation, a series of works published around 2014 has laid out the foundation for architecture based on recurrent neural networks (RNNs): Chung at al. first extended RNN with sophisticated recurrent units that implement a gating mechanism [14]; Sutskever at al. [40] incorporated the Long Short-Term Memory cell (LSTM) [23] into RNN and discovered the importance of bi-directional encoding; Bahdanau et al. [4] and Luong at al. [32] discovered the attention mechanism and demonstrated its effectiveness

in handling long sequences. These contributions eventually led to Luong's NMT implementation [31].

Meanwhile architectures that are not based on RNN have also been developed: Gehring et al. (2017) [20] applied convolutional neural networks (CNNs) in sequence to sequence translation. Vaswani et al. (2017) [47] invented a self-attention mechanism called the Transformer that overcomes the lack of parallelization in RNN-based training. This parallelization of encoding also enabled the use of pre-training which led to the seminal BERT model developed by Devlin et al. [18] The UNMT model [30] we are using is based on the Transformer architecture, adding a novel back translation technique for unsupervised learning. The XLM model [29] extends the UNMT model with BERT-style pretraining. Unsupervised learning of UNMT is then performed on the pre-trained model to fine-tune the model parameters.

3.2 RNN-Based NMT

The fundamental idea behind neural machine translation is to encode source and target sentences as lists of numeral vectors and feed them into a network architecture for computation of a loss function. Since the neural network is essentially a

differentiable function, training the neural network amounts to minimization of the loss function. This is a gradient optimization problem and in neural networks the gradients of network parameters are computed by the back-propagation algorithm.

At the top level, the NMT architecture consists of an encoder network and a decoder network, each of which is a multi-layer recurrent neural network with LSTM cell as its network cells.

During training, the encoder takes in a source sentence one word vector at a time. After all the word vectors have been put in, an end-of-sentence marker initiates the decoder which takes in the hidden vector of the encoder. The decoder produces a target sentence one word vector at a time and simultaneously takes in the correct target sentence. The evaluated target sentence is then compared with the correct target sentence and the loss value is computed.

During inference, only a source sentence is needed. The source sentence is first taken by the encoder. The encoder then passes the hidden vector to the decoder. The decoder then produces the evaluated target sentence one word vector at a time. Each time a word vector is generated it is put directly back to the decoder input for the evaluation of the next word vector. There are two ways to pick a word vector. First, given a probability distribution of words, the *greedy* approach always picks the next word as the one with the highest probability. Second, the *beam search* approach also picks the next word from the probability distribution of words, but it is chosen in such a way the sentences with the highest n joint probabilities are kept (n is the *width* of the beam search). We use the greedy approach in our model comparison experiments for the fairness of the comparison, while we will use the beam search approach to further augment the data in Section 5.

One crucial add-on to the encoder-decoder architecture is the attention mechanism, which can be considered as a side network connecting the encoder and the decoder, forming a larger network. The attention network records the hidden vector at each word input of the source sentence. Depending on the type of attention, different computation steps are involved to obtain an attention vector. This attention vector is then fed into the decoder to affect the generation of the target sentence.

Our NMT experiment uses 2-layer LSTM cells in both the encoder and the decoder. The dimension of hidden vectors in a LSTM cell is 1024. We used scaled Luong's attention, bi-directional encoding and 20% dropout rate. Due to the size of this network we cannot fit into a single GPU, so we train our model in CPU-only mode. The total training time is 17.9 hours. The synthetic dataset is divided into a 90:10 ratio between training and inference set. An extra 4000 sentence pairs are reserved, in which 2000 are used for model validation and the remaining half are used for testing and model evaluation.

3.3 UNMT
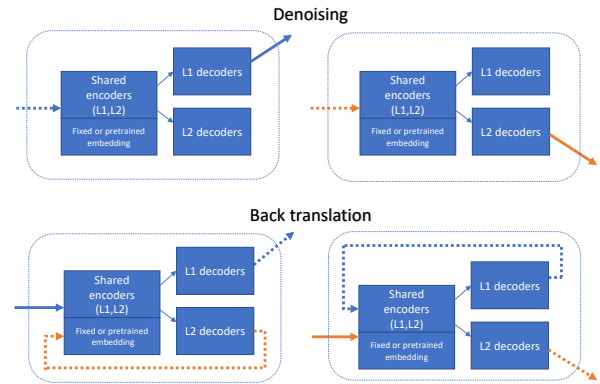
Figure 1. Denoising and back translation in UNMT and XLM

The key idea of the unsupervised learning according to Lample et al. [30] is to transform the unsupervised learning problem into a series of supervised learning problems. This idea is illustrated in Figure 1. First, the two monolingual corpora are concatenated and are processed by fastText [10], a word2vec [33] implementation for learning word representation. This bilingual word representation is kept fixed throughout the training and evaluation phase.

The neural network architecture of UNMT consists of a shared encoder and two decoders. To initialize the parameter values, a denoising step is performed: for each monolingual corpus, we corrupt each sentence by randomly permuting and dropping a few words. We then use the corrupted corpus as source language and the original non-corrupted corpus as target language. This gives us two aligned corpora. We use one of them for initializing the parameters of the shared encoder and the L1 decoder. And the other for refining the parameters of the shared encoder and initializing the parameters of the L2 decoder.

