Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Improving Performance of Natural Language Inference Models for Numerical Queries

Bachelor's Thesis of
Viktor Korladinov

Supervisor: **Ing. Jan Drchal, Ph.D.**
Study program: **Open Informatics**
Specialization: **Artificial Intelligence and Computer Science**
May 2023

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Korladinov Viktor Emil**
Personal ID number:	**498844**
Faculty / Institute:	**Faculty of Electrical Engineering**
Department / Institute:	**Department of Cybernetics**
Study program:	**Open Informatics**
Specialisation:	**Artificial Intelligence and Computer Science**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Improving Performance of Natural Language Inference Models for Numerical Queries

Bachelor’s thesis title in Czech:

Zkvalitňování modelů Natural Language Inference pro numerické dotazy

Guidelines:

Experiment with NLI model pre-training based on numerically oriented datasets.
1) Review state-of-the-art methods for Natural Language Inference as well as NLP methods to solve numerically oriented tasks (e.g., word problems).
2) Collect available datasets. Use machine translation methods to unify data language.
3) Perform experiments on public English NLI data and/or other data supplied by the supervisor focusing on precision when solving numerically oriented queries.

Bibliography / sources:

[1] Ronald, Krist. Metody zpracování přirozeného jazyka pro řešení slovních úloh. MS thesis. České vysoké učení technické v Praze. Vypočetní a informační centrum., 2022.
[2] Storks, Shane, Qiaozi Gao, and Joyce Y. Chai. "Recent advances in natural language inference: A survey of benchmarks, resources, and approaches." arXiv preprint arXiv:1904.01172 (2019).
[3] Ullrich, Herbert, et al. "CsFEVER and CTKFacts: Acquiring Czech data for fact verification." arXiv e-prints (2022): arXiv-2201.

Name and workplace of bachelor’s thesis supervisor:

Ing. Jan Drchal, Ph.D. Artificial Intelligence Center FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **31.01.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

Ing. Jan Drchal, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt — Student’s signature

Acknowledgements

I'd like to thank my supervisor, Ing. Jan Drchal, Ph.D., for his tremendous support during the writing of this thesis. I also extend my gratitude to my sister, for her eagerness to help and give feedback during various stages of the writing process.

The access to the computational infrastructure of the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics" is also gratefully acknowledged.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, 26. May 2023

Abstract

This thesis examines the numeracy and quantitative reasoning abilities of state-of-the-art models. It focuses on math word problems and models trained explicitly to solve them. Apart from solvers, Large Language Models are also benchmarked, as their multi-faceted nature creates expectations of proficiency in the field. Attention is also cast on the available datasets & their impact. The Ape210K dataset is the primary observation target, and this thesis proposes a model that filters out problems deemed unfit for training solvers and QR reasoners.

Keywords: word problems, natural language processing, quantitative reasoning

Supervisor: Ing. Jan Drchal, Ph.D.

Abstrakt

Tato práce zkoumá numerické a kvantitativní uvažování u nejmodernějších modelů. Zaměřuje se především na matematické slovní úlohy a modely přímo vycvičené k jejich řešení. Kromě těchto modelů jsou zde srovnávány také Large Language Models, u kterých lze předpokládat zběhlost v této oblasti vzhledem k jejich povaze. Pozornost je dále věnována dostupným datasetům a jejich vlivu. Největší část je věnována Ape210K datasetu — tato práce navrhuje model filtrující problémy, které jsou považované za nevhodné pro trénování výše zmíněných modelů.

Klíčová slova: slovní úlohy, zpracování přirozeného jazyka, kvantitativní uvažování

Contents

Figures

Tables

Chapter 1
Introduction

Numeracy and quantitative reasoning are central to everyday life because logical, critical thinking is only achievable with proficient numerical literacy. Muzaini et al., 2019 came to this conclusion in a case study exploring students' QR (quantitative reasoning) abilities.

With the steady improvement and refinement of computer hardware, machines have become much faster than humans at performing computations. However, that does not mean computers are better quantitative reasoners. Machines become useful once they have received complete, detailed instructions on the computations needed — essentially once the logical thinking has been performed for them.

Computers would benefit significantly from software that would enable them to solve less detailed tasks, as it would vastly expand the range of what they can calculate. Researchers have focused their efforts on creating artificial intelligence capable of finding the correct steps to a solution without outside help, eliminating the need for those aforementioned precise instructions.

This thesis aims to explore, review and improve the quantitative reasoning abilities of those models in a meaningful way. There are different ways to look at and benchmark the QR aptness of current SOTA (state-of-the-art) models; In this thesis, the approach opted for and the models being considered are focused on math word problems. This angle provides an easily verifiable result and showcases the decision process behind it because most models trained on solving MWP (math word problems) return not only a result but also the equation constructed to get to it. Figure 1.1 showcases a SOTA model's output and the degree to which one can peer into the process.

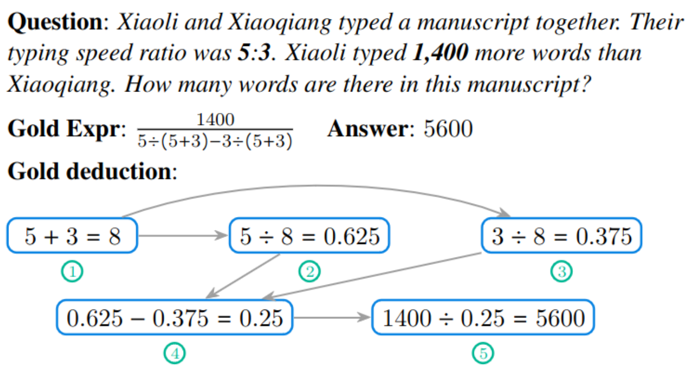

Figure 1.1: Calculation process of the DeductiveMWP model (Jie et al., 2022)

Instead of focusing solely on models specifically trained for solving MWP, it also makes sense to investigate the level of numeracy in LLMs (Large Language Models) (ChatGPT, GPT-4, etc.). That is because chatbots' main objective is to interact with humans, which in turn means they need versatile problem-solving skills.

1.1 Motivation

Solving MWPs exactly and reliably would have a bigger impact than it may seem. Those skills could then be translated into bigger models, where numerical aptness is crucial to succeeding in their tasks. For example, an area that would greatly benefit from robust numeracy mechanisms is fact-checking:

Claim: 800 more job opportunities will emerge in the district after the eighth of March 2023.

Fact: On March 1st, Company X will begin the hiring process for 300 new employees following a recent expansion announcement. Their rival, Y, announced shortly after they plan to expand their staff by 500 new members starting next week.

Processing correctly the example above needs proper QR in two different areas — first, to evaluate whether the overall amount of jobs told adheres to the facts and second, to assess whether the date stated is in accordance with the factual timeline of events.

Solving the entire problem of teaching numerical literacy at once would be very difficult, and because of that, it is fragmented into multiple steps. The first step is also the main focus of this thesis — an effective MWP solver. (Later steps could then use this model to train other, bigger multi-functional models.)

1.2 Related Work

The quantitative abilities of the two types of models are of interest. The first are the MWP solvers mentioned in the preceding section. The second type of models examined are LLMs because of their recent leap forward in all areas of language inference.

Math word problem solvers have become very proficient at their goal — the RobertaDeductiveReasoner (Jie et al., 2022) has a 92% solve rate[1] on one of the most popular datasets used for benchmarking and training — MAWPS (Koncel-Kedziorski et al., 2016). It treats the issue as a complex relation extraction problem, in contrast to previous models, such as the Zhang et al., 2020 Graph2Tree model, which has obtained an 83.7% accuracy[1] and the Xie et al., 2019 GTS (Seq2Tree), which has not been benchmarked on MAWPS but has achieved 74.3% on the Math23K dataset[2].

The graphs linked in the footnotes and models mentioned above demonstrate the notable accuracy boost achieved by developing models with attention mechanisms and other tools aimed at helping the algorithm consider the problem as a whole instead of relying on a sentence-by-sentence approach to solve it.

[1] https://paperswithcode.com/sota/math-word-problem-solving-on-mawps

[2] https://paperswithcode.com/sota/math-word-problem-solving-on-math23k

Aside from the models specifically trained for solving MWP, the dominant players in the field have been LLMs — Large Language Models, such as ChatGPT[3] (Liu et al., 2023), Llama (Touvron et al., 2023), and others. Because of the enormous scale of the data used for training, these models are quite effective at solving some MWPs. Their shortcomings are examined in section 4.

[3] https://chat.openai.com/

Chapter 2
State-of-the-art

The state-of-the-art label is given to models that are currently on top of various benchmarks' leaderboards or close to the top.

2.1 Transformers and their significance

All models currently considered state-of-the-art are based on the transformer architecture introduced by Google in the paper Attention is all you need (Vaswani et al., 2017).

2.1.1 Architecture

The architecture that served as a stepping stone to transformers is called Encoder Decoder. Its premise is that mapping input directly to output is not always trivial, and so a mediating state should be introduced in between them to help with transforming the output. An example of an area where this is very beneficial is translation — often, sentences in the desired language are of a different length than the initial, plus many times there is a need for restructuring and reordering.

The encoder encodes the received data, which means it modifies the given input to create a numerical representation that contains the information transmitted. In translation context, that numerical representation is the state between the sentence in the original language and the sentence in the desired secondary language. The decoder takes that state and decodes it to the desired output.

Since the Encoder-Decoder architecture precedes transformers, initially, different models were used inside it — mostly CNN and RNN (a model using RNN is DNS, reviewed in subsection 2.3.1). However, using them raises the complexity quite significantly and doesn't provide a way to consider the input as a whole but rather encodes/decodes it piece by piece.

Transformers shed away most of the complexity and also process the input at once, attributing different weights to different parts. Because of that, it is also much easier to parallelize, making it a great choice for models needed to be trained on colossal corpora.

