University of Bergen
Department of Informatics

Rule learning of the Atomic dataset using Transformers

Author: Kristoffer Æsøy
Supervisor: Ana Ozaki

UNIVERSITETET I BERGEN
Det matematisk-naturvitenskapelige fakultet

May, 2023

Abstract

Models used for machine learning are used for a multitude of tasks that require some type of reasoning. Language models have been very capable of capturing patterns and regularities found in natural language, but their ability to perform logical reasoning has come under scrutiny. In contrast, systems for automated reasoning are well-versed in logic-based reasoning but require their input to be in logical rules to do so. The issue is that the conception of such systems, and the production of adequate rules are time-consuming processes that few have the skill set to perform. Thus, we investigate the Transformer architecture's ability to translate natural language sentences into logical rules. We perform experiments of neural machine translation on the DKET dataset from the literature consisting of definitory sentences, and we create a dataset of if-then statements from the Atomic knowledge bank by using an algorithm we have created that we also perform experiments on.

Acknowledgements

I could not have undertaken this journey without my supervisor Ana Ozaki, whose patience, encouragement, and expertise made all of this possible. I am also grateful for my family, who have supported me through everything and helped me every step of the way.

Kristoffer Æsøy
26 May, 2023

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Artificial intelligence is becoming a larger and larger part of our everyday lives, from major leaps such as self-driving cars to smaller everyday conveniences like optimizing our music playlists. In particular, the subfield of natural language processing (NLP) has come a long way in helping us analyze and generate text that solves a lot of different tasks. Sentiment analysis, text-to-speech, and machine translation are becoming widespread, and the state-of-the-art language models (LM) have become so powerful that they are being adopted as mainstream tools for the general public.

For the LMs to be capable of producing adequate results for the tasks they are designated to perform, they need to be able to reason in some form. They need to grasp the nature of language, what words mean, and which words are important for the current context. To achieve this, the LMs are analyzing the statistical features of the text they train on, to look for patterns and regularities in an unsupervised fashion. Afterward, they may be fine-tuned to be more fitting to the specific task the model is intended to perform, such as question-answering or text completion. By exposing them to this labeled data, they can learn to generalize from provided examples and generate appropriate responses.

LMs have become incredibly good at capturing patterns, but studies have observed an incapability to perform logic-based reasoning [8, 28]. If the quality of the data is subpar, biased, or contains factually incorrect statements, the model will not be aware of these faults and may treat them as truths to generalize from. The community has also observed results that indicate that models do not perform logical reasoning in situations where it is the only

trait of the data [9], but instead that the model learns from patterns in the input, rather than a generalized reasoning function to work on other distributions or comprehend certain aspects of compositionally which would display logical reasoning.

On the other hand, the automated reasoning community has developed several tools to perform logic-based reasoning over the years. Logical rules play a fundamental role as they provide a formal framework for representing knowledge and reasoning about it. Automated reasoning systems use logical rules and algorithms to manipulate and derive new statements from the given axioms. They follow a deductive process where logical rules are applied to a knowledge base to infer new information or verify the validity of existing statements. However, a concern of the reasoners is that the production of the logical rules to be used for the systems is a tedious and time-exhaustive task. It requires expertise in the field, which there are few who have, and to manually design all the rules necessary to capture knowledge found in natural language.

Related work There are other works that try to semi-automate the ability to create rules from natural language. A notable example is LExO [26], which aimed to transform definitions in natural language into OWL-DL axioms using a syntactic parser. In terms of using neural networks in an end-to-end fashion, the experiments using recurrent neural networks to translate from definitory sentences into the DL language ALCQ [19] are the biggest influence on our work. The original model created was very limited, but the syntactic approach was later improved by introducing a new model capable of translating definitions with unknown words in them and working outside of a controlled language domain [20].

In our work, we seek to test the ability of the Transformer LM architecture to perform neural machine translation from natural language to logical rules. We focus on a subset of natural language, in the form of if-then statements found in the knowledge base known as Atomic [23]. We have created a dataset to train the LM using a self-made algorithm that finds atoms in the event and inference and we transform them into the bodies and heads of rules that capture the meaning of the natural language if-then expression. We then train the Transformer model to learn to translate from the if-then expression into these logical rules and observe how well it is able to do so. In this fashion, we exploit the best of both worlds, where machine translation from the AI community is used to create the rules, and reasoners developed by the automated reasoning community can use the rules to perform

logic-based reasoning. We present our results on datasets from previous work [20] and the Atomic dataset [23] in both small and large quantities where we measure the accuracy of translating the token of the rules and the rules as a whole.

In this thesis, we will first look at what reasoning is, both in terms of inductive and deductive reasoning and the language models that employ it to perform NLP tasks. Then we will look at the previous work of creating logical rules and a model that is capable of performing syntactic translation from definitions into the created rules [20]. Then we will talk about the Atomic [23] knowledge bank, and the challenge of creating a dataset consisting of logical rules from it, resulting in our very own algorithm to perform this for us. Finally, we will look at the experiments performed, and conclude how the results and work done can shape future applications of this task.

Chapter 2

Background

In this chapter, we will look at the background material for both inductive and deductive reasoning and the language models that are relevant to this work. We will look at how reasoning relates to logic, and how to represent it syntactically. Then we will talk about Recurrent Neural Networks (RNN) and Transformers, and how the mechanics they use are used for Natural Language Processing tasks. In addition, we will take a deeper look at the attention mechanic, and how the one used to improve RNNs' abilities to perform sequence-to-sequence tasks differs from the one that is the fundamental part of the Transformer.

2.1 Inductive reasoning

Acquiring knowledge by inductive reasoning is one of the most intuitive ways we learn to understand things. Collecting larger and larger amounts of empirical data strengthens our ability to draw conclusions that are increasingly probable according to our data. Many relations between entities can be learned as such, without actually being required to understand the properties of the entities that are involved in the resulting outcome of mixing them. You can intuitively learn the correlation between running on ice and falling on your face, without having to understand the physics of friction. The concept of a *slippery* surface such as ice can be attributed to many things, but there is no need to understand a generalized principle that can be applied to understand the nature of all slippery surfaces to be able to interact

and understand any singular example. This human ability to draw inferences from repeated empirical experiences has been integral to our ability to survive and adapt to all societies we have lived in throughout our history as a species.

In many instances, applying the practice of inductive reasoning to the phenomena around us has caused us to believe in many *unwritten* truths. Things that we assume to be right because we have received no reason to not believe them to be so. "It is obvious that if we are asked why we believe the sun will rise tomorrow, we shall naturally answer, 'Because it always has risen every day.' We have a firm belief that it will rise in the future because it has risen in the past." [22]

However, the issue at hand that makes inductive reasoning a figurative slippery slope is the fact that there is a possibility that our sensory impressions may betray us. In other words, the things we observe lead us to draw untruthful conclusions. Inductive reasoning relies on probabilities rather than certainties making it prone to counterexamples, which are instances that contradict the generalization or prediction made. Thus, observations that may be biased to our established interpretations can still not provide a causal explanation for the observed patterns. A good example is the idea of the geocentric model, which was the predominant explanation for the nature of the universe in ancient civilizations. To the naked eye, looking up at the sky from the Earth's surface, it would look like the sun is revolving from horizon to horizon, while our planet stands still. Without the combination of a plethora of other observable natural phenomena that provided counterexamples to the inductive hypothesis, we would not have developed the heliocentric worldview, which could further explain the nature of the other celestial bodies and their relationship to one another.

The idea of drawing inferences and acquiring knowledge through induction has been a part of the philosophical questions about knowledge as a whole for millennia [5]. The fact that beliefs can be both justified and true but still be logically incorrect, showcases that there is a lot of uncertainty involved despite the incredible amount of perceived knowledge we can acquire through inductive reasoning. Still, inductive reasoning is an incredibly powerful tool for us humans as well as systems that try to capture how our world works if we combine it with a tiny bit of critical thinking.