The training is done using a back translation technique. For each sentence in language L1 as input to the neural network, we first generate a sentence in L2 from the L2 decoder. This generated sentence is then put back into the shared encoder again and we now generate a sentence in L1 from the L2 decoder. The combined result is a merged network taking L1 sentence as input and generating L1 sentence as output, and this can be treated as a supervised problem. Similarly we can do the same on L2, generating L2 → L1 → L2 data flow and obtain an L2 → L2 supervised problem. The UNMT model is trained in alternative fashion: a few iterations L1 → L1 followed by a few iterations of L2 → L2. This alternation continues until a reasonable stopping criterion is met.

In UNMT we can configure the encoder-decoder architecture as either based on RNN or based on Transformer. In our experiments we use the Transformer architecture, since it provides a similar quality, while allowing for more efficient training on GPU architectures. The key feature in a

Transformer block is a multi-headed attention cell, which is a multi-copy version of so called self-attention cells. A self-attention cell computes several feature vectors for each word in a sentence in parallel. Then a cross-weighting and averaging step is computed. Since the weighting step is computed in parallel and the input sentence is read in parallel, the Transformer cell can maximally harness the power of GPU and is usually trained much faster than RNN.

The UNMT model was trained on one NVIDIA GeForce GTX 1080 Ti Graphics Card. The training times are 30 minutes for the ProofWiki — Mizar topology dataset and 1 hour 8 minutes for the ProofWiki — Mizar full dataset.

3.4 XLM

The key difference between UNMT and XLM is that in UNMT the word embedding is fixed throughout the training process while in XLM it is also pre-trained. XLM uses Masked Language Model (MLM) method borrowed from BERT [18]. In MLM, each word in a sentence has 15% chance of being selected. If a word is selected, it has 80% chance of being substituted by a [MASK] token; 10% chance of being substituted by a random token; and 10% chance being unchanged. After pre-training, back translation similar to UNMT is performed to fine-tune the parameters in both encoders and decoders.

The XLM model was also trained on one NVIDIA GeForce GTX 1080 Ti Graphics Card. The training times are 4 hours 24 minutes for the ProofWiki — Mizar topology dataset and 7 hour 53 minutes for the ProofWiki — Mizar full dataset.

4 Experiments and Results

We compare the three models introduced in Section 3 by three metrics that are commonly used in evaluating the quality of language translation models: the BLEU rate [34], the perplexity score and the Levenshtein edit distance.

The calculation of the BLEU rate involves counting of n-grams, geometric averaging, as well as an exponentially scaled length penalty. It computes a score from 0 to 100, with larger values indicating better translation quality. In NLP research, bilingual machine translation attains a typical BLEU score ranging from 25.16 to 40.51 depending on the languages (see Table 1. of [20]).

The perplexity also involves counting of n-grams with exponentials. It intuitively captures the idea of the number of words randomly picked before obtaining the correct word. It is a positive number with smaller value denoting better translation quality. In Table 4 and 5 of [20] we witnessed the perplexities ranging between 6 to 7 in bilingual translation.

The edit distance is also a very common evaluation metric. It calculates the number of insertions, deletions and substitutions from a translated sentence to its corresponding correct sentence. The calculation is done using a dynamic programming algorithm. The smaller the edit distance is, the closer the translated sentence to correctness.

It is noted that all the above three metrics only compare the syntactic closeness of a translated corpus from the reference corpus. Although in terms of formalization evaluating syntactic closeness is primitive (e.g. missing a 'not' in the statement negates its meaning), for convenience and speed we temporarily borrow what is available in existing NLP research. It would be interesting if new techniques on measuring closeness of formal statements by semantics could appear.

Table 7 gives the metrics of a 2000-sample evaluation on the synthetic dataset. Because of the maximum use of alignment, we can see that NMT performs consistently better than both UNMT and XLM in all metrics. As XLM uses a pre-training step instead of fixing a word embedding, we also anticipate that XLM performs better than UNMT. This is indeed verified by the results shown.

For truly non-aligned datasets, we can only compare the two unsupervised models (Table 8). Due to the small size of our aligned corpus (as well as the fact that each sentence pair is generally longer than the synthetic dataset), we witness a significant decrease of performance in all the metrics. However, if we compare within the dataset, we can still see a trend that, except the surprising perplexity score, XLM performs generally better than UNMT. In addition, since our aligned corpus is only focused on topology and lattice theory, we see the trend that no matter which model we used, the metrics show better results on the topology sub-dataset instead of the full ProofWiki — Mizar dataset.

Table 7. Model evaluation on the synthetic dataset

2000 samples	NMT	UNMT	XLM
BLEU	70.9	27.14	43.61
Perplexity	1.58	3.00	2.91
Edit distance 0	65.2%	26.8%	34.1%
Edit distance ≤1	74.6%	34.4%	38.5%
Edit distance ≤2	81.5%	41.8%	42.1%
Edit distance ≤3	83.9%	46.3%	45.9%

5 Data Augmentation Using Type Elaboration

The major motivation of this experiment is to form a feedback loop such that the evaluated target sentence can be deterministically transformed back to a source sentence. By using beam search during the inference phase of the NMT model, we can generate more than one target sentence. If some of these target sentences can be translated back to the source sentences, we can add the new sentence pairs into our training dataset and retrain our model. We anticipate that the model with augmented data will have more syntactically

Table 8. Model evaluation on the Proofwiki — Mizar dataset

131 samples	UNMT		XLM	
	topology	full	topology	full
BLEU	4.03	1.55	7.87	6.07
Perplexity	11.57	10.73	33.01	39.70
Edit distance 0	0%	0%	0%	0%
Edit distance ≤1	0%	0%	0%	0%
Edit distance ≤2	0%	0%	0.76%	0%
Edit distance ≤3	9.92%	2.29%	6.11%	2.29%

correct translation, or in a way, we manage to "teach" the model the correct syntactic format.