Another great quality of transformers is their transferability — transformers can be "taught new tricks" by tuning them on smaller datasets — meaning that a pre-trained model (such as the BERT family and other models trained on huge corpora) can be taught to do a specific task very well with a much smaller dataset. This ability of theirs is called transfer learning and is described below (Transfer learning, 2.1.2).

The combination of these qualities is the reason transformers currently dominate all NLP areas, including the ones discussed in this paper.

2.1.2 Transfer learning

As stated, transformers are the go-to architecture for building efficient NLP models. One of the reasons for this is the fact that they are capable of transferring already learned skills. The best way to leverage that ability of theirs is to divide the training into multiple steps. The process is separated into task-specific learning and general language and syntax learning — the latter is called pre-training and does not need to be taught for every single model separately, which greatly reduces the computational complexity.

Pre-training is a form of training done on a vast dataset with a language modeling objective (T. Wang et al., 2022). The language modeling objective involves training the model to predict the next word in a sentence, given the previous words as context. It is often unsupervised[1] — the bigger the model and the number of parameters, the better it fares with general data. Sifting through enormous amounts of data teaches the model language structure, syntax, and other universal skills. This part of the training process is not done for every new model as it is very computationally expensive; instead, a few big pre-trained models (reviewed in 2.5) are used.

The next stage is unique for every model and is where transfer learning comes into play: it allows for honing the pre-trained model's ability to perform a particular downstream task (smaller task in the general domain of the model's aptness) very well with minimal data. This phase is called downstream adaptation.

It consists of the following:

1. Fine-tuning

   Frozen / Unfrozen layers: Freezing layers of the network ensures the parameters of that layer are unaffected by additional training. Usually, many of the deeper layers are frozen during fine-tuning, and just the top few (containing the least general knowledge) are unfrozen. Unfreezing deeper allows for stronger fine-tuning. (Howard et al., 2018)

2. (Optional) Task-specific modifications

   Often, the model has to be modified to be able to adapt to the task — frequently, the models are fitted with task-specific layers such as classification or regression heads on top of the pre-trained layers.

2.2 Sections

Quantitative reasoning and numeracy are very important skills for all Natural Language Inference models. The application of those skills, however, varies depending on the model's needs and its aim. For that reason, the models below are grouped into sections depending on the way they need and use quantitative reasoning:

[1] https://huggingface.co/tasks/zero-shot-classification

- Math problem solving
  - *Q: Joan found 70.0 seashells on the beach. She gave Sam some of her seashells. She has 27.0 seashells. How many seashells did she give to Sam? A: 43.0* (Koncel-Kedziorski et al., 2016)
- Textual entailment (Fact-checking)
  - *Premise: Sam had 9.0 dimes in his bank, and his dad gave him 7.0 dimes.*
    Hypothesis: Sam has 16.0 dimes (Ravichander et al., 2019)
  - *Fact checkers often rely on numeracy — for example, if a week-old article states that in two weeks, event A will happen, then the hypothesis "Event A is a week away" is correct.*
- Chatbots and large multimodal models
  - *Chatbots mimic humans' conversation skills and thus use quantitative reasoning in numerous tasks, including the ones mentioned above. They use it to understand the other interlocutor, but it is also often needed in order to compute the reply correctly.*
  - *LMMs (large multimodal models) such as the GPT family strive to match humans' performance in a plethora of different tasks with one model.*

2.3 Math problem solvers

These models are designed specifically to solve math problems. The expected result of these models is either just a number or an equation — equations are helpful because they provide insight into the solving process and also help generalize the problem (having the correct equation negates any issues that might arise from swapping some of the numbers. Correct result can be achieved by just swapping them in the equation as well.)

There are different approaches to solving an MWP. First, there are seq2seq models like the Deep Neural Solver (Y. Wang et al., 2017). These algorithms, however, work sequentially and miss the mark of how MWPs are designed to be solved. The issue is that inserting tokens sequentially and generating output based on that doesn't allow a view of the bigger picture; In many areas of NLP, that is unneeded, and those algorithms work very well, but MWPs expect the solver to consider all the available information before attempting to form a solution. (Teachers often order students to read the problems aloud before the class is allowed to solve them, which is an indication of the importance of processing the whole input and then forming equations.) Also, the question part of the problem is usually the last sentence. Thus, it is impossible to generate anything before processing that first, at least.

Seq2tree algorithms eliminate some of those problems by initially identifying what needs to be calculated and then recursively splitting the problem into smaller chunks, creating a tree. Once the chunk is simple enough, it is calculated, and the result is propagated upwards(Xie et al., 2019). Some other approaches, although less often used, yield good results as well, and so they are mentioned in the respective subsection.

2.3.1 Deep Neural Solver

This model is one of the most accurate models that does not use transformers — it instead leverages RNNs and achieves an accuracy of 64.7%[2] on the Math23K dataset. It's the most accurate seq2seq solver.

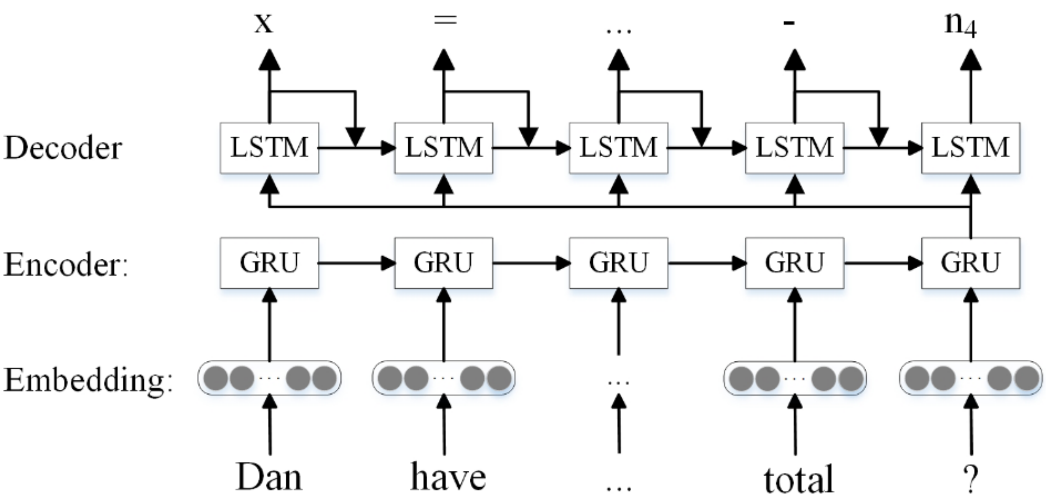

Figure 2.1: Showcase of sequentiality of the DNS (Y. Wang et al., 2017)

2.3.2 GTS

GTS is a shorthand for Goal-Driven Tree-Structured Neural Model for Math Word Problems (Xie et al., 2019). Its main advantage over previous competitors is already mentioned in the title — instead of analyzing and creating a solution sentence-by-sentence, the model creates a tree by finding the goal of the problem and splitting it recursively into smaller chunks until the chunks are simple enough to be easily computed. Figure 2.2 is an example of how an MWP would be solved using that technique. This seq2tree model has an accuracy of 74.3%[2] on the Math23K dataset.

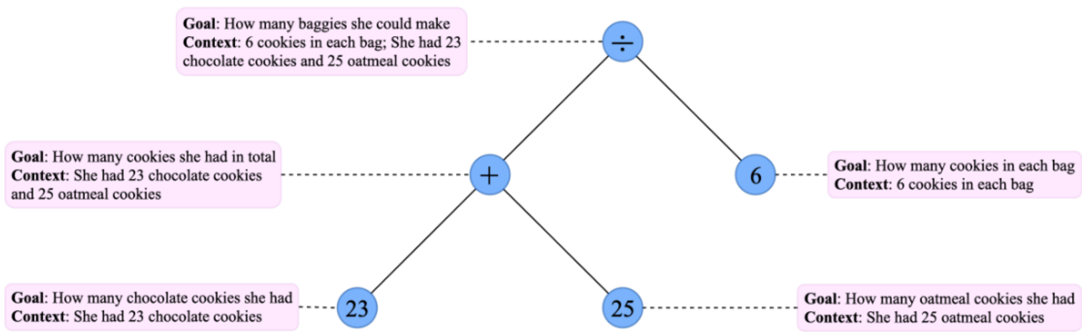

Figure 2.2: GTS's solution tree for the problem: "Robin was making baggies of cookies with 6 cookies in each bag. If she had 23 chocolate cookies and 25 oatmeal cookies, how many baggies could she make?" (Xie et al., 2019)

2.3.3 Graph2Tree

The next marginal improvement over preceding models was achieved by solving another major problem of these solvers — extracting and understanding relations. Solving word problems relies on correctly identifying the underlying relations between the given numbers. To feed that information correctly to the encoder, this solver processes the word problem and creates two graphs from it, which are then fed to the encoder: Quantity Cell Graph and Quantity Comparison Graph. Each node in the first graph ties one specific numerical value with one or more descriptive words from the problem, basically storing what that numerical value represents. (Example: 23 | chocolate, cookies). The other graph stores relationships between those different nodes.