2.2 Logic and deductive reasoning

Logic is the study of reasoning and argumentation with a long, rich history spanning thousands of years, all the way back to the days of Ancient Greece where the titans of philosophy established the foundations of logic. In Aristotle's classic *Prior Analytics* [2] the concept of a syllogism was defined for the first time. The syllogism arises when two statements asserted to be true validly imply a conclusion. A classic example is: "All humans are mortal. Socrates is a human. Therefore, Socrates is mortal." As long as the first two statements are true, then the conclusion can not be anything but true. There is a transitive property $Socrates \rightarrow Human$ and $Human \rightarrow Mortal$, thus $Socrates \rightarrow Mortal$.

The syllogism ended up being a core part of what would later be known as historical deductive reasoning, the ability to draw inferences as logical consequences from its premises. The fact that a valid deduction requires an impossibility for the conclusion to be false if the premises are true provides us with the possibility to formalize and structure logic using different inference rules. Logic exists in many different forms and syntaxes that try to describe and reason about the relationships between statements and propositions in a rigorous fashion by entailing restrictions to their expressivity. For example, syllogism is a form of logic that offers the possibility to draw valid conclusions from axioms, formalized using categorical propositions. However, it is very bare-bones in terms of what you can express due to the nature of only drawing conclusions from two premises that are required to be assumed to be true.

Propositional logic is a branch of formal logic that studies the logical relationships between propositions, which are statements that can either be true or false. In propositional logic, the emphasis is on the logical connectives that are found in natural language such as "and", "or", and "not", that are used to combine propositions to form compound propositions. The primary goal of propositional logic is to determine whether a given compound proposition is true or false based on the truth values of its components and inference rules.

To demonstrate, the most simple inference rule is modus ponens which states that if you have two statements P and Q, and two predicates: that P is true and P implies Q, then Q has to be true as well. The same argument expressed with natural language could be: "If today is Monday, then Garfield wants lasagna. Today is Monday. Therefore, Garfield wants lasagna.". The important distinction between syllogisms and propositional logic is that the

former involves reasoning about the properties of concepts based on their relationships to each other, while the latter focuses on the logical relationships between propositions and the rules for combining them.

Propositional logic is also the foundation for another form of logic known as first-order logic. First-order logic, also known as predicate logic, is a more expressive formal system than propositional logic that allows for the representation and reasoning of complex properties and relationships of objects or entities. Statements are represented using symbols to denote concepts and predicates representing properties or relationships between them. First-order logic has also ultimately superseded syllogisms due to its larger scope and expressive power.

We consider a countably infinite set of variables $\mathcal{V}$, in which we use letters x, y, z to express, and expressions such as $\boldsymbol{x}$ for a tuple $\langle x_1, ..., x_n \rangle$ of the corresponding variable elements. We also have an infinite possible amount of predicates Π. A predicate $\pi \in \Pi$ is combined with variables $x \in \mathcal{X}$ to describe an atom α, an expression in the form of $\pi(v_1, ..., v_n)$ that has an *arity* n that determines the number of variables in the atom: $|\mathbf{x}| = ar(\pi)$.

A first-order *rule* is a formula defined in the following form:

$$\forall \boldsymbol{x}, \boldsymbol{z}.(\varphi[\boldsymbol{x}, \boldsymbol{z}] \rightarrow \exists \boldsymbol{y}.\psi[\boldsymbol{x}, \boldsymbol{y}])$$

In this work, the rule consists of a body φ and a head ψ, which are conjunctions of atoms where both φ and ψ contain at least one atom each. The variables in $\boldsymbol{x}, \boldsymbol{z}$ that appear in φ are universally quantified over the entire rule, despite only some of them, $\boldsymbol{x}$, can also appear in ψ. The variables $\boldsymbol{y}$ that only appear in ψ are existentially quantified in the head. The conjunction of the atoms is represented by $\wedge$ and the implication of the body to the head is represented by $\rightarrow$.

We can for example take a look at how the syllogism about Socrates would be represented as first-order logic. We are once again expressing concepts and individuals, but instead of considering boolean statements we can instead quantify, and generalize into a rule. The property of being a human can be represented as $Human(x)$ where the predicate is applied to any x, and the same can be done for the concept of being mortal as $Mortal(x)$. As being mortal is an implication drawn from being a human, we then say that $Human(x)$ appears

in φ, while $Mortal(x)$ appears in ψ. Finally, it applies to "all humans" and as such any member, x has to be universally quantified with $\forall x$, resulting in the following generalized rule:

$$\forall x(Human(x) \rightarrow Mortal(x))$$

Now we have defined a rule that states that "All humans are mortal". Socrates is a constant and is stated to be a human, which means that he is a member of $Human(x)$ and x can be substituted by him, resulting in:

$$Human(Socrates) \rightarrow Mortal(Socrates)$$

By applying the rule of modus ponens, which allows you to derive a conclusion from an implication and its antecedent, you can infer the conclusion that Socrates is mortal, represented as $Mortal(Socrates)$. As such, we have not only been able to represent the syllogism in first-order logic but in addition, defined a rule that can be generalized to not only Socrates but all other humans as well.

First-order logic is incredibly powerful and can be tailored and structured to capture the representation of knowledge in many different domains. The description logic (DL) community for example studies various fragments of first-order logic and applies it to knowledge representation [1]. There also exist large knowledge banks that capture concepts and their relationships [24, 23] using first-order logic as well.

2.3 Language models

In this section, we will look at the RNN and the Transformer language model. We describe their application, underlying structure, and limitations. In addition, we go further in-depth into the attention mechanic and how they exploit it in their respective architectures.

2.3.1 Recurrent Neural Networks

For a long time, RNNs were the de-facto standard for processing sequential data. The language model calculates an output for every token/string/tensor in the input sequence and uses the output as a factor to calculate the next token's output. The RNN can use an internal state of a cell as a working memory to be able to draw information from tokens hundreds of timesteps earlier in the input sequence. The nature of this controlled state that has a feedback loop can vary, and some of the most popular ones are gated recurrent units (GRU) and long short-term memory (LSTM) networks.

One critical aspect of the RNN architecture is that input is handled sequentially, which in itself works very well for language. An example is that the sentiment of an adjective in a sentence is flipped on its head if the previous word such as "not" negates it. Thus, it performs well in many tasks that require context from earlier in the sequence to predict what the output should be. This does not mean that the RNN is incapable of drawing information from both tokens appearing earlier in the sequence as well as later tokens at the same time. There exist models using bidirectional RNNs that go through a sentence in both directions and combine the results for a more comprehensive representation. Some tasks, such as target-oriented opinion words extraction have used this approach [6] to achieve greater results than previous strategies.

However, there is a fundamental restriction to this, and that is the fact that all comparisons are linear, in the sense that to get the information for any given word in the sentence we need to process the entire sentence as a whole. RNNs are designed around the Markov property [15], where each state is assumed to be dependent only on the previous state, i.e. that the previous state is supposed to hold all the relevant information about all previous states. This means that we cannot do a direct comparison between two words in different parts of the sequence without including information about between states, as it is assumed to be relevant despite not always being in some cases.

This property can result in a struggle to retain useful information from early tokens in the sequence, resulting in what is known as the vanishing gradient problem [12, ch. 9.5]. To improve on this issue a mechanic known as attention was introduced to address it. Instead of only processing the entire sequence, the relevance of each element is also calculated at every timestep, which allows the network to pay more attention to important parts of the

input sequence, and less attention to irrelevant parts. In the context of NLP, attention has been particularly effective for machine translation. By allowing the network to focus on relevant parts of the input sequence, attention-based models can produce more accurate translations, even for long input sequences. In addition, attention can be used to visualize the parts of the input sequence that are most important for a given prediction, providing greater transparency and interpretability. In the next subsection, we will describe attention in more detail.