In Section 5.1 we shortly recall the soft type system in order to cover the internal mechanism of our ELABORATOR that does Mizar-style type checking in Section 5.2. In Section 5.3 we illustrate how pseudo-patterns in a TPTP-THF formula are *type-checked* by the ELABORATOR. We integrate the ELABORATOR and our data transformation pipeline into the feedback loop in Section 5.4 and show experimental results in Section 5.5.

5.1 Mizar Soft Type System

All Mizar types are represented as predicates (soft types) in MPTP [46]. Hence even a Mizar type corresponding to the real numbers would be represented by a predicate over a type of individuals (sets).

Suppose we are given a binary operation + (written in infix), a constant $\hat{1}$ and a unary predicate real. For each positive natural number n, let $\hat{n}$ be the term $\hat{1} + \cdots + \hat{1}$ (associating to the left). Consider the following three formulas.

1. real $\hat{1}$
2. $\forall xy.\text{real } x \; \rightarrow \; \text{real } y \; \rightarrow \; \text{real } (x + y)$
3. $\forall xy.\text{real } x \; \rightarrow \; \text{real } y \; \rightarrow \; x + y = y + x$

In order to prove the conjecture $\forall x.\text{real } x \; \rightarrow \; x + \hat{n} = \hat{n} + x$ a theorem prover would need to instantiate using the commutativity axiom and then prove real $\hat{n}$. Proving real $\hat{n}$ requires applying the second axiom $n - 1$ times.

The formula above indicating the sum of two reals is real is an example of a "function type" axiom, indicating that if the inputs of a function has certain soft types then the result will have certain soft types. A different kind of soft typing axiom is a "type hierarchy" axiom, indicating that if a term has certain soft types then it also has certain "inherited" soft types. We can extend the example to include a type hierarchy axiom by adding a predicate for natural numbers.

Suppose we add a unary predicate nat and change the axioms of the problem to be the following:

1. (Function type for $\hat{1}$) nat $\hat{1}$

2. (real function type for +)
   $\forall xy.\text{real } x \; \rightarrow \; \text{real } y \; \rightarrow \; \text{real } (x + y)$
3. (nat function type for +)
   $\forall xy.\text{nat } x \; \rightarrow \; \text{nat } y \; \rightarrow \; \text{nat } (x + y)$
4. (Type hierarchy for nat) $\forall x.\text{nat } x \; \rightarrow \; \text{real } x$
5. $\forall xy.\text{real } x \; \rightarrow \; \text{real } y \; \rightarrow \; x + y = y + x$

The conjecture $\forall x.\text{real } x \; \rightarrow \; x + \hat{n} = \hat{n} + x$ still follows. In this case, however, a proof must make use of the type hierarchy axiom to infer that $\hat{n}$ is real. There are actually multiple derivations. At one extreme one can infer that $\hat{1}$ is real and use the real function typing axiom for $+ \; n - 1$ times. At the other extreme one can use the nat function typing axiom for $+ \; n - 1$ times to infer that $\hat{n}$ is a natural number and then infer that $\hat{n}$ is real. For a general theorem proving procedure this redundancy leads to a needless expansion of the search space. By treating soft types as special, the fact that $\hat{n}$ is real will be computed by the (deterministic) soft typing algorithm.

Until now we have only considered the unary predicates real and nat as soft types. In order to handle predicates corresponding to Mizar soft types we must be able to handle predicates with extra dependencies. For example, in Mizar there are types "Element of X" and "Function of A, B." These could appear as a binary predicate elt and a ternary predicate func in a problem, where we write elt $x \; X$ to mean x has type "Element of X" and write func $f \; A \; B$ to mean f has type "Function of A, B." We have also considered variants of our example with soft typing axioms such as

- (Function type for ap) $\forall abf.\text{func } f \; a \; b \; \rightarrow \; \forall x.\text{elt } x \; a \; \rightarrow$ elt (ap $a \; b \; f \; x$) b

where ap is a 4-ary operator for set theoretic function application.

5.2 Mizar Soft Type Inference

In [45] the notion of a (first-order) "Mizar-like Horn theory" is defined, distinguishing between a "type hierarchy part" and "functor types part." We lift this to the higher-order setting by carving out appropriate first-order formulas from higher-order formulas [15] satisfying similar conditions.