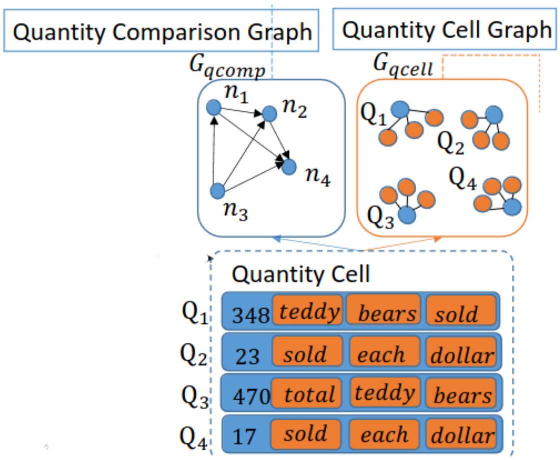

Figure 2.3: Graph2Tree Graphs Example

2.3.4 DeductiveMWP

The Roberta Deductive Reasoner (often labeled DeductiveMWP) is currently the most accurate MWP solver[1]. It expands on the idea of Graph2Tree and considers mapping precisely and correctly the relations between the different entities as an even more vital part of solving a mathematical problem. Furthermore, Jie et al., 2022 states that the models developed up until now generate linear equations or trees, which, although empirically adequate, do not provide enough logical explanation of the process. Thus, one of this reasoner's goals is to output a more transparent and concise solution that is easier to follow and verify. In order to accomplish that goal, the selected output form is not a tree nor a singular linear equation; rather, it is a sequence of simple equations that use previous results to arrive at the intended goal gradually. Figure 1.1 showcases an example pair of a problem and the reasoner's output.

2.4 Textual entailment

Textual entailment, also regarded as NLI (Natural Language Inference), is a discipline that aims to discern whether a hypothesis derived from a given premise is true, false, or unrelated. Consider the simple example:

P: Minerva had Starbucks yesterday — a large latte.

H: Minerva drank coffee yesterday.

The hypothesis *"Minerva drank coffee yesterday"* is entailed by the premise, and thus the correct label is *Entailment*. Label *Contradiction* would be accurate for a sentence such as *"H: Minerva did not drink anything yesterday"* because she drank coffee. The third option, *Neutral*, is reserved for hypotheses whose validity cannot be inferred from the premise: *"H: Minerva had tea yesterday"*. NLI often has to resort to numerical literacy. The example of TE (textual entailment) in 2.2 showcases how important QR is for determining factual correctness.

There are a few big datasets created for training NLI models — The Stanford Natural Language Inference (SNLI) Corpus (Bowman et al., 2015), the Multi-Genre Natural Language Inference (Kim et al., 2018), and others. Although QR is integrated into these datasets to an extent, its importance is mainly overlooked; an exception to the rule is the dataset EQUATE (Ravichander et al., 2019), whose aim is to teach models a higher degree of numeracy.

The model proposed in the paper, along with the published dataset, is called Q-REAS. Q-REAS scores much higher than models like GPT (98.1% versus 51.18%, Ravichander et al., 2019) in tests specifically designed to discern whether a model relies on QR or verbal and textual cues to guess the correct label. Q-REAS, however, performs worse than those LLMs when it comes to advanced verbal reasoning — because of that, this reasoner should be considered as a specialized tool that can and should be used parallel to a more multi-faceted model.

Chapter 3 offers a deeper dive into the key differences between EQUATE and other NLI datasets and the reasons behind its significantly higher need for proper QR.

2.5 Chatbots & Large multimodal models

Recently LMMs have become very efficient at their goal. The strongest competitors are the model families developed by OpenAI, Google, and Facebook — GPT, PaLM (Chowdhery et al., 2022), and LLaMA (Touvron et al., 2023), respectively. The fact that these corporations are in the lead should come as no surprise, as due to the nature of the models, the hardware needed for training them could only be supplied by the enormous infrastructures these companies already have set up.

These models are pre-trained on massive quantities of data and serve as a base for models with specific aims. Because of their transformer architecture (reviewed in section 2.1), these LMMs are designed to be adapted to a particular task with additional, much smaller datasets. There are different ways of downstream adaptation, but all utilize transfer learning[2].

Chatbots based on these models exhibit impressive verbal cue understanding, reasoning, and conversational abilities. Quantitative reasoning, however, along with numeracy and detection of complex relations, is not always up to par — deeper insight into their limitations in that department is provided in experiment 4.3.

[2]https://towardsdatascience.com/self-supervised-transformer-models-bert-gpt3-mum-and-paml-2b5e29ea0c26

Chapter 3
Data

One of the biggest problems that MWP solvers have to tackle is the scarcity of data. The main issue is that word problems alone are insufficient for a solver to be trained. The models tasked with finding a solution to a problem must return not only a number but also a generated response in the form of one or more equations. To do that, these models need to be trained with datasets whose labels, apart from the final result, contain the correct equation that would be used to get to that result. Hence, creating those datasets is manual and time-consuming, which has led to the aforementioned undersupply. Regardless, there are multiple arguments defending the decision to expect solvers to provide equations along with the numerical result:

- "Chain of thought"
  - *These equations are a window to the reasoning process and the steps taken to obtain that result. Almost all logical errors can be discovered by considering the validity of the equation returned.*
- Adaptability:
  - *Swapping a number with a similar one in a word problem presents no issue for a person that can already calculate that problem because the logic remains the same. However, that is not always the case with MWP solvers, especially if the new number is significantly higher than the previous one*[1]*. Equations should help alleviate that particular issue, as swapping it in the equation instead would be logically easier and, at the same time, a computationally cheaper variant.*
- Explainability:
  - *Ideally, QR should not only be used for arriving at the correct result but also for helping the user understand the problem and its solution: the more detailed the answer to the problem is, the easier it is to grasp. Some of the models detailed in section 2.3 even go beyond providing a single equation and opt for a more detailed approach.*[2]
- Reusability:
  - *Some models fragment complex problems into smaller chunks. Not all of these fragments have been solved for every problem if they are similar enough to a previously solved one.*

[1] This issue is explored in experiment 4.3.

[2] See RobertaDeductiveReasoner, subsection 2.3.4

In contrast, receiving just a numerical value offers no insight into the process, nor does it simplify further operation if one is needed.

3.1 Input and output

Different models preprocess the data differently, but the input for all remains the same: an MWP with varying difficulty, but mostly aimed at younger students.

A sample word problem consists of the following:

- Word problem: *Mary is baking a cake. The recipe calls for 12 cups of flour and 5 cups of sugar. She had already put in some cups of flour. If she still needs 2 more cups of flour*
- Question: *How many cups of flour did she put in?*
- Equation: (12.0 - 2.0)
- Answer: 10.0
- Type: Subtraction

This particular word problem is taken from the SVAMP (Patel et al., 2021) dataset (chal-41), described in subsection 3.2.4.

The output should be similar to what "Equation" and "Answer" contain — an equation showcasing the process and result, holding the correct outcome of the equation. Depending on the difficulty, the equation may be more complex; Some models generate that equation, while others generate a sequence of simpler and more understandable ones.

Processing of the input depends on the model as well — as reviewed in section 2.3, different solvers employ different solutions, but all fixate on improving a few key areas:

- Avoiding shallow heuristics
  - *It has been proven some solvers often rely on shallow heuristics in order to calculate the correct result — the SVAMP paper (Patel et al., 2021) explores this topic and proposes ways to alter the data, to make it impossible or significantly harder to do so.*
- Extracting relations
  - *Correctly extracting the relations between numbers and what they represent is extremely vital in this area of NLI. Understanding and answering the problem's question can only be done if the extracted relations between entities are accurate. Different methods for extracting those relations, such as trees and graphs, are reviewed in 2.3.*

3.2 Datasets for Math Word Problem Solving

There are multiple MWP repositories. The most extensive datasets available are Math23K (Y. Wang et al., 2017) and Ape210K (Zhao et al., 2020), both marginally bigger than any other dataset. However, neither is in English. Language matters because of the previously mentioned use cases of such a model in section 1.1; Training an MWP solver in a particular language enables us to then utilize transfer learning (subsection 2.1.2) and teach those skills to a larger, more multi-faceted model. Table 3.1 displays different datasets, their current availability, language, and size. Subsections 3.2.1 to 3.2.7. examine each one individually.

Name	Size	Language	Availability
Ape210K	210,488	Chinese	Dataset only
Math23K	23,162	Chinese	Dataset & paper
MAWPS	2,373	English	Paper only[3]
SVAMP	1,000	English	Dataset & paper
ASDiv-A	2,305	English	Dataset & paper
Dolphin1878	1,878	English	Dataset & paper
Dolphin18K	18,460	English	Dataset & paper
Alg514	514	English	Dataset & paper

Table 3.1: List of the different MWP datasets

3.2.1 ASDiv-A

The Academia Sinica Diverse MWP Dataset (ASDiv) (Miao et al., 2020) is an English dataset containing around two thousand and three hundred word problems. Apart from body, question, equation, and answer, each word problem in this set also has information about the type of the problem, its source, and its difficulty (based on in which grade a student could encounter the problem).

3.2.2 MAWPS

MAWPS: a Math Word Problem Repository (Koncel-Kedziorski et al., 2016) is an online repository with similar attributes to ASDiv-A (subsection 3.2.1). Where it differs is in its ability to be expanded — the repository is hosted online[4], and the University of Washington has provided the necessary tools to host a copy locally and also the tools to modify and enlarge that copy. Note: Currently, the online repository is down; It is unclear whether the repository has been discontinued or the unavailability is temporary. For now, the MAWPS dataset can be accessed from the SVAMP (3.2.4) Git repository.

3.2.3 Math23K

The Math23K dataset contains more than twenty-three thousand word problems, complete with equations and results. It is the go-to benchmark[5] for comparing models' efficiency. Note that the pre-trained models competing were trained in Chinese.

[3] Dataset accessible through SVAMP Git repository

[4] `http://lang.ee.washington.edu/MAWPS/`

[5] `https://paperswithcode.com/sota/math-word-problem-solving-on-math23k`

3.2.4 SVAMP

SVAMP (Patel et al., 2021) is a smaller dataset containing one thousand-word problems. Its contribution is quite significant regardless because of the extensive examination the paper puts both MAWPS and ASDiv-A under. It argues that neither dataset is suited to be a reliable benchmark; The issue stems from the way the problems are formulated. In order to reach that conclusion, several experiments are detailed and performed on the SOTA models that currently hold the highest accuracy scores on those datasets.