2.3.2 Attention in RNNs and Self-attention

Attention was introduced as a mechanism in NLP to improve the performance of sequence-to-sequence models, which were commonly used for tasks such as machine translation, text summarization, and speech recognition. This application was named Bahdanau attention, after its inventor Dzmitry Bahdanau who proposed it in a paper with Kyunghyun Cho and Yoshua Bengio in 2014 [3].

The basic idea behind Bahdanau attention is to allow the model to selectively focus on different parts of the input sequence when it is tasked to generate each output. The mechanic is a solution for the fact that the design of the RNN essentially causes the final encoded hidden state to have to hold all the available information about the input, essentially acting as a bottleneck. To solve this, we collect every encoded hidden state and collect them in a *context* vector, which captures a better representation of every step of the sequence. Then the next step is to weigh the elements of the context vector to figure out which parts of the sequence are the most *relevant* for the current output.

This is done by calculating an attention score, for each encoder state compared to the previous decoder state. It can be achieved in various ways, but the simplest, for example, would be to calculate the dot-product, called **dot-product attention**, where the similarity between the states is considered relevant. Then we normalize with a softmax to create a distribution of the weights, which will tell us proportionally how relevant each element of the sequence is to the current output token. The attention scores are recalculated for each new decoder state, to find the relevancy of the input for each output token.

The Bahdanau attention improves the performance of RNNs dramatically, but is a bandaid to a more fundamental problem of the architecture, being the sequentiality of the

input that relies on the Markov property [15]. Every hidden state inherently holds information from the previous states, and thus does not only represent the current token. This is something another type of attention addresses called self-attention, which does not rely on scoring sequential states but rather compare all elements of a sequence with all elements of another sequence simultaneously.

Self-attention can be described as a fuzzy dictionary matching. We have two vectors consisting of $Keys$ and their corresponding $Values$, and we wish to compare a $Query$ vector to see how well it matches with all the $Keys$, but unlike a regular dictionary, we are not looking for one-to-one matches but rather looking for which members of $Keys$ fit the *best*. When we multiply the $Query$ with the $Keys$, higher values from both vectors will be more prominent while lower values will be less pronounced. We consider this our attention weights, which determines what is considered relevant or not. Multiplying these attention weights with the $Values$ vector tells us how much of each of the values from $Values$ we are interested in.

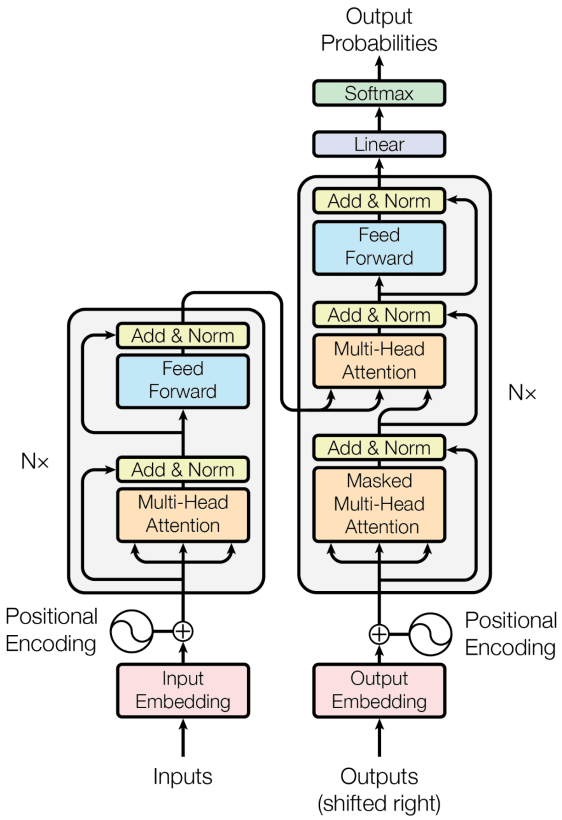

Figure 2.1: The Transformer - model architecture[25].

The Transformer model architecture as shown in Figure 2.1 uses three different self-attention blocks, all of which work a little differently from each other but are all designed around this $Query$, Key, $Values$ vector setup. The encoder block is a Global Self-Attention Layer, where the input sequence is assigned to all three vectors, and is simply compared to itself. This essentially increases the difference between high and low values in the sequence to create a better focus on important tokens and is the part that has ended up superseding the recurrent layer found in RNNs.

The decoder block contains the last two attention modules. The first one is a Causal Self-Attention Layer which is responsible for creating a representation of the output similar to how the encoder's attention layer handles the input. However, the reason why we want this layer to be *casual* (where the output is only dependent on previous sequence elements) is that we do not want the model to know the future parts of the sequence when guessing the next token to generate. Thus, we mask all the tokens after the current timestep as infinitely low values to avoid the look-ahead.

The last attention module is the one combining the previous two representations that have been created by the other attention layers. This layer can be referred to as a Cross Self-Attention Layer as it uses the output representation as the $Query$ vector which is compared to the input representation which serves as both the Key and $Values$ vectors. Thus we get a comparison between the tokens in the input and the output to help us figure out what token makes sense to generate next.

2.3.3 Transformers

Even though the Transformer model used for the experiments in our work is from the PyTorch [7] library for Python, we have also created our own model for didactic purposes. The reasoning is that it is important to be able to verify and understand the methodology of the model we use to experiment, and by creating our own we can know what happens in detail in terms of operations. When testing it, we observed that its accuracy was a few percent worse than PyTorch's implementation, which is the reason why we decided to use the latter for the results section. Our implementation and the PyTorch model are a part of our source code found in the project's repository[1].

[1] `https://github.com/KrisAesoey/AtomicTranslation`

The Transformer [25] is a language model designed to take full advantage of the innovation and improvement that the attention mechanic introduced to the RNNs for NLP, while simultaneously managing to mitigate the drawbacks when using recurrency. Instead of processing the input data sequentially, like RNNs, transformers use the self-attention mechanism to directly process all inputs simultaneously. This allows the model to capture global dependencies between input tokens, rather than just local dependencies that are captured by RNNs. This is done in an encoder-decoder fashion, where the encoder creates a representation for the input, the decoder creates a representation for the output and finally, they are both combined.

Transformers also lack the inherent ability to infer sequentially, and thus need some way to understand positional relations for the input. This can be solved in different ways by using positional encodings that are either fixed using a function or a separate embedding layer, that is combined with the embedding of the input. As such, we can inject order into the embedding for the sequence and the Transformer can learn it. Still, a large problem is that transformers are still heavily dependent on their data to learn and draw inferences. A recent state-of-the-art model ChatGPT [17] from the GPT-3.5 series has garnered a lot of attention in the media due to its seemingly infinite amount of knowledge on any given topic. It is capable of explaining concepts, writing code, and having conversations with the user due to its massive amount of training data.

Despite this, there are a lot of limitations to the model as stated by the authors: "ChatGPT sometimes writes plausible-sounding but incorrect or nonsensical answers." [17]. Language models, like any machine learning model, are prone to perform logical errors because they are based on statistical patterns [8] and associations in the data they are trained on. These models are not able to understand the meaning of the text in the same way that humans do, and they may make mistakes when trying to infer meaning or generate text. Another reason is that the quality of the data can vary, which can contain errors, inconsistencies, and biases. These errors and biases can be inadvertently learned by the model, leading to logical errors in its output. Ultimately, studies have shown that indicate that not even state-of-the-art LMs are capable of performing logical reasoning [28], even though the architecture they are based upon, the Transformer, has been shown to exhibit the capabilities of simulating any reasoning algorithm imaginable [21].

Chapter 3

Language to rule translation

In this chapter, we will look at an existing work that performed a natural language to description logic experiment as a neural machine translation task using a Recurrent Neural Network (RNN) approach. We will specifically take a look at the challenges they faced in terms of creating data for the task, and why they made the choices selected for the final product. Then we will look at our data of choice for the experiments, the Atomic knowledge base (KB), and how we decided to create logical formulas for the natural language *if-then* statements to perform machine translation experiments on.