- A *first-order term* is either a variable (of base type) or a term of the form $f \; t_1 \ldots t_n$ where f is an n-ary function (taking n terms of base type to a term of base type) and t_i is a first-order term for each $i \in \{1, \ldots, n\}$.
- A *first-order atom* is $p \; t_1 \cdots t_n$ where $n \geq 1$, p is an n-ary predicate over terms of base type and t_i is a first-order term for each $i \in \{1, \ldots, n\}$. Given a first-order atom A of the form $p \; t_1 \cdots t_n$, we say p is the predicate of A and write $\text{pred}^A = p$ and say t_1 is the primary term of A and write $\text{prim}^A = t_1$.
- A *first-order literal* is either a first-order atom or the negation of a first-order atom. We extend pred^L and prim^L to be $\text{pred}^{\neg A} = \text{pred}^A$ and $\text{prim}^{\neg A} = \text{prim}^A$ and call pred^L the predicate of L and prim^L the primary

term of L. We say L has positive polarity if it is an atom and negative polarity otherwise.

In [45] a chronological total order $<$ on functor and type symbols is used to add appropriate conditions on Mizar-like Horn clauses. Here we instead work with an strict partial order $\prec$ which is computed dynamically while determining the classification of axioms.

Let $\prec$ be a strict partial order on constants. Let φ be a formula

$$\forall x_1 \cdots \forall x_n.L_1 \wedge \cdots \wedge L_m \rightarrow L_{m+1} \wedge \cdots \wedge L_{m+k}$$

- We say φ is a *type hierarchy formula* if
  1. $n \geq 1$,
  2. $m > 1$,
  3. L_m is either of the form $px_1 \ldots x_n$ or $\neg px_1 \ldots x_n$,
  4. for each $i \in \{1, \ldots, m-1\}$ $\mathrm{prim}^{L_i} \in \{x_1, \ldots, x_{n-1}\}$ and x_n is not free in L_i and
  5. for each $i \in \{m+1, \ldots, m+k\}$ we have $\mathrm{prim}^{L_i} = x_n$ and $\mathrm{pred}^{L_i} \prec p$.
- We say φ is a *function type formula (for f)* if for each $i \in \{1, \ldots, m\}$
  1. $\mathrm{prim}^{L_i} \in \{x_1, \ldots, x_n\}$ and
  2. for each $j \in \{m+1, \ldots, m+k\}$ prim^{L_j} is $f\, x_1 \cdots x_n$ and $\mathrm{pred}^{L_i} \prec f$.

We now describe the soft typing inference mechanism designed for a type system given by function type and type hierarchy formulas. We first describe a procedure for "widening" a given soft type and then describe how all soft types of t are computed.

Suppose we have computed a soft typing literal L. We can compute new soft types by a "widening" procedure making use of type hierarchy formulas as follows.

- Find all type hierarchy formulas

  $$\forall x_1 \cdots \forall x_n.L_1 \wedge \cdots \wedge L_m \rightarrow L_{m+1} \wedge \cdots \wedge L_{m+k}$$

  where L_m has the same predicate and polarity as L. Let θ be a substitution such that $\theta(L_m) = L$. (The restrictions on type hierarchy formulas ensure such a θ exists.) Continue if $\theta(L_j)$ has already been computed to be a soft type of its principle term for each $j \in \{1, \ldots, m-1\}$.[1]
- For each $j \in \{m+1, \ldots, m+k\}$, $\theta(L_j)$ is a new soft type for the same term as the primary term of L. We add this as a known soft type and recursively widen each such $\theta(L_j)$.[2]

We now describe the algorithm for computing all soft types of a ground first-order term t. The algorithm proceeds by recursion.

___
[1]Note that this may give different results depending on the order in which soft types have been computed.
[2]Note that the $\prec$ relation guarantees termination.

1. Since t is ground, it must have the form $f\, t_1\, \ldots\, t_n$ where each t_i is a ground first-order term. Assume all soft types of each t_i have been computed.
2. We next find all function type formulas for f, e.g.,

   $$\forall x_1 \cdots \forall x_n.L_1 \wedge \cdots \wedge L_m \rightarrow L_{m+1} \wedge \cdots \wedge L_{m+k}$$

   where $f(x_1, \ldots, x_n)$ is the primary term of L_j for $j \in \{m+1, \ldots, m+k\}$. Let θ be the substitution with $\theta(x_i) = t_i$ for $i \in \{1, \ldots, n\}$.
   a. Note that for each $j \in \{1, \ldots, m\}$ the principle term of $\theta(L_j)$ is some t_i and we have already computed the soft types of t_i. Continue if $\theta(L_j)$ is a known soft typing literal for each $j \in \{1, \ldots, m\}$.
   b. For each $j \in \{m+1, \ldots, m+k\}$, add $\theta(L_j)$ as a known soft typing literal and call the widening procedure with $\theta(L_j)$.

The procedure described above is a simplification of the one used by the ELABORATOR, as there are still more soft typing formulas than function type and type hierarchy formulas. For example, there are redefinition formulas allowing one to declare that a new function is defined to be the same as some previous function but with new typing information for the new function. The equations introduced by redefinitions require introducing some degree of equational reasoning within the soft typing inference procedures. For brevity we omit details.

5.3 ELABORATOR — Patterns and Elaboration

In order to allow for partially specified terms and formulas, we extend the language to allow *pattern names*. A pattern name is used as a predicate or function with a fixed arity, but should not be thought of as having a clear intended semantics. Instead each pattern name will be associated with a different *constructor name* which can be thought of semantically. We call terms and formulas with occurrences of pattern names *pseudo-terms* and *pseudo-formulas*. Our primary use of ELABORATOR is to try to compute formulas corresponding to pseudo-formulas produced by the translation mechanism.