One of those tests is removing the question from the problem altogether — an actual QR model would fail, just like a person would because nothing is being asked. That is not the case with the leaderboard leaders, showing they utilize shallow heuristics (mentioned in 3.1) instead of true numeracy. The types of experiments done in the paper are shown using an example in figure 3.1.

PROBLEM:
Text: Jack had 8 pens and Mary had 5 pens. Jack gave 3 pens to Mary. How many pens does Jack have now?
Equation: 8 - 3 = 5

QUESTION SENSITIVITY VARIATION:
Text: Jack had 8 pens and Mary had 5 pens. Jack gave 3 pens to Mary. How many pens does Mary have now?
Equation: 5 + 3 = 8

REASONING ABILITY VARIATION:
Text: Jack had 8 pens and Mary had 5 pens. Mary gave 3 pens to Jack. How many pens does Jack have now?
Equation: 8 + 3 = 11

STRUCTURAL INVARIANCE VARIATION:
Text: Jack gave 3 pens to Mary. If Jack had 8 pens and Mary had 5 pens initially, how many pens does Jack have now?
Equation: 8 - 3 = 5

Figure 3.1: Example of a Math Word Problem along with the types of variations that are created for SVAMP (Patel et al., 2021)

3.2.5 Dolphin datasets

Dolphin1878 (Shi et al., 2015) and Dolphin18K (Huang et al., 2016) are datasets created by scraping the internet. The web pages most of the problems come from are Yahoo! answers and Algebra.com. Due to the method of collecting these problems, the quality is slightly lower than what the manually created sets offer. The dataset is already deployed with tools that help clean and streamline the data. Table 3.2 showcases an example of a Dolphin problem so that a comparison can be drawn.

Answer	Equation	Index	Sources	Body
9	unknown: x equation: x+2*(x+1)=29	191	algebra.com.306431	For two consecutive integers, the sum of the smaller and twice the larger is 29. Find the smaller integer

Table 3.2: Dolphin example word problem

3.2.6 Alg514

Alg514 (Kushman et al., 2014) is a dataset that, similarly to Dolphin, contains word problems from Algebra.com. The difference is that the problems in this set were selected, checked, and cleaned manually. It contains 514 problems overall.

3.2.7 Ape210K

Ape210K (Zhao et al., 2020) is the most extensive MWP set. It contains a much wider range of word problems and templates. Using this dataset to learn would drastically improve an MWP solver's capabilities, thanks to the sheer volume of training material. However, a few critical problems outlined below are stopping it from becoming the norm for MWP training. The majority of these issues are not unique to this set but are amplified because of the vastness of the data. They include but are not limited to:

- Language
  - *The dataset is in Chinese. As mentioned previously, one of the most practical applications of this technology would come from transferring the model's skills to a bigger model for a quantitative reasoning and critical thinking boost. That would still be possible, but it would have to be paired with a Chinese model, which is off the target of this paper.*
- Availability
  - *There is a certain unresolved issue with the availability of this dataset. The official paper on arXiv.org*[6] *has been withdrawn. The reason, as stated by the authors, is that the dataset is not going public. Thus the results and experiments detailed in the paper are irreproducible and meaningless*[7]*. The dataset was soon after published, but the paper remained withdrawn.*
- External constants
  - *External constants (EC) & knowledge (EK) were not mentioned up to this point because of their negligible impact on the previous datasets. External knowledge is a general label for all knowledge that is vital for obtaining a solution but is not contained in the body itself. Consider the following problem:*

    The volume of a cylinder is $36.15m^3$; what is the volume of a cone with equal height and equal bottom surface area?

    In this instance, there is both EC and EK:. The model needs to know how to calculate the volume of a cylinder, yet that information is not presented in the problem. Also, it would need to know the values of external constants like π.

Experiments 4.4 and 4.5 try to mitigate some of these issues and process & alter the data in a way that would allow models to train from it.

[6] https://arxiv.org/abs/2009.11506

[7] See: Comments section of the web page linked in footnote 6.

3.3 Datasets for Entailment & LLMs

Textual entailment, as reviewed in chapter 2.4, is a prevalent topic, and many different and large datasets are used to train and evaluate Natural Language Inference. Some of these include The Stanford Natural Language Inference (SNLI) Corpus (Bowman et al., 2015) complete with a leaderboard and accuracy scores[8] for models, and The Multi-Genre Natural Language Inference (MultiNLI) corpus (Williams et al., 2018), crowd-sourced thanks to the initiative of NYU, both vast, diverse and annotated.

The benchmark evaluation framework EQUATE Ravichander et al., 2019 uncovered a problem — by testing the SOTA NLI models on the dataset curated precisely for the purpose of evaluating numeracy aptness, a conclusion was reached that none of them are capable of quantitative reasoning; instead, very similar to past MWP solvers, the models rely on lexical cues to form the label. The set is compiled from five sources, three extracted from real-world data and two synthetic. Figure 3.2 showcases those different sources and an example from each source.

RTE-QUANT	
P:	After the deal closes, Teva will generate sales of about $ 7 billion a year, the company said.
H:	Teva earns $ 7 billion a year.
AWP-NLI	
P:	Each of farmer Cunningham's 6048 lambs is either black or white and there are 193 white ones.
H:	5855 of Farmer Cunningham's lambs are black.
NEWSNLI	
P:	Emmanuel Miller, 16, and Zachary Watson, 17, are charged as adults, police said.
H:	Two teen suspects charged as adults.
REDDITNLI	
P:	Oxfam says richest one percent to own more than rest by 2016.
H:	Richest 1% To Own More Than Half Worlds Wealth By 2016 Oxfam.

Figure 3.2: EQUATE dataset examples (Ravichander et al., 2019)

As mentioned in chapter 2.5, LLMs are trained with enormous amounts of data. The datasets for these models are essentially big chunks of the internet, so evaluating their QR during training is quite complex. It is considerably more straightforward to evaluate their quantitative reasoning once they are published. Experiment 4.3 investigates the numerical aptitude of the current SOTA LLM — ChatGPT.

[8] https://nlp.stanford.edu/projects/snli/

Chapter 4
Data analysis & Experiments

4.1 Motivation for experiments

The aim of this paper is to review and attempt to improve the quantitative reasoning abilities of state-of-the-art Natural Language Inference models. In line with that, the experiments I performed and that I have outlined below fall into one of these categories:

1. Testing numeracy & critical thinking on models that do not delve into the issue within the research paper published about them

2. Modifying, altering and cleaning existing datasets to enhance the usability and quality of the sets

3. Utilizing existing and/or creating new tools to achieve better accuracy results of current SOTA models

My focus is once again on the models capable of solving math word problems — be they dedicated solvers or big, multi-purpose ones.

4.2 Sections & Categories

As I already mentioned, the experiments I have performed either attempt to collect more data from models that are not open-sourced through empirical testing or to augment existing datasets for better potential utilization.

The first experiment is in a separate category, as it deals with LLMs — more specifically, ChatGPT. OpenAI provides very little insight into the numeracy capabilities of their model in the publishing article[1]. The Summary of ChatGPT Research (Liu et al., 2023) explores ChatGPT's ability to handle mathematics & MWPs, but it uses a dataset automatically collected and does not detail the model's weak spots. Because of that, I've decided to examine its quantitative reasoning in this paper, in section 4.3.

Experiment 4.4 and 4.5 deal with data augmentation; Note that I have created a special tool to aid me in further annotating the data I have processed from the existing datasets — more information on it is available in the documentation of experiment 4.5 (section Tools & Execution, 4.5.2).

[1] `https://openai.com/blog/chatgpt`

4.3 Experiment #1: Large Language Models' Quantitative Reasoning

As already discussed, apart from considering (trained explicitly for the task) MWP solvers, it pays to examine LLMs' capabilities — the idea is that they are trained on data, so vast, numerical, and logical problems should be easier to solve.

4.3.1 Experiment #1: Implementation Choices

I decided to test a chatbot implementation of an LLM. The main reason behind that decision was that they are the easiest to access for the average user, meaning they are most likely to encounter issues that rely on critical thinking (very common during unscripted conversations with humans, which is the main objective of these models) to be resolved.

The chatbot I opted for is ChatGPT, as it was the newest and most capable chatbot accessible during the writing of this thesis.[2] The project is not open-sourced. Thus, the testing had to be done through one of the interfaces offered by the provider, OpenAI. Overall, the key things that had to be resolved before benchmarks could begin were as follows:

- Interface:
  - *OpenAI offers a paid API endpoint that provides a straightforward way of automation — unfortunately, the service is paid, so I chose to exhaust all other options before purchasing a commercial license.*
  - *The aforementioned API is targeted towards commercial consumers — OpenAI offers interaction with their model to the public & researchers through a webpage [3] that is somewhat limited [4]. There are measures against scraping/ automation of the interactions with this faucet, so some manual work is required.*
- Problem set
  - I reviewed the current datasets in chapter 3 — I had to make a decision, which dataset, or a mix of which datasets, to use while conducting the testing.
  - I considered multiple points during the selection:
    - Language — The tests being in English is quite important in this context, so that immediately ruled out the bigger datasets such as Math23K (subsec. 3.2.3) and Ape210K (subsec. 3.2.7).
    - Quality of problems — It is a good idea to limit the outside factors that could sway the results. Thus I chose to work with hand-picked and hand-cleaned datasets, so I had to eliminate the ones scraped from the internet & processed by bots (Dolphin sets, subsec. 3.2.5).
    - Diversity of problems — some sets, such as Alg514 (subsec. 3.2.6), offer too little variation for me to draw conclusive results.