3.1 DKET - Ontology Learning in The Deep

There exists previous work [19] known as the Deep Knowledge Extraction from Text (DKET) project that intended to test the capabilities of training an RNN model to perform neural machine translation from natural language into description logic (DL). The motivation for such a project is part of a larger ongoing field of researching the possibility of semi-automating the generation of logical rules from natural text. This is because the automated reasoning community has created tools that can perform this quite well, but they are time-consuming to design and produce rules for. Thus, there is a need for large-scale amounts of formulas that can be used in such contexts, which is quite limited right now. So, having a language model that can analyze natural language sentences and capture their elements into logical components would be a great tool for this task.

They designed an RNN model that encodes the natural language sentence and then chooses between two actions to perform when decoding it, either to copy one of the words found in the original sentence or to output a logical symbol from a shorthand vocabulary of the desired output language, which was ALCQ in their case. The model does not care about the semantics of the words but instead looks at each word's position in the sentence to create a syntactic understanding of where the *content* words that define the meaning are located. This approach comes with both some advantages and drawbacks. Due to the fact that the model only treats the words in the sentence as indexes, it allows the model to work well with unknown words as well. On the other hand, the model cannot create formulas that contain words other than the ones found in the sentence, mixed with the logical symbols it has available.

To train such a model they need a dataset of natural language where each sentence has a description logic equivalent that the RNN can learn the relation between. A challenge was the lack of existing appropriate data for this type of task. This led to the need to create new data specific to this task. Since the model was purely focused on the synthetic structure of sentences there was also no need to create data that actually required any semantic reasoning. As long as the sentence structure makes sense the model should be able to capture the relevant words to form the correct description logic ontologies.

3.2 The DKET Dataset

There were some characteristics of the data that were necessary for the model they designed to work in its intended way. The DL ontology that is equivalent in meaning to the natural language sentence has to only contain words from the original sentence, due to the model's capabilities of only either copying words or outputting logical symbols. The result of this was that they restricted themselves to work on a subset of natural language, specifically definitory language; a set of sentences used by humans to characterize a set of entities. This was also motivated by the fact that treating ontology learning as a process from natural language is a massive task, and tackling the entire domain at hand is too large of an undertaking all at once.

The linguistic requirements for sentences to be considered definitory are that they consist of a *definiendum* (the concept being defined), a *genus proximum* (the class or family that the

definiendum is being defined as a part of) and a potential *differentia specifica* that specifies a unique characteristic for the defined concept that is not shared with the rest of its family.

Example:
A human being is an animal that has the capacity to reason.

As seen in the example above the definiendum is "human being", the genus proximum being "animal" and the differentia specifica of "capacity to reason". In this fashion, the input language for their machine translation has been limited from being every possible kind of natural language sentence to only containing those that explicitly define a concept.

A very important distinction about this limitation is that despite it reducing the number of eligible sentences used for the experiments, it does not in fact result in a controlled natural language (CNL). A CNL is a subset of natural language that is created by restricting the grammar and potential vocabulary to make it easier to understand both by human readers and language models. This is generally achieved by using a combination of rules for the writer, such as "Keep sentences as short as possible.", "Do not use different words for the same concept." or "Use the active voice.". [16]

Again, the lack of existing datasets for this task and the restrictions described above meant that the authors had to create synthetic data to test the model. The solution they then came up with for creating a large amount of definitory sentences with corresponding correct DL ontologies was to design a hand-crafted context-free grammar. Using a total of 158 production rules, they were able to generate so-called *sentence templates* that could be a total of 16.5 million different strings with anonymized content words. [20] In other words, a massive syntactic variation of sentences where only the *definiendum*, *genus proximum*, and any *differentia specifica* needed to be filled in from a vocabulary.

With this grammar, they were capable of creating as many necessary examples as they required to create the DKET datasets, by filling in the sentence templates with a catalog of 2841 nouns, 1629 adjectives, and 897 verbs [20]. With this, they had all they needed to test the RNN model's ability to translate between natural language definitory sentences to DL ontologies. The only caveat is that the sentences created do not bear any semantics that can be considered logically sound. The stochastic choice of content words causes definitions of concepts to be defined by information that does not have any real-world appliance. This is very clear from just a few examples:

Inf. dim.	Question	Category
xIntent	What does PersonX **intend** to do?	Mental-State
xReact	How would PersonX typically **react** after this event?	Mental-State
xNeed	Does PersonX **need** to do anything before this event?	Event
xWant	What does PersonX likely **want** to do next after this event?	Event
xEffect	What **effect** would this event typically have on PersonX?	Event
xAttr	What characteristic would you **attribute** to PersonX?	Persona
oReact	How would others typically **react** after this event?	Mental-State
oWant	What would others typically **want** after this event?	Event
oEffect	What **effect** will this event typically have on others?	Event

Table 3.1: The inferential dimensions of Atomic and their corresponding categories

Example:
Kernel summary of trunk forgive also principal of string or of fever.
All civilian of shrimp tune invent more than textbook or parsimonious lung mayor.

However, this is not an issue for the usefulness of the model as it only learns to translate syntactically, and works well with unknown words as well. Thus, training on these meaningless sentences will allow the model to translate other logical definitions later that follow the same syntactic structures.

3.3 The Atomic Dataset

Atomic [23] is a massive atlas consisting of 877k descriptions of inferential knowledge. The KB focuses on the everyday commonsense understanding of rules, organized as typed if-then relations. The data in Atomic was collected using a crowdsourcing framework, where responders are asked to write answers to specific events. In other words, *if* a certain event happened, *then* a question is asked to infer something about that event. This could be what would necessarily precede the event to make it possible, what the event says about the person performing the event, or what could follow as a result of the event. All of these aspects relating to the event, which can be asked questions about, are known in the KB as *inferential dimensions*, listed in Table 3.1.

Example:
PersonX browses PersonY's collection, xAttr, curious

In the example above we see an event, an inferential dimension `xAttr` which represents one of the questions from Table 3.1, and an inference. The event of "PersonX browses PersonY's collection" serves as a situation from which we can ask "if this happened, then what can we assume?". What we want to assume is represented by the inferential dimension and "then" we get a resulting inference that fits the question. As such, we get a complete if-then expression.

There are multiple dimensions where the different if-then relations are split. The first is the matter of who the subject of the inference is. In most cases, we are interested in PersonX, who is always the subject of any given event in the dataset, but in some situations, we would like to know how others are affected by the event as well. To distinguish which one we are looking at in any given case, we use the prefix of either **x** for PersonX or **o** for Others. Who this "other" is can either be explicitly mentioned in the form of a PersonY or be an implied participant. For example, an event such as "PersonX calls for help", implies that there is someone who will answer the call, which is very contrary to most previous work of this type of if-then reasoning [23]. Nonetheless, in this way, it is easy to know who the inference is talking about at a simple glance.

The second way of splitting up the inferential dimensions is by which type of content is being predicted. The authors defined three different categories that each inferential dimension is exclusively part of: mental state, event, or persona.

Mental State The mental state category consists of relations connected to the mental pre- and post-conditions of an event. Here we look at likely intents as well as possible reactions of both the subject and others. An example event would be "PersonX pays PersonY a compliment" with an inference in the xReact dimension being "PersonX will feel good".

Event In the event category, we note events that are likely to precede and follow the given events. This includes pre-conditional events that can be inferred as necessary to be able to perform the given event, as well as both voluntary and involuntary post-conditions for both the subject as well as other (potentially implied) participants. An example event could be "PersonX kills PersonY's father" and an inference in the xEffect dimension could be "Goes to prison".

Persona The final category is persona which consists of inferences on how we can describe or perceive the subject in relation to a given event. In other words, we are trying to characterize the subject based on its actions in the isolated event. If an event was "PersonX makes PersonY's coffee" then a natural inference in this category could be "PersonX is helpful". Why specifically the word persona is used for this category is not explained, but can be inferred to be of the Jungian theory [11] that the persona is a mask or appearance one presents to the world. We cannot know the subject's true nature based on the isolated instance, but can instead infer how they appear as a result of it.