Recall the constructor name ap with function type formula

$$\forall ab f.\mathrm{func}\, f\, a\, b \rightarrow \forall x.\mathrm{elt}\, x\, a \rightarrow \mathrm{elt}\, (\mathrm{ap}\, a\, b\, f\, x)\, b.$$

Here func and elt are also constructor names. For each of these constructor names there will be corresponding pattern names $\underline{\mathrm{ap}}$, $\underline{\mathrm{func}}$ and $\underline{\mathrm{elt}}$. Along with the formulas defining the type system we give ELABORATOR a pseudo-formula for each pattern name. For the pattern names above, three pseudo-formulas similar to the following would be given.

1. $\forall AB.\underline{\mathrm{elt}}\, B\, A \equiv \mathrm{elt}\, B\, A$.
2. $\forall ABf.\underline{\mathrm{func}}\, f\, A\, B \equiv \mathrm{func}\, f\, A\, B$.
3. $\forall CDfc.\mathrm{func}\, f\, C\, D \wedge \mathrm{elt}\, c\, C \rightarrow \underline{\mathrm{ap}}\, f\, c = \mathrm{ap}\, C\, D\, f\, c$.

In each case there is an equation or equivalence indicating how to elaborate a pattern name into a term or formula using corresponding constructor name, possibly guarded by soft

typing constraints on the variables. In the case of $\underline{ap}$ the soft typing constraints determine the extra two arguments required by ap.

Given these pattern pseudo-formulas in addition to the typing formulas, the algorithm used by the ELABORATOR to elaborate pseudo-formulas into formulas is relatively straight-forward. The essential idea is to recursively traverse pseudo-formulas and pseudo-terms elaborating from the bottom up. At each step there will be type constraints required by the pattern pseudo-formulas and the type checker is used to check if these type constraints are satisfied. If the type constraints are satisfied, terms for the missing arguments will be determined. When traversing binders, introduce a fresh name to replace the variable and assume this fresh name has all appropriate soft typing assumption. For example, to elaborate $\forall x.\varphi(x) \rightarrow \psi(x)$, we extract the parts of φ that correspond to soft typing literals, create a fresh name a, assume a satisfies these soft typing literals, and then continue by trying to elaborate $\varphi(a) \rightarrow \psi(a)$. Similar remarks apply when attempting to elaborate $\exists x.\varphi(x) \wedge \psi(x)$.

Example 5.1. Suppose we wish to elaborate the following pseudo-formula:

$$\forall Ax.\underline{elt}\ x\ A \rightarrow \exists f.\underline{func}\ f\ A\ A \wedge \underline{ap}\ f\ x = x.$$

The elaborated formula is

$$\forall Ax.elt\ x\ A \rightarrow \exists f.func\ f\ A\ A \wedge ap\ A\ A\ f\ x = x.$$

This elaborated formula is determined as follows. We choose fresh names a and y for A and x and try to elaborate the antecedent of the implication: $\underline{elt}\ y\ a$. This succeeds giving elt $y\ a$. We recognize this as a typing literal and assume this as typing information about y. We then begin elaborating the succedent of the implication. We choose a fresh name g for f and attempt to elaborate the left side of the conjunction: $\underline{func}\ g\ a\ a$. This succeeds giving func $g\ a\ a$ which is then assumed as type information about g. We finally elaborate $\underline{ap}\ g\ y = y$. To elaborate $\underline{ap}\ g\ y$ we must find types of g and y of the form func $g\ Y\ Z$ and elt $y\ Y$. Using the typing assumptions above we know func $g\ a\ a$ and elt $y\ a$. Hence we can elaborate $\underline{ap}\ g\ y$ as ap $a\ a\ g\ y$. When passing back through the quantifiers, the fresh names are replaced with the original bound variables.

5.4 Data Augmentation Pipeline

Figure 2 illustrates the idea of the feedback loop mentioned above. In the chart the x-axis denotes edit distances while the y-axis denotes the number of sentences inferred. We use the Mizar — TPTP dataset as described in Section 2.3. Instead of using the standard TPTP first-order format, we use a prefix notation. This notation not only largely reduces the need for matching parenthesis, but it also reduces the overall length of the TPTP sentence. This has been previously found to benefit the neural translation task [36]. Sample prefix notation as
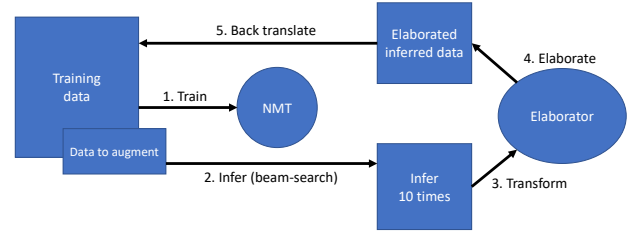

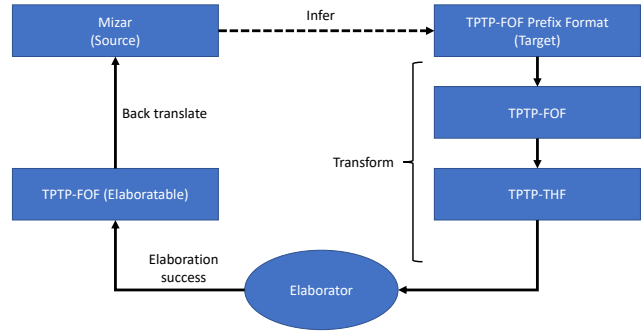**Figure 2.** Feedback loop Supervised NMT + ELABORATOR

Figure 3. Format transformation pipeline

well as its corresponding TPTP statements can be seen in Table 6.