[2] On May 4th Microsoft released their improved Bing search engine, empowered by GPT-4 (Mehdi, 2023), which is supposedly much more capable than the GPT-3.5 powering ChatGPT. Its release is too close to publishing this paper; thus, examining it will be delegated to future work on the subject.

[3] `https://chat.openai.com/`

[4] Limited in requests per hour, not capability. The performance of the publicly available model closely matches the performance of the paid version

I considered all the factors outlined above and concluded that the best initial course is to attempt to use the publicly available interface paired with the SVAMP data.

The reasons behind my selection of interface are simple — a relatively small dataset should suffice for what my goal is — thus, the manual work I'd have to do would not be of a magnitude I cannot handle; Purchasing a month-long license, I consider as a last resort because the interval I'd need it for is much shorter.

Choosing the SVAMP dataset was a straightforward process as well because of all the limiting factors detailed in the description of the issue — from the viable datasets, I chose to use SVAMP instead of MAWPS, as SVAMP was explicitly created to combat the shortcomings of the latter, the most important of which is its inability to differentiate between calculations based on shallow heuristics & lexical cues and computations performed with the help of true quantitative reasoning.

4.3.2 Experiment #1: Tools & Execution

The main obstacle I had to find a solution for in order to benchmark the model was accessing the interface as efficiently as possible. Because of the chosen route, that was a bit more complex, but I eventually found a library that helped transfer the publicly available web faucet to a CLI[5]. Unfortunately, this was patched by the OpenAI team during testing, and the tool was labeled deprecated.

To streamline the process of inputting the problems and extracting the results, I created a tool in Python (`chatGptParser.py` & `logParser.py`) that:

1. Prepares a problem:
   - The problems in the SVAMP dataset are already separated into *Body* and *Question* fields, simplifying it for solvers but creating an issue for models trained to work with more natural sounding requests/problems. Because of that, the problem is "glued back together":

     Body: Dan had 3$ left with him after he bought a candy bar. If he had 4$ at the start,

     Question: How much did the candy bar cost?

     becomes

     Problem: Dan had 3$ left with him after he bought a candy bar. If he had 4$ at the start, how much did the candy bar cost?
2. Automatically copies the "glued" problem to clipboard
3. Enters "receiving" state, that accepts lines until EOF is reached, which then saves in field *Answer*
4. Asks the user if the answer provided is *Correct* and saves in a field bearing the same name.
5. Loops through steps 1 to 3 100 times and then saves the set of solved problems in a JSON file. The resulting JSON file has the structure showcased in Listing 4.1:

[5] `https://github.com/mmabrouk/chatgpt-wrapper`

Listing 4.1: JSON Structure of res.json

```
[
    {
        body: str,
        answer: str,
        correct: bool,
    },
]
```

4.3.3 Experiment #1: Results

There are several key things I noticed during the benchmark, even before all the results made it apparent; Because I had to manually approve/disprove the solutions during the examination and evaluation, I also read multiple times through the questions and concluded that some of them are not logically sound. The example below displays one of the more common types of logical fallacy found in the set:

Zachary did 35 push-ups and three crunches in gym class today. David did 58 more push-ups but 87 fewer crunches than Zachary. How many more push-ups than crunches did Zachary do?

It is very apparent where the logical error is — if Zachary has done three crunches, how could David do 87 less? However, this is not a mistake in the set, though it may seem that way — SVAMP uses these types of problems and changes the question, so it can be answered correctly with 100% certainty, while the question that usually gets asked in these types of problems can not. An expected question would be "Who did more exercises", or something similar, that needs all the information in the problem. That would fail here because the second part is non-sensical. Thus, if the model relied on past experience and shallow heuristics to derive the goal of the problem, it would fail to find a solution for it — but if it genuinely read and answered the question given, it would not.

Overall, of 100 math problems, there were 22 inconsistencies — meaning the answer in the label and the answer given by the model were different. Not all of them, however, are mistakes I attribute to the model. I present a detailed report showcasing the shortcomings of the model, of the dataset, and general points of failure of either:

- Correct evaluation: 78%
    - Given the sheer size of the model, it comes as no surprise it handled the majority of these simple logical problems.
    - The solution is considered correct if:
        - Both the logic & the answer is correct
        - The model finds a logical fallacy and **correctly** explains it
        - The model ignores a present logical fallacy not pertaining to the goal and answers correctly
- Wrong logic: 8%
    - The model exhibits flawed logic in some MWP solutions

- An example pair of Q&A that I categorized here:

  Question (input entered by me):
  Jerry had three action figures on a shelf in his room. Later he added four more action figures to the shelf and removed some of the old ones. If there are six action figures on his shelf now, how many action figures did he remove from the shelf?
  Answer (generated by ChatGPT):

```
To find out how many action figures Jerry removed from the shelf,
we can subtract the initial number of action figures from
the current number of action figures.

Initial number of action figures = 3

The current number of action figures = 6

Number of action figures added = 4

Number of action figures removed = Current number of action
figures - (Initial number of action figures + Number of action
figures added)

Number of action figures removed = 6 - (3 + 4)

Number of action figures removed = 6 - 7 = -1
```

- Misguided use of information (5%), Misunderstanding of the problem (3%) & Wrong question answered (1%): 9%
  - I bundled here all the cases where the model failed to understand or use the provided information correctly — be it that it used information that was not necessary, misunderstood a part of it, or ignored some of it and generated something it was not tasked with.
  - An example pair of Q&A that I categorized here:

    Question (input entered by me):
    Tiffany was collecting cans for recycling. On Monday, she had 7 bags of cans. The next day she found 12 more bags worth of cans. How many more bags did she find on the next day than she had on Monday?
    Answer (generated by ChatGPT):

```
Tiffany found 19 more bags on the next day than she had on Monday.
To see why, we can use simple subtraction:
Number of bags on Tuesday = Number of bags on Monday +
Additional bags found
Number of bags on Tuesday = 7 bags + 12 bags
Number of bags on Tuesday = 19 bags
```

    The answer is inaccurate because the question was how many **more** — meaning the correct answer is 12.

- Wrong arithmetic: 2%
  - The model failed basic arithmetic operations (in both instances multiplication) in only 2% of the set. However, it failed the only two problems with bigger values

than a few hundred — prompting the question, how good is it with elementary mathematics with big numbers?

- I tested ten of the previous MWPs with much bigger numbers and asked the chatbot to multiply three big numbers ten times. Of the ten problems the model had no problem with previously, zero returned a correct result — all ten solutions had sound logic but wrong calculations. Multiplication yielded the same results.

- Problematic wording/badly constructed problems (Fault not in GPT): 3%

	Issues stemming from ChatGPT	Issues from dataset
Of the one hundred problems	19%	3%
Of the incorrect problems	86.4%	13.6%

Table 4.1: Incorrect solution/problem ratio

4.3.4 Experiment #1: Conclusion

Several important conclusions can be drawn from the experiment's results; First, my initial assumption about the LLMs QR proved correct, with a few caveats. I went into the experiment expecting the model to perform quite well, and with its 78% accuracy, it did. Considering it was not fine-tuned to solve MWPs at all, the result is truly satisfactory.

Thanks to my decision to use the SVAMP datasets, it was possible to measure and uncover the extent to which the model substitutes quantitative reasoning with shallow heuristics & lexical cues. Many of the problems designed to test that failed, meaning the model truly struggles with this.

The second big issue connected with ChatGPT's numeracy is its inability to handle problems and requests containing bigger numbers. The bigger the number, the lower the odds of calculating it correctly — the problem probably stems from the fact that smaller numbers are much more often encountered throughout the dataset on which ChatGPT was trained.

To conclude, Large Language Models seem to be quite capable of solving the majority of these problems — however, there are still quite a few issues in certain areas, so creating a robust MWP solver and utilizing transfer learning would provide a non-negligible boost to ChatGPT's abilities.

4.4 Experiment #2: Translating Ape210K

Experiment 4.3 proved the need for better quantitative reasoning in Large English Language Models — one of the sure ways to achieve that is to train the MWP solvers on more data. The main issue with doing that is the scarcity of data; Collecting and cleaning problems is time-consuming and non-trivial because there aren't many sources for MWPs.

Instead of turning to other sources, I decided to attempt to translate a large dataset in a foreign language to English. The problems can then be used to train English MWP solvers, which in turn would boost the QR of future-to-come models.

4.4.1 Experiment #2: Implementation Choices

First and foremost, a decision had to be made on which dataset to work with. There were two choices, Math23K and Ape210K (both reviewed in chapter 3.2). Math23K is widely used as a benchmark in the MWP solver scene[6], and so I decided to explore the other set — Ape210K. It seems quite underutilized, considering its size and diversity.

There isn't much information about Ape210K — because of that, experiment 4.5 examines the set more thoroughly. That was made possible thanks to the translation efforts in this experiment, as the problems were initially only in Chinese.

There are a few possible ways and tools to translate data from Chinese to English:

- Online interface for translators offered by Google, DeepL, etc.
    - Multiple corporations offer translation services for free through their website:
        - Google Translate[7] (Google)
        - DeepL Translator[8] (DeepL)
        - Bing Translate[9] (Microsoft)
        - Baidu Translator[10] (Baidu)
    - Most of these translators work very well and offer great service. However, they are not designed to handle data in bulk — the size of the dataset is a very limiting factor in this case.
- API interfaces of aforementioned translators
    - Unlike the web interfaces, the API endpoints of the previously listed companies handle bulk very well. The issue is almost all of these APIs require a commercial license. The sheer volume of Ape210K rules out this option.
    - The API endpoint of Baidu, however, is free[11], thus translating via Baidu is a possible course of action.