In addition to being asked to answer questions about short events describing an isolated situation, there are also a lot of incomplete events described in the datasets. On a syntactic level, they are practically identical to the complete events, except that a single word has been replaced with a blank placeholder.

Example:
PersonX plays ___ at school

A very important aspect of the knowledge base is the nature of how the inferences were collected. Using a crowdsourcing framework, workers were asked to answer questions such as the ones shown in Table 3.1 about events provided by the authors. The answers given using a small textbox were then connected as an if-then statement through the dimension of the question. Thus, the syntactic nature of the inferences has a huge variance depending on how every single responder would individually choose to answer the question.

There has also been made further work to improve and expand upon the Atomic KB focusing on improving the number of relations available to increase coverage of everyday commonsense situations. Recent research on the topic [10] features a new dataset known as $Atomic^{20}_{20}$ where they kept all the relations from the original Atomic as a subset, and also introduced two whole new categories.

The relations we have used that could be split into mental-state, event, and persona were collectively categorized as Social-Interaciton relations. The first of the new ones was *Physical-Entity* describing inferential knowledge about common entities and objects, such as capabilities, desires, or spatial properties. This type of inference was captured by relation

types such as *ObjectUse*, *CapableOf*, or *AtLocation*. The other new category was Event-Centered, which describes common events that are related to one another. It has some overlap with relations such as *xEffect* from the original Social-Interaction subset, but the events here are generally describing conditions that are outside human control. A relation such as *Causes* that exists in this category would take an event such as "bad weather" and infer postconditions such as "power outages" which is not an event caused by either PersonX or any other person in general. Other examples of relations found in the category are *isBefore* and *HinderedBy*.

A noteworthy aspect of the new relations introduced is the lack of representation some of them have. The earlier mentioned *Causes* relation has less than 400 instances, which is a small portion compared to the total of 1.33M inferences. This was not an issue in the original Atomic, where each relation had a great representation, as the difference between the most common (xAttr with 148k) was only twice the least common (oReact with 67k examples). In the $Atomic^{20}_{20}$ KB this number skyrockets as the most common (ObjectUse with 165k) is almost 500x more than the least common (xReason with 334).

Overall, the introduction of these new relations does not in any way make the work performed in this thesis obsolete. The original set of relations now referred to as the Social-Interaction subset is still the same, and has not been replaced with new ones in the new dataset. As such, the work done to use neural machine translation to learn logical formulas from them is still applicable, and the new KB only offers more possibilities to expand on the idea in the future.

3.4 Creating logical formulas for Atomic

Naturally, this leads us to how we can create logical formulas that fit the syntax and potential semantics found in the Atomic KB, to have a similar natural language to logic pairings such as seen in the DKET datasets. The first idea on how to do this would be to simply mimic the approach used to create the DKET datasets. For this, we would need to create our own hand-crafted grammar that generates *if-then* relations of similar syntactic structures found in the Atomic KB.

This would be an incredibly time-consuming task, and frankly impossible in terms of capturing the inferences found in Atomic due to their crowdsourced nature causing them to not follow any consistent syntactic structure. A solution to solve this would be to select the most common event and inference structures that we find in Atomic and ignore the rare and unique syntaxes. This would work very well for the *Persona* category, where we find the most predictability in terms of inferences for the `xAttr` dimension.

Example:
PersonX paints PersonX's portrait, xAttr, creative
PersonX publishes PersonY's work, xAttr, proud

In the examples above we see the most common event structures. What makes these examples easy to create templates for is the fact that the `xAttr` dimension asks for an attribute of PersonX to be inferred. This is most often simply stated as a singular adjective, which means that we could capture most of these if-then relations with a singular template:

Example:
PersonX VB NNP's NN, xAttr, JJ

Here we have anonymized the verb, individual, and noun as well as the inferred adjective to describe PersonX. In addition to this, we could also create more variations for this event template with other inference syntaxes where we see modals or "to" included. Creating a simple algorithm that replaces these part-of-speech tags with a fitting word from a catalog would allow us to quickly create a massive amount of examples for any created template. Of course, the issue is that the general if-then relation is not as straightforward as the ones found in the Persona category. In the *Mental-State* and *Event* categories we see similar types of event, but with a much more unpredictable variation of inferences in terms of syntax.

Example:
xEffect: PersonX sails PersonX's boat → it capsizes
xNeed: PersonX studies as much → sign up for class
xWant: PersonX eats oatmeal → to put the dish in the sink

As we can see from just a few examples above, the inferences are very different for these other types of dimensions. This is strictly a *feature* of the Atomic KB, due to the collection of inferences being crowdsourced. The authors want the responses to capture the freeform nature of language, but this also makes it harder to create logical structures from it. This approach of creating templates that *resemble* the Atomic if-then relations comes with a ton of necessary decisions on what inference syntaxes to use, which to omit, and how to balance out the distribution of them to make it appear as much as Atomic as possible.

No matter what choices are made, we compromise the Atomic KB in some way and end up creating a pastiche instead of using the actual dataset. The end result might *look* like Atomic, but it would not *be* it. In addition to this, we are also losing out on the fact that Atomic contains actual typed commonsense if-then relations that a model could potentially learn semantic understanding from. This was one of the key factors that the authors of DKET suggested as a future possibility for improved translations, as a purely syntactic model would lack the understanding of the words to perform substitutions from the original sentence [20].

Their example was the idea of two definitory sentences being expressed in almost the exact way, that conveys the same meaning:

Example:
A bee produces honey $\rightarrow$ Bee $\sqsubseteq \exists$produce.(honey)
A bee makes honey $\rightarrow$ Bee $\sqsubseteq \exists$make.(honey)

Both examples are correct, but one could argue that the "produce" is more concrete for this specific case, and therefore more desirable for a logical formula than the generic "make". If a model is capable of semantic understanding, it could potentially learn that bees are insects, and that honey is an artisan product, and could then be trained to understand the relation between bees and honey similarly to how for example cows and milk are related. Then, the resulting formula for "A bee makes honey" could replace make with produce, which could not be possible using a purely synthetic approach.

Thus, if we wish to be able to utilize a model's ability to learn semantically from the data in Atomic, utilize the full range of syntactic structures found in Atomic, as well as not have to create the logical formulas by hand then there is only a single solution. Conceive an algorithm that analyzes the natural language if-then relations of Atomic and creates equivalent formulas in first-order logic to construct our dataset.

3.5 Creating logical rules with Atomic

If it was possible to create an algorithm that could perfectly look at any natural language sentence and create a perfect first-order logic representation of it, then we would not need the work in this thesis. Our goal is not to create a perfect one, but one that is good enough to allow us to properly test our model's capability to translate Atomic if-then relations into logical rules.

The strategy we opt for is to deconstruct the natural language sentences into the natural parts that occur in any given English sentence, its subject, verb part, and object part. We exploit that Atomic events consist of an independent clause and that compound or complex sentences only occur in some inferences. This, combined with the fact that every event starts with the subject "PersonX" creates simple patterns that we can model into first-order logic.

3.5.1 Removal of PersonZ

The Atomic KB concerns itself with the individual PersonX, often how it would be affected by events but also sometimes in relation to others. In most cases the other is the individual "PersonY", and sometimes it is simply implied. Though, in a few cases a "PersonZ" is also included, when PersonX is related to more than one other individual for the event. However, they are exceptionally rare and introduce weird situations that are not easy to model.

Example:
PersonX takes PersonY in PersonZ arms.
PersonX puts PersonY end to PersonZ.
PersonX invites PersonZ's friend PersonY.

As seen in the examples above, it is not always very clear what the intention is when PersonZ is involved. It also creates an even stronger uncertainty in terms of which individual the inference references when we ask about how the event relates to others. In most cases, we already need to create an assumption that when PersonY is involved that they are considered the "other", but with an additional individual this becomes even more dubious. Thus, we have decided to remove all events where PersonZ appear, which were a total of less than 1000 total.