Our ELABORATOR takes a TPTP-THF pseudo-formula as input and returns a boolean value indicating whether the pseudo-formula can be elaborated or not. To use the ELABORATOR we translate the TPTP-FOF prefix format into the TPTP-THF format by a series of Prolog programs and related scripts. If the pseudo-formula can be elaborated, the ELABORATOR will also output the corresponding elaborated TPTP-FOF formula. In that case, the original pseudo-formula is back-translated by the MPTP toolchain into a Mizar statement, forming the feedback loop. Figure 3 illustrates the transformation of the file formats in the feedback loop.

5.5 Experiments

Table 9. Model evaluation on the Mizar-TPTP dataset

2000 samples	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5
BLEU	42.3	85.5	88.7	88.0	87.4
Perplexity	2.68	1.27	1.14	1.18	1.17
Edit distance 0	2.05%	2.40%	2.00%	5.15%	4.45%
Edit distance ≤1	9.70%	13.4%	20.9%	21.6%	19.2%
Edit distance ≤2	22.3%	25.0%	38.7%	36.9%	34.6%
Edit distance ≤3	32.6%	34.3%	49.3%	48.5%	45.4%

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Iteration 1	41	153	251	206	141	104	85	71	60	65	42	37	38	42	31	24	47	41	25	33
Iteration 2	48	220	232	186	181	104	80	63	50	58	52	41	41	35	38	19	33	24	21	25
Iteration 3	100	318	355	213	140	88	56	56	52	34	33	29	35	23	20	26	18	14	20	17
Iteration 4	103	329	305	232	141	85	64	48	36	47	25	21	21	37	28	21	16	22	25	18
Iteration 5	89	294	308	216	147	74	71	53	46	34	41	31	27	27	28	20	27	13	18	20

Edit Distance

Figure 4. Distribution of edit distances in the elaboration experiment (total 2000 sentences)

We evaluate the result of the data augmentation toolchain based on type elaboration by comparing the distribution of edit distances across the first five iterations of the feedback loop.

In Figure 4 we can see that there is nearly an 100% increase of values at edit distance 0 (exact match), indicating an improvement of translation quality. The whole distribution curves in the first three iterations are becoming more skewed to the left, indicating a general improvement of translation quality toward exact matches. However, as we can also see from Table 9, starting from the fourth iteration the improvement is no longer significant. The fourth iteration has the highest concentration of edit distance up to 2. After that the distribution begins to drop and flatten. This indicates that our trained model is beginning to overfit. Such an overfit is likely to happen given limited training data. This is unfortunate. However, it is notable that from from Iteration 1 to Iteration 3, we have only had a 16.41% increase in the size of the training data but we achieved a 100% increase in the number of exact matches. This shows that data augmentation based on strong semantic methods such as type elaboration can be very useful.

6 Related Works

There are several works directly related to autoformalization of mathematics. The earliest goes back to Wang (1954) [48] which hinted at the potential automation of formalization of mathematics. In 1994, a group of anonymous authors declared the *QED Manifesto* [2] which set out the vision of formalization of all mathematical knowledge in a computer-based database. Zinn (2004) [50] used domain discourse theory to parse a number theory textbook. The Naproche system [16] invented a controlled natural language that looks very similar to normal natural language text but compilable

to proof obligations for automated theorem prover. The vision set out in the QED Manifesto is exactly what the authors are trying to achieve. It is our hope that the recent advent of neural network-based machine learning can finally provide a push to its realization.

7 Conclusion

In this paper we have proposed autoformalization of mathematics and performed first experiments using unsupervised machine learning for translating LaTeX to formal proof assistant statements. We have also proposed a soft type elaboration mechanism for data augmentation and integrated it into the supervised learning framework for autoformalization. We practically evaluated various approaches and showed the results. We think this is a promising direction and envision the possible future of the formalization community and the AI community embracing each other.

There are several directions that are worthy of further investigation: a primary concern is collecting more high-quality data and conducting more comprehensive experiments. It will be good if we can merge the ideas from new models and quickly test them in a single pipeline. This could help us push to the extreme the capability of neural machine translation on our informal-to-formal datasets.

To adapt the existing models the format of our datasets is based on sequences only. We have so far not experimented with the inclusion of other features such as the syntax tree of a formula or the required environment declaration of a statement. This enriched information could have the potential to enrich the translation quality. It could also open possibility for us to explore other non-sequential input formats.