[6] https://paperswithcode.com/sota/math-word-problem-solving-on-math23k
[7] https://translate.google.com/
[8] https://www.deepl.com/en/translator
[9] https://www.bing.com/translator
[10] https://fanyi.baidu.com/
[11] https://fanyi--api-baidu-com.translate.goog/api/trans/product/prodinfo?_x_tr_sl=zh-CN&_x_tr_tl=en&_x_tr_hl=en&_x_tr_pto=sc

- Third-party options
  - In his thesis, NLP Methods for Word Problems (Krist, 2022), Krist encounters a similar obstacle because of his intentions to translate English problems to Czech. In his work, he uses a third-party tool written in Javascript[12]. There is merit to attempting to use the tool, as his bulk translation using it is a success.
- Large models from HuggingFace running on the RCI cluster
  - The Artificial Intelligence Center has provided access to their cluster for this thesis — thanks to them, I can attempt to translate the data using a model too computationally expensive for a personal computer.
  - The model chosen is the WMT 21 X-En (Tran et al., 2021).

4.4.2 Experiment #2: Tools & Execution

Unlike experiment #1, I chose to explore multiple possible options — the tools were different for each method. The environment, language, and hardware I chose for each were dictated by the needs of the tool/library/model used.

The npm module used by Kirst is written in Javascript, and so I had no choice but to code the "steering" of it in JavaScript as well (`baiduNpmTranslate.js`). In Kirst's work, he has reached the same conclusion.

The Baidu API is a REST API, and so I concluded that I best access it via JavaScript, too, as I have experience with dealing with APIs only in that language. Regrettably, there were some major issues regarding acquiring access, which are detailed in subsection4.4.3.

The WMT 21 X-En model is a multilingual translation model uploaded to and accessible through HuggingFace. Because of that, I did this part of the experiment I wrote in Python. This part is also the only computationally heavy one. Because of the resources needed to run the model, this method of translation was done on the RCI cluster.

4.4.3 Experiment #2: Results

I started testing out the different methods of translation to find the best-suited one. I began with the **npm module baidu-translate-api**, which supposedly simplifies the usage of the Baidu translator API. My attempts failed to produce the result I expected, as the library kept returning error codes. I ruled out the possibility that my code was the issue, as the code in Kirst's work also threw the same error once I ran it — meaning between the writing of his thesis and mine, the library became broken.

The next tool I attempted to use was the **Baidu API** itself — I figured the API had changed, and so the wrapper module no longer worked, but I should be able to use the API directly and write the wrapper myself. I started researching how to gain access to the API, and therein I found an issue. Baidu offers its services to parties outside mainland China via a special portal designed for foreign users[13]. Unfortunately, due to undisclosed reasons, the registration form is currently non-functional and only residents of mainland

[12] `https://www.npmjs.com/package/baidu-translate-api`
[13] `https://passport.baidu.com/v2/?reg&overseas=1`

China are allowed to register. Workaround methods[14] also do not work at this point in time.

Lastly, I turned to translation models I could run myself. The **WMT 21 X-En model** is a multilingual model developed by Facebook, which is their submission for the Sixth Conference On Machine Translation (Barrault et al., 2021) and its main aim is to translate news articles to and from English. Math problems often use similar language and structure to convey information, so a model designed specifically for news should yield better results than non-specific ones. The results, however, are subpar — despite its size, the model often missed the mark with its translations, and although it translated the problems well enough for a human to understand them, the quality is not high enough to be used as training samples.

4.4.4 Experiment #2: Conclusion

Although I tried many different venues, none of them produced results that could be used for training and subsequently improving an MWP solver. Though that may be true, the results from the WMT model are good enough for me to understand the problems (For example, often the translation model fails to translate the question — Figure 4.1. More examples in subsection 4.5.3). This means I can use these translations to examine the data I previously could not. The initial goal of translating the dataset and using it for training is not yet easily achievable with tools at our disposal, as the only possible method I have not tried that could yield satisfactory results is utilizing the commercial version of the translators mentioned in 4.4.1, which for now does not make financial sense.

To conclude, the results from this experiment cannot directly be piped to a model but could be used for reviewing and possibly cleaning & augmenting the dataset Ape210K. As for now, the set is underutilized, so perhaps a refined version of it could find its way to a more mainstream place.

A batch of goods weighs 160 tons, and after a part of it is transported away, the ratio of the transported and the remaining tonnage is 3:7.

Figure 4.1: Example of a badly translated problem

[14] https://www.adchina.io/how-to-open-a-baidu-account-outside-china/

4.5 Experiment #3: Examining Ape210K

Ape210K is the largest MWP dataset currently available, but its adoption as the new benchmark for math word problem-solving systems has yet to be seen. To understand the issue and subsequently attempt to fix it, I examined the dataset thoroughly to find where the main issue lies.

The main hurdle that had to be overcome to achieve that was the fact that the set is in a foreign language — thankfully, thanks to the previous experiment (4.4), I now had a translated chunk of the data, ready to be reviewed.

4.5.1 Experiment #3: Implementation Choices

I know very little about the data, so figuring out the types of problems and templates is the primary goal. The issue with the Ape210K paper (explained in 3.2.7) leaves the dataset almost entirely undocumented, so there is no evidence to compare my results with.

4.5.2 Experiment #3: Tools & Execution

Due to the circumstances surrounding the paper, it proved hard to find any information about the set. The abstract (available on arXiv.org[15]) hints that some problems require "commonsense" knowledge, and so I decided to investigate the extent and type of needed external knowledge.

Combing through the entirety of the set would be very time-consuming and with diminishing effect — it is a better idea to annotate a large enough chunk manually and then run automatic scripts to annotate the rest based on the labeled parts. The main goal was to label the problems with "contains external knowledge" and "does not rely on external knowledge". External knowledge & constants are things MWP solvers will eventually have to tackle, but first, robust, efficient solvers for less-problematic MWPs have to exist.

I created a tool that aided me in labeling a section of the dataset Ape210K. I considered different languages, libraries, and frameworks and ultimately decided on Javascript + React. The main reason is that there is much less boilerplate and code needed in order to produce a functioning user interface. I opted not to use already existing annotators[16], as they were too complex for the task at hand (plus I would have to upload the problems contained in a JSON to a database, creating additional overhead).

The application I built for the purpose of annotating the data has a simple work cycle, which is outlined in the steps below:

1. Parsing & Loading the unlabeled but translated chunk.
   - In this step, the application loads the JSON as if it's a normal Javascript object (another reason for selecting JavaScript for this project — it greatly simplifies interactions with JSON files).
2. Rendering the problems

[15] `https://arxiv.org/abs/2009.11506`
[16] such as `https://github.com/doccano/doccano`

- The number of problems initially was around nine thousand; Neither the Document Object Model (DOM) nor React is optimized for rendering that many objects at once. Because of that, problems are displayed one by one, each being replaced by the next, once a decision is reached on whether the problem contains external knowledge or not.
- Marking the problem as OK (=not containing EK, nor EC) can be done by either pressing the Button with that label or by pressing the key + on the numerical keypad. Labeling it negatively can be done by pressing the other button (also bearing an apt description) or by pressing − on the numerical keypad. These keyboard buttons were selected with ergonomics in mind.
- There is some additional information rendered, visible in Figure 4.2

3. Saving the annotated data

    - The format of the annotated data should be the same as to what the format of the initial data was. Because of that, the application returns a JSON. I used blobs and an npm module[17] to go around the need for a backend for interacting with the filesystem (a security measure in all browsers, but under certain circumstances, downloading small files called blobs are allowed).
    - Web applications are not very optimized for working with large arrays and sets, and so to ease the manipulation with them, the app just discards the OK problems and remembers only which ones are bad — after the file is downloaded, the initial file containing all problems and the downloaded file containing only the bad ones are used in conjunction, to create a labeled set of problems.

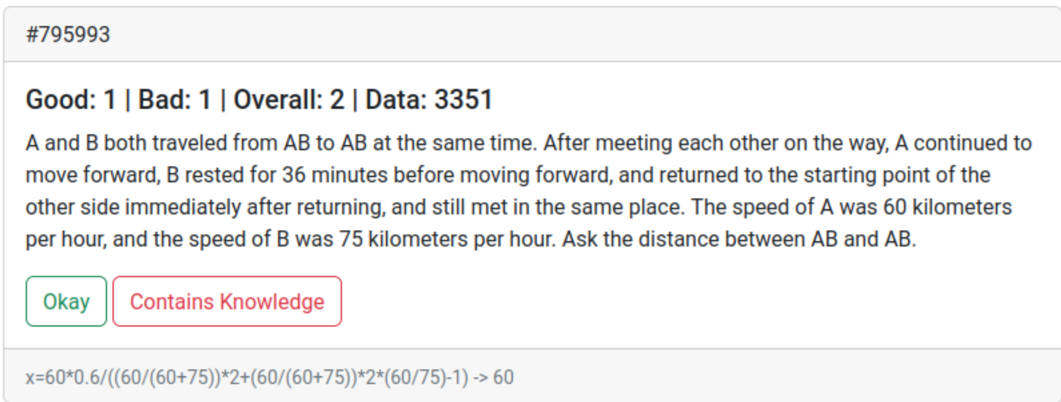

Figure 4.2: Problem showcase in the application; Shows additional information, such as problems annotated overall, # of ones containing knowledge, etc.

[17] https://www.npmjs.com/package/file-saver

Ok:
- 472537 • 535732 • 127854
- 442846 • 1073895 • 503335
- 443478 • 416151 • 445527
- 190732

Bad:
- 183817 • 795993 • 522885
- 464401 • 845907 • 510550
- 807889 • 673053 • 97648

Figure 4.3: Lists of annotated problems' id, separated into two columns, based on whether the problem they identify contains EK or not.