3.5.2 Natural Language Toolkit part-of-speech tagging

The major feature of the sentences that we use to create logical formulas is the use of part-of-speech (POS) tagging to find atoms in the natural language. To perform this POS task, we use an existing Python library Natural Language Toolkit, colloquially known as NLTK [4] that has an existing function `nltk.pos_tag(tokens)` that can do it for us. We simply feed it a natural language sentence as a list of all the words as strings, and it returns a list with tuples consisting of the word and its corresponding POS tag. The choice of NLTK to perform this task was due to familiarity gained from previous academic instruction.

This is somewhat effective but at the same is a massive constraint as it means that we are also prone to face all the challenges that the POS tool has to deal with. A very common issue seems to be the tool's ability to distinguish between the word classes of `VBZ` which indicates a verb and `NNS` which indicates plural nouns. Specifically, it sometimes mistakes homonyms where a verb conjugated in the third person singular is the same as the plural of a noun with similar roots. An example from the Atomic dataset:

Example:
if-then expression
PersonX paints PersonX's portrait, xWant, to hang the painting
POS-tagged
PersonX/NNP paints/NNS PersonX's/NNP portrait/NN, xWant, to/TO hang/BV the/DT painting/NN

Here we see that the verb of the sentence "paints" has mistakingly been identified as a noun. This could be a difficulty that arises when working with Atomic data due to the fact that individuals that are non-existent in the English language have been introduced. PersonX and PersonY show up in all or many of the samples and are not actual English words, which causes the POS tagger to work with unknown words to grasp the meaning of the sentence. However, we can observe that the POS tagger has accurately captured the individual in the class of `NNP` which is the identifier for a proper noun, which PersonX is. Despite this, it is also fair to proclaim that the sentence is not completely natural, as most people don't refer to their proper name when stating possession such as "PersonX's work" would rather be stated as "their work" when the gender is not specified. Nonetheless, it

is not grammatically incorrect, and by following the natural sentence structure of subject-verb-object that English follows the verb should still have been inferred. The result of these issues is that using POS tagging as a strategy for identifying FOL atoms and conjuncts comes with caveats related to the strength of the tools available. An improvement in the field of POS tagging could therefore also improve on such strategies. On the other hand, what it explicitly means for the work done in this thesis is that we cannot reliably use the identification of verbs as means of finding specific atoms in the sentences.

The events described in Atomic always start with the subject PersonX followed by the verb part of the sentence. There are a varying amount of supporting words that can be a part of the verb part, such as to, modals, or adverbs, but lastly, there is always a verb included hence the namesake. Looking for the verb to indicate the conclusion of this part of the sentence is then not feasible if the POS tagger does not reliably tag it correctly. So, in our algorithm, we need a slightly more convoluted way of capturing it. In addition to this, we perform a slight postprocessing altercation to the POS tagging, by changing the tag of the individuals PerxonX and PersonY from `NNP` to `IND` for "individual". This is done so that we can easily track which specific characters are involved in the sentence, and change the appropriate arity of some atoms as a result of it.

3.5.3 Gramatically correct inputs in Atomics

One of the fundamental features of the Atomic KB is the crowd-sourced nature of the inferences in the *typed if-then* relations. This allows the responder to express themselves however they want, and thus more accurately reflects the natural language of responses than simply restricting the user to only use certain words or sentence lengths. Inherently, this is not an issue, but there is an overall issue with what seems to be a lack of postprocessing performed on the inferences collected. In the original work [23] there is no indication of such a process, nor can it be observed from working with the data as we find certain elements expressed in many ways.

A difficulty arises if you wish to capture, or analyze every instance of the individuals involved in the sentences. In the events provided by the framework, we find that they are always referred to as PersonX and PersonY with these exact spellings. This is far from the case when we look at the inferences, where the respondees have found a multitude of ways to refer to them.

Example:
PersonX builds PersonX's houses. → **X's** family has a new home to live in.
PersonX replaces PersonY's tire. → to pay **person x**.
PersonX provides PersonY description. → to help **him**.
PersonX worships the ground PersonY walks on. → gets nervous from **peronsx's** actions.

Observing the few examples above we can immediately see that the issue is encountered in many different ways. In the first case, we notice that the "Person" part of "PersonX" has been omitted by the responder, which can be considered quite natural as we humans still infer quite easily which individual we are referring to as the distinctive part is still mentioned. We can see another example of the same issue handled differently in the second inference, where instead the "Person" and "X" parts have been separated by a space. Again, quite a natural way of expressing anonymous individuals seen in other literature and it still stays unambiguous to who they refer to. In both of these cases, the problem boils down to pattern matching, as we can still capture them as instances of "PersonX" by looking for instances where "X" is isolated and where it follows after the word "person" in the inferences. They merely introduce the necessity to perform more exhaustive searches through more patterns to look for.

The other two examples on the other hand create different, less easily solvable issues. In the third example, we can see that the gender of "PersonY" has been assumed and is just referred to as "him". This introduces the problem of us having to infer which individual these terms are assigned to in each instance individually. In this example, it is quite clear, but there is no guarantee that we can correctly identify who a pronoun is describing in all inferences. Especially, in the types where we ask how it relates to *others* in a given event such as in `oEffect` and `oReact` seen in Table 3.1. Atomic refers sometimes to *implicit* individuals involved in an event[23], which means that sometimes we also need to make assumptions about if anyone referred to ambiguously in the inference is one of the named individuals of the event or a potential third party. In addition, we wish to automate the process of creating the formula for each if-then relation, and would therefore somehow find some consistent way of handling them. Thus, we decided to consider who the pronouns describe to be determined as a result of the appearance of PersonX and PersonY in the event and what type of inferential dimension we are dealing with.

The last example is the most obvious evidence of lack of postprocessing, and therefore also the most frustrating. In this case, we observe that PersonX has actually been misspelled by the responder. This is not something that is exclusive to the individuals either, but a phenomenon that appears throughout the Atomic KB where we can find both spelling errors as well a lack of spacing that combines two words that should be separate. This could have a dramatic impact on learning semantic understanding, where the model learns to memorize certain if-then relations where a misspelled word appears once instead of generalizing the words in inference with other inferences where the word has been spelled correctly.

To solve this issue the only solution would be to either have a spell-checking algorithm that goes through all the relations and find any misspelled words, or to manually go through each one. In both cases, there is uncertainty if it would solve the issue entirely depending on how sophisticated the correcter is, in some cases, words can transform into a different word of the same POS class if a letter is missing or not such as "winds" and "wins". The one who corrects the inferences would then also need to infer the contexts of the words and clearly understand the underlying meaning of the incorrect inference. We do include some measures for this, where we check for spellings that are super close to person such as "perons" and "pernos" that have an edit distance of one from a correctly spelled individual and correct them.

This would not actually solve all the issues, as there are even some examples where the events contain some type of error, that does not look like grammatical errors at all.

Example:
PersonX hats dogs, xAttr, hate
PersonX hats dogs, xAttr, dislike
PersonX hats dogs, xAttr, unhappy
PersonX hats dogs, xAttr, avoids dogs

The event given "PersonX hats dogs" is a grammatically correct sentence, despite being an unconventional way of expressing the idea of applying headgear on dogs. However, when looking at the inferences from this in the `xAttr` dimension where the responder is asked to infer an attribute about PersonX from the event we see some inconsistency. The inference lacks any concrete connection with the event, as all of them express a disdain PersonX has for dogs instead of relating to the concept of hatting them. It makes perfect sense though if

you infer that the event *should* be "PersonX **hates** dogs". Therefore we can assume some kind of spelling error has happened between the event given in the crowdsourcing framework and what ended up being collected in the KB. Thus, no spelling checker could potentially find this inconsistency. There is also no guarantee that this was not intentional, or that it is a rare case where all the responders misread the prompt they answered.