As we have seen in Section 4, even in unsupervised learning, it is still important to obtain a sizable aligned datasets for a reasonable evaluation. To further harness the power

of neural machine translation, we are currently still bound by the limitation of relevant datasets available. Looking into the informal world, the amount of mathematics literature is actually quite abundant, but we still lack a way to combine those with the ITP knowhow we have at hand. We believe that this is the major issue to be solved in autoformalization of mathematics. It will be the long-term focus of our continual research effort. At the current stage, we are exploring the possibility of joint embedding of multiple proof assistant libraries to see for opportunity to increase the size of our formal data.

Acknowledgement

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[2] Anonymous. 1994. The QED Manifesto. In *Proceedings of the 12th International Conference on Automated Deduction (CADE-12)*. Springer-Verlag, London, UK, UK, 238–251. http://dl.acm.org/citation.cfm?id=648231.752823

[3] Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised Neural Machine Translation. *CoRR* abs/1710.11041 (2017). https://doi.org/10.18653/v1/D18-1399 arXiv:1710.11041

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1409.0473

[5] Grzegorz Bancerek. 2006. Automatic Translation in Formalized Mathematics. *Mechanized Mathematics and Its Applications* 5, 2 (2006), 19–31.

[6] Grzegorz Bancerek, Czeslaw Bylinski, Adam Grabowski, Artur Kornilowicz, Roman Matuszewski, Adam Naumowicz, and Karol Pak. 2018. The Role of the Mizar Mathematical Library for Interactive Proof Development in Mizar. *J. Autom. Reasoning* 61, 1-4 (2018), 9–32. https://doi.org/10.1007/s10817-017-9440-6

[7] Grzegorz Bancerek and Patricia Carlson. 1994. Mizar and the Machine Translation of Mathematics Documents. *Available online at http://www.mizar.org/project/banc_carl93.ps* (1994).

[8] Grzegorz Bancerek, Adam Naumowicz, and Josef Urban. 2018. System Description: XSL-Based Translator of Mizar to LaTeX. In *Intelligent Computer Mathematics - 11th International Conference, CICM 2018, Hagenberg, Austria, August 13-17, 2018, Proceedings (Lecture Notes in*

Computer Science), Florian Rabe, William M. Farmer, Grant O. Passmore, and Abdou Youssef (Eds.), Vol. 11006. Springer, 1–6. https://doi.org/10.1007/978-3-319-96812-4_1

[9] Jasmin Christian Blanchette, Max W. Haslbeck, Daniel Matichuk, and Tobias Nipkow. 2015. Mining the Archive of Formal Proofs. In *Intelligent Computer Mathematics - International Conference, CICM 2015, Washington, DC, USA, July 13-17, 2015, Proceedings (Lecture Notes in Computer Science)*, Manfred Kerber, Jacques Carette, Cezary Kaliszyk, Florian Rabe, and Volker Sorge (Eds.), Vol. 9150. Springer, 3–17. https://doi.org/10.1007/978-3-319-20615-8_1

[10] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606* (2016).

[11] Chad E. Brown. 2012. Satallax: An Automatic Higher-Order Prover. In *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings (Lecture Notes in Computer Science)*, Bernhard Gramlich, Dale Miller, and Uli Sattler (Eds.), Vol. 7364. Springer, 111–117. https://doi.org/10.1007/978-3-642-31365-3_11

[12] Chad E. Brown and Josef Urban. 2016. Extracting Higher-Order Goals from the Mizar Mathematical Library. In *Intelligent Computer Mathematics - 9th International Conference, CICM 2016, Bialystok, Poland, July 25-29, 2016, Proceedings (Lecture Notes in Computer Science)*, Michael Kohlhase, Moa Johansson, Bruce R. Miller, Leonardo de Moura, and Frank Wm. Tompa (Eds.), Vol. 9791. Springer, 99–114. https://doi.org/10.1007/978-3-319-42547-4_8

[13] Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, Iryna Gurevych and Yusuke Miyao (Eds.). Association for Computational Linguistics, 76–86. https://doi.org/10.18653/v1/P18-1008

[14] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR* abs/1412.3555 (2014).

[15] Alonzo Church. 1940. A Formulation of the Simple Theory of Types. *The Journal of Symbolic Logic* 5 (1940), 56–68.

[16] Marcos Cramer, Bernhard Fisseni, Peter Koepke, Daniel Kühlwein, Bernhard Schröder, and Jip Veldman. 2010. The Naproche Project Controlled Natural Language Proof Checking of Mathematical Texts. In *Controlled Natural Language*, Norbert E. Fuchs (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 170–186.

[17] George Cybenko. 1989. Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals and Systems* 2, 4 (01 Dec 1989), 303–314. https://doi.org/10.1007/BF02551274

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[19] Simon S. Du, Jason D. Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. 2018. Gradient Descent Finds Global Minima of Deep Neural Networks. *CoRR* abs/1811.03804 (2018).

[20] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *Proc. of ICML*.

[21] M. J. C. Gordon and T. F. Melham (Eds.). 1993. *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic.* Cambridge University Press, New York, NY, USA.