4.5.3 Experiment #3: Results

With the help of the tool, I annotated over two and a half thousand problems. There were some ambiguities as to what should be considered external knowledge, so I've detailed below all the different types of problems that I've concluded that contain EC (External constants) or EK (External knowledge). Based on my annotations, approximately 34.4% of the problems contain EK. Their distribution is as follows:

- Area / Volume: 43%
  - Many of the word problems are geometry related — this is generally problematic, as almost all geometry problems require external knowledge and constants. The formulas for areas and volumes of different shapes are never in the problem but are rarely not expected. Constants such as π are also very often required.
  - An example of such a problem:

    The surface area of a cylinder is 1884 square centimeters, the bottom radius is 10 centimeters, and its height is how many centimeters.

    Here the area and volume of a cylinder is absolutely necessary for solving the problem, yet it is not provided. Thus, this particular problem relies on external knowledge.
- Velocity / Distance: 16%
  - Problems pertaining to calculating the velocity of entities, their distance from each other, or time of interception all fall into the category EC & EK. One needs to be aware of the relation between velocity, distance, and time to be able to calculate them.
  - Example:

    A truck and a passenger car leave from A and A at the same time and follow the same route to B. It is known that the truck travels 75 kilometers per hour, and the speed of the passenger car is 1.2 times that of the truck. After 3.6 hours, the passenger car arrives at B. At this time, how far is the truck from B?

    The model would have to know how to derive the distance traveled from the velocity of the entity and the time spent moving.
- Interest rate / Compound interest: 11%
  - Understanding and working with interest definitely require external knowledge of financial mathematics.

- Example of such a problem:

  Dad gave Xiao Ming a 20,000 yuan education deposit of 20,000 yuan, with a maturity of three years and an annual interest rate of 5.40%. After the maturity, how much of the principal and interest can be withdrawn?

- Combinations / Variations / Permutations: 7%
  - Such calculations require knowledge of probability, which is an external, non-elementary domain of mathematics. Problems requiring combinations, variations, or permutations all receive an EK label.
  - Example:

    There are 5 white balls and 3 yellow balls in a box, and you can touch any one of them. What is the probability of touching the yellow ball once?

- Miscellaneous: 23%
  - All other problems containing EC or EK fall into this category because they are much less common. These problems require various external knowledge, like the number of edges of different shapes, knowledge of the mechanism of leap years, etc.

To make full use of the newly gained information, I used it to fine-tune a pre-trained BERT model. The model was pre-trained in Chinese, and I adapted it to a classification task; The models fit for training solvers were labeled with 1, the labeled containing EK, and thus unfit, with 0. Using the three thousand samples and HuggingFace's library, I trained the model on the original text, not the translated one. This meant one crucial thing — the unfit problems can be filtered out in their original language, saving a lot of computational power.

4.5.4 Experiment #3: The Model

The model I developed is fine-tuned version of a pre-trained model. Because the text I wished to filter is in Chinese, I had to select a model that was pre-trained on Chinese data. I selected one from the BERT family — `bert-base-chinese`[18]

Using the transformer library and the Trainer API[19] from HuggingFace, I fine-tuned the model to differentiate between problems that contain external knowledge or constants and those that do not. The overall dataset was divided as follows:

Overall	Training	Testing
2744	2195	549

Table 4.2: Train/Test split of annotated data

The data that I input was the data I received from the annotation tool and later processed with `process_chunks.py`:

[18] `https://huggingface.co/bert-base-chinese`
[19] `https://huggingface.co/docs/transformers/main_classes/trainer`

Listing 4.2: Structure of input for the model

```
[
    {
        original_text: str,
        label: int,
    },
    ...
]
```

After the model was fine-tuned, I ran it on 200,488 problems. The results are illustrated below:

Overall	Contain EK	Do not contain EK
200,4888	61563	138925

Table 4.3: Ratio of problematic/non-problematic problems in Ape210K

Overall, 69.3% of the problems remained, while 30.7% were purged. This is within 4% of what I observed during the annotation of 3000 samples. The gap between the two can be explained by the fact that sometimes the translation of the problem was not good enough for me to comprehend the problem, and I had to guess the label based on the equation. These occurrences were rare, but I believe they are the majority of the misalignment.

To further test the model's accuracy, I took the data it had labeled and translated 300 problems at random to verify manually if the labels generated by the model matched the correct category of the problem. The results are presented in Table 4.4.

	Contain EK	Do not contain EK
Model marked as OK	4 (4.3% Error)	192
Model marked as NOT OK	90	14 (6.8% Error)

Table 4.4: Ratio of problematic/non-problematic problems in Ape210K. Error rates inside cells expose incorrectly labeled X : all labeled X

The model has an accuracy of 94%, which can also be attributed to mistakes I made during the annotation process. It errs more often on the side of caution, pruning viable problems — this is the less problematic option because the database of problems is large enough. It is better to shrink it more than to lose quality.

4.5.5 Experiment #3: Conclusion

Although the translations from experiment 4.4 were not of high enough quality to be used as training samples, the majority were translated well enough to be understandable by a human reader. Thanks to that, I was able to utilize it to examine, detail & process the foreign data. The results indicated that a substantial portion of these problems rely on external knowledge and constants. Such problems add a lot of complexity to the problem, and they should be avoided until solvers are proficient in less daunting types of MWPs.

Ridding the dataset of EK-reliant problems also shrinks the volume of the set considerably, which makes it less computationally expensive to translate. As new translation models emerge, soon this route of obtaining new problems will be open — it pays to optimize the data before the time comes.

Thanks to the data I annotated and prepared, I was able to do just that — the model pre-trained on Chinese corpora and fine-tuned on my labeled section was able to finish the annotation automatically in the original language. It filtered out roughly thirty percent of the problems in the set, reducing the overall size. From the initial 200,488 math word problems, 61,563 were removed, leaving 138,925 viable items.

Thanks to this examination and the proposed model, future endeavors with this dataset will yield better results and will be less computationally expensive — the model filters out problems containing external knowledge with an accuracy of 94%[20], meaning fewer resources will have an to be spent, and the resulting dataset will be of much higher quality.

[20] manually tested on 300 samples

Chapter 5
Conclusion

Numeracy, quantitative reasoning, and critical thinking are interconnected concepts vital to every aspect of life. This thesis aimed to explore the extent to which Artificial Intelligence is capable of mimicking those attributes and building upon them. I decided to perform my analysis on models focused on solving math word problems; Quantitative reasoning is indispensable in this context, and the fact that the models generate equations as part of the result allows us to peer further into the thought process.

The first section of my work explored and reviewed the state-of-the-art models of the past few years, focusing on those that significantly improved their accuracy over their predecessors. I also examined the general architecture they were built on and the additional mechanisms that provided them their edge.

Next, I focused on the datasets and the types of problems these different sets offered. I reviewed and outlined each set's positives and negatives, which significantly contributed to understanding and fixating on the possible issues the models trained from these datasets could have. Thanks to the detailed review of the different datasets, I knew what to look for when inspecting a model and its true QR capabilities.

That knowledge allowed me to analyze more thoroughly a model not explicitly trained for MWP but large enough to be quite proficient at it — ChatGPT. I deemed that an essential task for my thesis, as I consider the eventual goal of MWP solvers to be transferring their advanced numeracy understanding to LLMs, such as ChatGPT or other big models, such as fact-checkers. I concluded that ChatGPT often relied on shallow heuristics and struggled heavily with larger numbers, proving that there is room for improvement and that the work on MWP solvers is far from futile.

After reaching that conclusion, I shifted my efforts towards improving current SOTA models. There are multiple ways to achieve that, and I focused on expanding the data from which English MWP solvers trained. To do that, I resolved to translate a Chinese problem set to English — Ape210K.

The first part of the experiment was collecting and detailing all the possible ways to translate the large bulk of data; Then, I systematically attempted all methods. The most promising one still yielded unsatisfactory results, but I used the data I had already translated in the next experiment.

Although the problems I translated could not be used directly to improve English solvers' effectiveness, I created a tool in React to help me annotate a big portion of them and subsequently train a model that filtered out problems too problematic for current solvers, reducing the overall size of the set by roughly thirty percent.

My initial intentions were to prepare and train a model that surpasses the current state-of-the-art math word problem solvers by translating the Ape210K dataset. The experiments I conducted revealed the set needed advanced processing & filtering before being used for training, and so after consulting my supervisor we decided to divert the course of this thesis and instead train a model focused on purging problems containg external knowledge from the set.

Future work on the subject will be easier now that external knowledge & constants have been removed from the set — Chinese models will have a cleaner set to work with. Not only that but with the rapid advancement of AI, soon even more capable translation models will emerge — once they do, translating the set will be that much computationally easier, thanks to the fact that unusable problems have been filtered out pre-translation.

Bibliography

BARRAULT, Loic; BOJAR, Ondrej; BOUGARES, Fethi; CHATTERJEE, Rajen; COSTA-JUSSA, Marta R.; FEDERMANN, Christian; FISHEL, Mark; FRASER, Alexander; FREITAG, Markus; GRAHAM, Yvette; GRUNDKIEWICZ, Roman; GUZMAN, Paco; HADDOW, Barry; HUCK, Matthias; YEPES, Antonio Jimeno; KOEHN, Philipp; KOCMI, Tom; MARTINS, Andre; MORISHITA, Makoto; MONZ, Christof (eds.), 2021. *Proceedings of the Sixth Conference on Machine Translation*. Online: Association for Computational Linguistics. Available also from: `https://aclanthology.org/2021.wmt-1`.