No matter what measurements we would take to correct this, we would run into the issue of ruining the *integrity* of the dataset in the process. The data is as is, and if the responders who drew the inferences are drawing them mistakenly or in a nonsensical fashion, it is still true to the human nature of interpreting what is natural for the if-then relation. If we perform any changes to "correct" them beyond what we can concretely guarantee would not be altering the intentions of the responder, then we could instead ultimately undermine this concept. Due to this, and the frame of time of the thesis restricting further work into these concepts, we have decided to leave any spelling mistakes and dubious inferences as is. We believe that the creators should be responsible for the quality of the data they provide, and it is simply in our best interest to use what we have as best as possible. This does of course impact the logical formulas, where one can in some instances clearly observe their lesser quality, due to the algorithm not being designed to handle these issues. Nonetheless, it still creates atoms that are apt and that the language model can learn to translate.

3.5.4 The algorithm

The algorithm works on each if-then statement individually. Firstly, it splits it into three parts, the event, the inferential dimension, and the inference. The atoms are done for the body first, as they are completely independent, while the information in the head relies on both the event and the inferential dimension. The event-to-body transformation is performed by Algorithm 1, the inference-to-head transformation is done by Algorithm 2 and they are both called upon by Algorithm 4 to construct the logical rule.

In the event, we try to capture three separate types of atoms, the individuals, the verb expression, and the object expression (if it exists). To do this we first identify all the words that have the `IND` tag in the event, which represents the individuals, and remove them from the event. Afterward, we try to separate the verb expression and the potential object. The sentences that describe the events are always in the format of PersonX doing an action,

either to someone or something, or just an act in itself. So, to separate them we simply have to look for when the verb part ends, and everything following will then be an object expression. Due to the issue with the POS-tagger sometime identifying the verb as a noun, we have to make sure we capture at least a single word as a part of the verb expression before we start looking for the words that express a potential object. To find this split, we look for adjectives and nouns, which mark a potential object atom in the sentence. Once the entire sentence has been traversed, we combine the atoms with the appropriate variables, to complete the logic of the body.

Algorithm 1: Event to Body

input: List of pairs (w, t) called *event*, where w is a word and t is a POS-tag.
output: The body of the Atomic rule and a list of variables found in the body.

```
1  verb = obj = body = variables := ∅
2  verb_finished := False
3  Remove (PersonX, IND) and (PersonY, IND) (if it exists) from the event list,
     and add them to body.
4  Add x, z to variables and also y, if PersonY occurs in event.
5  for (w, t) ∈ event do
6      if verb ≠ ∅ and t ∈ [JJ, NN, NNS] then
7          verb_finished := True
8      if verb_finished then
9          Add w to obj.
10     else
11         Add w to verb.
12 Let v be the concatenation of the words in verb.
13 if (PersonY, IND) ∈ event then
14     Add v(x, z, y) to body.
15 else
16     Add v(x, z) to body.
17 if obj ≠ ∅ then
18     Let o be the concatenation of the words in obj and add o(z) to body.
19 return (body, variables)
```

Algorithm 2: Inference to Head

input: List of pairs (w, t) called $inference$, where w is a word and t is a POS-tag, the *body* as a list of atoms and a word that is the $inferential\_dimension$.

output: The head of the Atomic rule and a list of variables found in the head.

```
1  verb = cur_obj = objects = head = variables := ∅
2  verb_finished := False
3  Remove PersonX and PersonY from the inference, and add them to head if they
     do not occur in the body.
4  Add x and y to variables if PersonX and PersonY appear in the inference but
     are not already in the body.
5  for (w, t) ∈ inference do
6      if verb ≠ ∅ and t ∈ [CC, DT, PRP, PRP$] then
7          if verb_finished = True then
8              Add cur_obj to objects.
9              Add variable for cur_obj to variables.
10             cur_obj := ∅.
11         else
12             verb_finished := True
13     if verb_finished then
14         Add w to cur_obj.
15     else
16         Add w to verb.
17 if cur_obj ≠ ∅ then
18     Add cur_obj to head.
19     Add variable for cur_obj to variables.
20 if inferential_dimesion = PersonX then
21     subject := x
22     if PersonY occurs in body or head then
23         target := y
24     else
25         target := none
26 else
27     if PersonY occurs in body or head then
28         subject := y
29     else
30         subject := u
31     target := x
```

```
Let v be the concatenation of the words in verb.
if objects ≠ ∅ then
    if |objects| = 1 and equals the object atom in the body then
        if target ≠ none then
            Add v(subject, z, target) to head.
        else
            Add v(subject, z) to head.
    else
        for obj ∈ objects do
            Let o_var be the variable for obj in variables.
            Let o be the concatenation of the words in obj.
            if target ≠ none then
                Add v(subject, o_var, target) to head.
            else
                Add v(subject, o_var) to head.
            Add o(o_var) to head.
else
    if target ≠ none then
        Add v(subject, target) to head.
    else
        Add v(subject) to head.
return (head, variables)
```

Example:
PersonX paints PersonX's portrait $\Rightarrow Person(x) \wedge Paints(x, z) \wedge Portrait(z)$

In this example, we can see how the individual PersonX, the action of painting, and the object being painted have been split into separate atoms. Despite appearing twice, PersonX is represented as the variable x, and its connection to the painting is instead found as a term for the predicate of $Paints$.

The strategy for finding the atoms in the inference is very similar to the one used for the event. Still, the big difference is that we might encounter more than a single object

due to the more freeform structure provided by the respondees. In addition to this, we also have to account for the fact that some inferential dimensions do not concern themselves with PersonX, but rather how *others* are affected by the event. This affects which variables become connected to the atoms.

Example:
xWant To hang the painting $\Rightarrow ToHang(x, a) \wedge Painting(a)$

As seen in the example above, the subject of the action performed in the inference is PersonX, as he is our subject as stated by the inferential dimension of `xWant` where the x represents that we are interested in PersonX's wants. A thing to note here is the fact that the algorithm uses a new variable to represent the painting. This might seem unnatural, as one can infer that the portrait from the body is the same as the painting in the head, and should therefore share the same variable. The issue is that two different words have been used to describe the same object, as the algorithm will detect this type of callback, but *only* if they reuse the same word. Otherwise, there is a certain ambiguity if this is the intended implication from the responder, and thus we consider them different entities for the rule.

A thing to note about the splitting of objects is, amongst others, looking for the POS-tag `CC` that consists of conjunctions. In some rare cases, we observe that the word "or" shows up, and as all the atoms are conjuncts in the rule, it means that the or is treated as an and logically. In most cases, this is functionally equivalent.

Example:
PersonX picks last, xNeed, know who **or** what to pick
PersonX is really sad, xWant, to speak with a friend **or** family member
PersonX makes PersonX's decisions, xWant, to complete the task **or** process

The intention is that both are equivalent, possible outcomes and therefore saying that both happen does not completely alter the meaning. Therefore, despite there being a few cases where this does not apply, we do not treat or differently from and, which keeps the algorithm a lot simpler but also slightly more incorrect.

Finally, after the event and the inference has been transformed into the body and head of a rule, and we have collected all variables involved, we then add the quantifiers or omit

them depending on the desired output. All of the body's variables are universally quantified, while the ones exclusively appearing in the head are existentially quantified.

Algorithm 3: Atomic to Rule

input: Two lists of pairs (w, t) called *event* and $inference$, where w is a word and t is a POS-tag,
a word that is the $inferential\_dimension$, and a boolean $add\_quantifiers$ to decide whether to quantify the variables or not.

output: A FOL-rule for the Atomic if-then statement.

$body, body\_vars := EventToBody(event)$
$head, head\_vars := InferenceToHead(inference, body, inferential\_dimension)$
if $add\_quantifiers$ **then**
    $rule := \forall body\_vars(body \rightarrow \exists head\_vars(head))$
else
    $rule := body \rightarrow head$
return $rule$

Example:
If-then statement:
PersonX paints PersonX's portrait, xWant, to hang the painting
Rule:
$\forall x, z(Person(x) \wedge Paints(x, z) \wedge Portrait(z) \rightarrow \exists a(ToHang(x, a) \wedge Painting(a)))$

At last, we get a complete rule from the if-then statement. Despite the lack of connection between the portrait and the painting, we still get a set of atoms and variables that connect to each other logically in a syntax structure that makes sense. So, even though there are a few statements where it is less than perfect, it works excellently on most of Atomic's if-then statements and gives us a solid dataset that the Transformer can train on.