[22] Thomas Hales, Mark Adams, Gertrud Bauer, Tat Dat Dang, John Harrison, Le Truong Hoang, Cezary Kaliszyk, Victor Magron, Sean Mclaughlin, Tat Thang Nguyen, Quang Truong Nguyen, Tobias Nipkow, Steven

Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, Thi Hoai An Ta, Nam Trung Tran, Thi Diep Trieu, Josef Urban, Ky Vu, and Roland Zumkeller. 2017. A Formal Proof of the Kepler conjecture. *Forum of Mathematics, Pi* 5 (2017). https://doi.org/10.1017/fmp.2017.1

[23] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[24] Kurt Hornik. 1991. Approximation Capabilities of Multilayer Feedforward Networks. *Neural Netw.* 4, 2 (March 1991), 251–257. https://doi.org/10.1016/0893-6080(91)90009-T

[25] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer Feedforward Networks Are Universal Approximators. *Neural Netw.* 2, 5 (July 1989), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8

[26] Cezary Kaliszyk, Josef Urban, and Jiří Vyskočil. 2015. Learning To Parse on Aligned Corpora (Rough Diamond). In *Proc. 6h Conference on Interactive Theorem Proving (ITP'15) (LNCS)*, Christian Urban and Xingyuan Zhang (Eds.), Vol. 9236. Springer-Verlag, 227–233. https://doi.org/10.1007/978-3-319-22102-1_15

[27] Cezary Kaliszyk, Josef Urban, and Jiří Vyskočil. 2017. Automating Formalization by Statistical and Semantic Parsing of Mathematics. In *8th International Conference on Interactive Theorem Proving (ITP 2017) (Lecture Notes in Computer Science)*, Mauricio Ayala-Rincón and César A. Muñoz (Eds.), Vol. 10499. Springer, 12–27. https://doi.org/10.1007/978-3-319-66107-0_2

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1097–1105. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[29] Guillaume Lample and Alexis Conneau. 2019. Cross-lingual Language Model Pretraining. *arXiv preprint arXiv:1901.07291* (2019).

[30] Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-Based & Neural Unsupervised Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[31] Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. 2017. Neural Machine Translation (seq2seq) Tutorial. *https://github.com/tensorflow/nmt* (2017).

[32] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton (Eds.). ACL, 1412–1421.

[33] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 3111–3119.

[34] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, 311–318. https://doi.org/10.3115/1073083.1073135

[35] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga,

and Adam Lerer. 2017. Automatic Differentiation in PyTorch. In *NIPS Autodiff Workshop*.

[36] Bartosz Piotrowski and Josef Urban. 2019. Guiding Theorem Proving by Recurrent Neural Networks. *CoRR* abs/1905.07961 (2019). arXiv:1905.07961 http://arxiv.org/abs/1905.07961

[37] Itay Safran and Ohad Shamir. 2016. Depth Separation in ReLU Networks for Approximating Smooth Non-Linear Functions. *ArXiv* abs/1610.09887 (2016).

[38] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* 529 (2016), 484–503. http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html

[39] Geoff Sutcliffe, Christian B. Suttner, and Theodor Yemenis. 1994. The TPTP Problem Library. In *Automated Deduction - CADE-12, 12th International Conference on Automated Deduction, Nancy, France, June 26 - July 1, 1994, Proceedings (Lecture Notes in Computer Science)*, Alan Bundy (Ed.), Vol. 814. Springer, 252–266. https://doi.org/10.1007/3-540-58156-1_18

[40] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. *CoRR* abs/1409.3215 (2014).

[41] Matus Telgarsky. 2015. Representation Benefits of Deep Feedforward Networks. *CoRR* abs/1509.08101 (2015).

[42] Andrzej Trybulec. 1989. Tarski Grothendieck Set Theory. *Journal of Formalized Mathematics, Axiomatics* 1 (1989).

[43] Josef Urban. 2004. MPTP - Motivation, Implementation, First Experiments. *J. Autom. Reasoning* 33, 3-4 (2004), 319–339. https://doi.org/10.1007/s10817-004-6245-1

[44] Josef Urban. 2005. XML-izing Mizar: Making Semantic Processing and Presentation of MML Easy. In *MKM (Lecture Notes in Computer Science)*, Vol. 3863. Springer, 346–360. https://doi.org/10.1007/11618027_23

[45] Josef Urban. 2006. MoMM - Fast Interreduction and Retrieval in Large Libraries of Formalized Mathematics. *Int. J. on Artificial Intelligence Tools* 15, 1 (2006), 109–130. http://ktiml.mff.cuni.cz/~urban/MoMM/momm.ps

[46] Josef Urban. 2006. MPTP 0.2: Design, Implementation, and Initial Experiments. *J. Autom. Reasoning* 37, 1-2 (2006), 21–43. https://doi.org/10.1007/s10817-006-9032-3

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 5998–6008. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[48] Hao Wang. 1954. The Formalization of Mathematics. *The Journal of Symbolic Logic* 19, 4 (1954), 241–266. http://www.jstor.org/stable/2267732

[49] Qingxiang Wang, Cezary Kaliszyk, and Josef Urban. 2018. First Experiments with Neural Translation of Informal to Formal Mathematics. In *11th International Conference on Intelligent Computer Mathematics (CICM 2018) (LNCS)*, Florian Rabe, William M. Farmer, Grant O. Passmore, and Abdou Youssef (Eds.), Vol. 11006. Springer, 255–270. https://doi.org/10.1007/978-3-319-96812-4_22

[50] Claus Werner Zinn. 2004. *Understanding informal mathematical discourse*. Ph.D. Dissertation. University of Erlangen-Nuremberg.