BOWMAN, Samuel R.; ANGELI, Gabor; POTTS, Christopher; MANNING, Christopher D., 2015. A large annotated corpus for learning natural language inference. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

CHOWDHERY, Aakanksha; NARANG, Sharan; DEVLIN, Jacob; BOSMA, Maarten; MISHRA, Gaurav; ROBERTS, Adam; BARHAM, Paul; CHUNG, Hyung Won; SUTTON, Charles; GEHRMANN, Sebastian; SCHUH, Parker; SHI, Kensen; TSVYASHCHENKO, Sasha; MAYNEZ, Joshua; RAO, Abhishek; BARNES, Parker; TAY, Yi; SHAZEER, Noam; PRABHAKARAN, Vinodkumar; REIF, Emily; DU, Nan; HUTCHINSON, Ben; POPE, Reiner; BRADBURY, James; AUSTIN, Jacob; ISARD, Michael; GUR-ARI, Guy; YIN, Pengcheng; DUKE, Toju; LEVSKAYA, Anselm; GHEMAWAT, Sanjay; DEV, Sunipa; MICHALEWSKI, Henryk; GARCIA, Xavier; MISRA, Vedant; ROBINSON, Kevin; FEDUS, Liam; ZHOU, Denny; IPPOLITO, Daphne; LUAN, David; LIM, Hyeontaek; ZOPH, Barret; SPIRIDONOV, Alexander; SEPASSI, Ryan; DOHAN, David; AGRAWAL, Shivani; OMERNICK, Mark; DAI, Andrew M.; PILLAI, Thanumalayan Sankaranarayana; PELLAT, Marie; LEWKOWYCZ, Aitor; MOREIRA, Erica; CHILD, Rewon; POLOZOV, Oleksandr; LEE, Katherine; ZHOU, Zongwei; WANG, Xuezhi; SAETA, Brennan; DIAZ, Mark; FIRAT, Orhan; CATASTA, Michele; WEI, Jason; MEIER-HELLSTERN, Kathy; ECK, Douglas; DEAN, Jeff; PETROV, Slav; FIEDEL, Noah, 2022. *PaLM: Scaling Language Modeling with Pathways*. Available from arXiv: `2204.02311 [cs.CL]`.

HOWARD, Jeremy; RUDER, Sebastian, 2018. *Universal Language Model Fine-tuning for Text Classification*. Available from arXiv: `1801.06146 [cs.CL]`.

HUANG, Danqing; SHI, Shuming; LIN, Chin-Yew; YIN, Jian; MA, Wei-Ying, 2016. How well do Computers Solve Math Word Problems? Large-Scale Dataset Construction and Evaluation. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 887–896. Available from DOI: `10.18653/v1/P16-1084`.

JIE, Zhanming; LI, Jierui; LU, Wei, 2022. Learning to Reason Deductively: Math Word Problem Solving as Complex Relation Extraction. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5944–5955.

KIM, Seonhoon; KANG, Inho; KWAK, Nojun, 2018. *Semantic Sentence Matching with Densely-connected Recurrent and Co-attentive Information*. Available from arXiv: `1805.11360 [cs.CL]`.

KONCEL-KEDZIORSKI, Rik; ROY, Subhro; AMINI, Aida; KUSHMAN, Nate; HAJISHIRZI, Hannaneh, 2016. MAWPS: A Math Word Problem Repository. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, pp. 1152–1157. Available from DOI: `10.18653/v1/N16-1136`.

KRIST, Ronald, 2022. *NLP Methods for Word Problems*. MA thesis. Czech Technical University.

KUSHMAN, Nate; ARTZI, Yoav; ZETTLEMOYER, Luke; BARZILAY, Regina, 2014. Learning to Automatically Solve Algebra Word Problems. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, pp. 271–281. Available from DOI: `10.3115/v1/P14-1026`.

LIU, Yiheng; HAN, Tianle; MA, Siyuan; ZHANG, Jiayue; YANG, Yuanyuan; TIAN, Jiaming; HE, Hao; LI, Antong; HE, Mengshen; LIU, Zhengliang; WU, Zihao; ZHU, Dajiang; LI, Xiang; QIANG, Ning; SHEN, Dingang; LIU, Tianming; GE, Bao, 2023. *Summary of ChatGPT/GPT-4 Research and Perspective Towards the Future of Large Language Models*. Available from arXiv: `2304.01852 [cs.CL]`.

MEHDI, Yusuf, 2023. *Announcing the next wave of AI innovation with Microsoft Bing and Edge*. Available also from: `https://blogs.microsoft.com/blog/2023/05/04/announcing-the-next-wave-of-ai-innovation-with-microsoft-bing-and-edge/`. Accessed on May 15th, 2023.

MIAO, Shen-yun; LIANG, Chao-Chun; SU, Keh-Yih, 2020. A Diverse Corpus for Evaluating and Developing English Math Word Problem Solvers. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 975–984.

MUZAINI, M; JUNIATI, D; SISWONO, T Y E, 2019. Exploration of student's quantitative reasoning in solving mathematical problem: case study of field-dependent cognitive style. *Journal of Physics: Conference Series*. Vol. 1157, no. 3, p. 032093. Available from DOI: `10.1088/1742-6596/1157/3/032093`.

PATEL, Arkil; BHATTAMISHRA, Satwik; GOYAL, Navin, 2021. Are NLP Models really able to Solve Simple Math Word Problems? In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, pp. 2080–2094. Available from DOI: `10.18653/v1/2021.naacl-main.168`.

RAVICHANDER, Abhilasha; NAIK, Aakanksha; ROSE, Carolyn; HOVY, Eduard, 2019. *EQUATE: A Benchmark Evaluation Framework for Quantitative Reasoning in Natural Language Inference*. Available from arXiv: `1901.03735 [cs.CL]`.

SHI, Shuming; WANG, Yuehui; LIN, Chin-Yew; LIU, Xiaojiang; RUI, Yong, 2015. Automatically Solving Number Word Problems by Semantic Parsing and Reasoning. In: *The 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1132–1142. Available also from: `https://aclanthology.org/D15-1135`.

TOUVRON, Hugo; LAVRIL, Thibaut; IZACARD, Gautier; MARTINET, Xavier; LACHAUX, Marie-Anne; LACROIX, Timothée; ROZIÈRE, Baptiste; GOYAL, Naman; HAMBRO, Eric; AZHAR, Faisal; RODRIGUEZ, Aurelien; JOULIN, Armand; GRAVE, Edouard; LAMPLE, Guillaume, 2023. *LLaMA: Open and Efficient Foundation Language Models*. Available from arXiv: `2302.13971 [cs.CL]`.

TRAN, Chau; BHOSALE, Shruti; CROSS, James; KOEHN, Philipp; EDUNOV, Sergey; FAN, Angela, 2021. Facebook AI's WMT21 News Translation Task Submission. In: *Proc. of WMT*.

VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia, 2017. *Attention Is All You Need*. Available from arXiv: `1706.03762 [cs.CL]`.

WANG, Thomas; ROBERTS, Adam; HESSLOW, Daniel; SCAO, Teven Le; CHUNG, Hyung Won; BELTAGY, Iz; LAUNAY, Julien; RAFFEL, Colin, 2022. *What Language Model Architecture and Pretraining Objective Work Best for Zero-Shot Generalization?* Available from arXiv: `2204.05832 [cs.CL]`.

WANG, Yan; LIU, Xiaojiang; SHI, Shuming, 2017. Deep Neural Solver for Math Word Problems. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 845–854. Available from DOI: `10.18653/v1/D17-1088`.

WILLIAMS, Adina; NANGIA, Nikita; BOWMAN, Samuel, 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 1112–1122. Available also from: `http://aclweb.org/anthology/N18-1101`.

XIE, Zhipeng; SUN, Shichao, 2019. A Goal-Driven Tree-Structured Neural Model for Math Word Problems. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, pp. 5299–5305. Available from DOI: `10.24963/ijcai.2019/736`.

ZHANG, Jipeng; WANG, Lei; LEE, Roy Ka-Wei; BIN, Yi; WANG, Yan; SHAO, Jie; LIM, Ee-Peng, 2020. Graph-to-Tree Learning for Solving Math Word Problems. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 3928–3937. Available from DOI: `10.18653/v1/2020.acl-main.362`.

ZHAO, Wei; SHANG, Mingyue; LIU, Yang; WANG, Liang; LIU, Jingming, 2020. *Ape210K: A Large-Scale and Template-Rich Dataset of Math Word Problems*. Available from arXiv: `2009.11506 [cs.CL]`.

Appendix A
Acronyms

BERT Bidirectional Encoder Representations from Transformers

DOM Document Object Model

EC External constants

EK External knowledge

GPT Generative Pre-trained Transformer

LLM Large language model

LLaMA Large Language Model Meta AI

LMM Large multimodal model

MWP Math word problems

NLI Natural language inference

PaLM Pathways Language Model

SOTA State-of-the-art

TE Textual entailment

QR Quantative reasoning

Appendix B

Repository Structure

```
korlavik.zip ............................................. main directory
├── exp1
│   └── chatGpt_interactor.................................code related to 4.3
│       ├── chatGptParser.py ......................... algorithm explained in 4.3.2
│       ├── logParser.py...........rebuilds set from logs in event of premature cancel
│       ├── separateIncorrect.py ......................................... utility
│       └── ...
├── exp2 ................................................. code related to 4.4
│   ├── js_module
│   │   ├── package.json
│   │   ├── baiduNpmTranslate.js....attempt at using the module mentioned in 4.4.2
│   │   └── ...
│   └── rci_translation
│       └── translate.ipynb......................translation using WMT X-21 En
└── exp3 ................................................. code related to 4.5
    ├── data_from_rci.............................................analysis tools
    │   ├── model.ipynb.........................................model fine-tuning
    │   ├── *dataX.json[1]
    │   ├── process_chunks.py ............................................ utility
    │   ├── visualise.ipynb.....................................data examination
    │   ├── *filter_model..........................trained model & tokenizer saved
    │   │   └── ...
    │   └── ...
    └── annotation_tool
        ├── package.json
        └── src
            ├── App.js .....................................contains main logic
            ├── data ..............................contains problems to be annotated
            │   ├── 9ktranslated.json
            │   └── 9ktranslated_untouched.json
            └── ...
```

[1] items marked with * can be provided on request