Chapter 4

Experiments

In this chapter, we will use the previously described related work and our created dataset for Atomic to perform experiments on neural machine translation using the Transformer language model. We will discuss the specific model settings, including their motivation and reflections on possible extensions. After this, we will present how the datasets were assembled and how their sizes relate back to the related work using the RNN approach[20]. Then we will also present what evaluation metrics were used, how they work, and what information they convey about the result. Finally, we will present the results which are in themselves split into four. Firstly, we compare the results of the previous work [19] using the RNN approach achieved on their Deep Knowledge Extraction from Text (DKET) datasets compared to our transformer model. Then we perform experiments on the Atomic dataset using similar dataset sizes as the DKET to see how well the model performs with little data samples and asses the difficulty of the data in relation to the DKET datasets. Afterward, we also run some similar experiments on the Atomic dataset, but where the quantification element of the logical formulas have been removed, to examine if it increases the ability to capture concepts and variables on shorter sequence lengths. Lastly, we allow the Transformer to train with all the data in Atomic, in a natural 85/15 training and testing split to see how well the model performs when allowed to learn as much as possible from the dataset. We do this for both types of datasets, the regular ones and the ones where the quantification has been omitted to see if the models level when they have a lot more data to train on, or rather if the difference becomes even more noticeable. If you wish to reproduce our experiments, then you can find the code in our Github[1] repository.

[1] `https://github.com/KrisAesoey/AtomicTranslation`

4.1 Model settings

Setting	Value
embedding size	512
attention heads	8
encoder layers	6
decoder layers	6
feedforward dimensions	2048
dropout	0.1
batch size	128
sequence length	50
epochs	30
warmup steps	4000
optimizer	Adam
learning rate	$\text{embedding size}^{-0.5} * \min(\text{step}^{-0.5}, \text{step} * \text{warmup steps}^{-1.5})$
label smoothing	0.1

Table 4.1: Settings used for the Transformer models used in the experiments

We treat the natural language to logical formula translation as a generic neural machine translation task, making it natural to use a standard model for this task. In the related work [20] we saw a specialized RNN model was designed for the task, as removing any form of semantic understanding of the sentence to enhance the model's comprehension of its syntactic structure. We, on the other hand, want to use the Transformer language model for what it is, with semantic comprehension and all, to examine how this type of attention-based model *generally* can perform this task.

The transformer models used for the experiments are modeled to use the default settings of the prebuilt Transformer model which is a part of the PyTorch framework for machine learning in Python. The transformer models used for the experiments have the default settings of the prebuilt PyTorch Transformer model [7]. The default settings are the same as the settings used in the original experiments when the model was introduced [25], which is the work we based our training loop upon. Their experiment settings and ours are the same in terms of the learning rate, optimizer, label smoothing, and dropout which proved to be very effective early on in testing. Due to the time restraints for the thesis work, we did not perform a grid search to test different settings and training setups in extensive detail. The results might therefore have the potential to be even better, or worse when some of the setup elements are changed. We consider at this as potential improvements and possibilities

for future work in addition to creating Transformer models that are even more specialized for this type of machine translation task.

4.2 DKET and Atomic datasets

As previously mentioned, we use the DKET datasets that were made for an RNN model to translate from definitory language into DL ontologies [19] that inspired this thesis. We use them as a point of comparison for how well the Transformer model can translate using its semantic approach in relation to the syntactic RNN approach. The data is located on the project's Github repository[2], where we acquired them to use in our experiments by testing the Transformer's ability to translate them correctly. They come in four different sizes: 2k, 5k, 10k, and 20k training samples and all of them come with their own validation set of 30k samples. The RNN model's ability to translate showed great differences using just these dataset sizes which were something we would like to check if is the case for the Transformer on the Atomic dataset.

We want to test Atomic in a similar fashion, where each set was tested with similar dataset sizes as the DKET ones. Therefore, in every experiment, we run four models, where each of them is trained on random samples equal to the sizes referenced above, and 30k other examples are also chosen randomly to validate the results. In addition to this, we also test out the separate categories of the Atomic set: event, mental state, and persona. This is to check if certain types of inferences are easier to learn than others such as persona for example having less syntactic variation in its inferences and are always about the subject of the event, which is not the case with the other two. This will give us more insight into what might cause challenges for the model in the translation, rather than just looking at the overall result for everything combined.

There was also the additional wish to perform a reverse experiment, where we tested the RNN model's ability to translate the Atomic datasets as well, which could provide additional insight into comparing the translation difficulty of the two types of datasets. This was another motivating factor for the choice of using similarly sized datasets. However, due to the outdated nature of the DKET code and the difficulty of setting up an environment

[2] `https://github.com/dkmfbk/dket`

where we could run the model, this was not feasible. The only choice then is to remake the functionality of the model using the methodology described in its original paper [20] using updated compatible versions of the framework, but we did not find it a priority due to the time available and lack of guarantees that a new model will act in the same way as the original. Thus, this part of the experiment was scrapped but is still a considerable candidate for possible future work.

4.3 Evaluation metrics

Three main criteria were chosen to check the correctness of the translations from natural language into description logic. These are the same as the ones used to evaluate the results of the RNN approach [20] created to translate from synthetic definitory sentences [19] to ALCQ, to be able to compare the results from our experiments with previous work. This also allowed us to run the datasets from their work on our model for a direct comparison with the RNN approach and our dataset separately. Every criteria's goal is to compare how well a given prediction $\hat{f}$ from the set of predicted formulas $\hat{\mathcal{F}}$ matches the correct formula f from the set of golden truths $\mathcal{F}$, divided by the total amount M of formulas in the validation set to get an average.

Using BLEU [18] as a metric is also an extremely common way of scoring machine translation, but we found it to be not so applicable in our case. In our experiments, there is a golden truth that we wish to achieve a literal perfect match of. This means that there is no room for alternative translations to a given sentence, and the strengths of its function are diminished compared to its great use case in evaluating natural language to natural language translations. The quality of translation shown by a BLEU score on any given training setup will also be reflected in a more nuanced and telling way by the other metrics choices in our setting.

Average per-formula accuracy (FA) is our first and most intuitive metric. FA is defined as the percentage of correctly predicted formulas related to the total number of formulas in the validation set.

$$\mathrm{FA}(\hat{\mathcal{F}}, \mathcal{F}) = \frac{\sum_{k=1}^{M} \begin{cases} 1, \text{if} f^k \equiv \hat{f}^k \\ 0, \text{otherwise} \end{cases}}{M} [20]$$

A higher percentage of FA directly corresponds to how accurate the model is and the ultimate goal of the model is to achieve the highest FA possible, while the other criteria are used to figure out how well the model performs despite sometimes being incorrect, as there are varying levels of how incorrect a predicted sequence can be.

Average edit distance (ED) is our second metric. Edit distance is an algorithm that tries to capture how far away a given prediction is from the correct answer, as a result of performing operations to transform the prediction into the golden truth. These actions as well as their corresponding cost to perform them can vary from algorithm to algorithm, but as previously stated we are using the same metrics as previous work which specifically uses the Levenshtein Distance (δ) algorithm. The actions are then to add a token, remove a token or substitute a token which all have a cost of one. A low value directly correspondent with good translations, as it intuitively means that a machine or a human would need to perform fewer operations on the output to get the desired result. Thus, the optimal value of ED is 0, in other words, a perfect match that requires no actions to be performed.

$$\mathrm{ED}(\hat{\mathcal{F}}, \mathcal{F}) = \frac{\sum_{k=1}^{M} \delta(f^k, \hat{f}^k)}{M} [20]$$

Average per-token accuracy (TA) is the final metric that gives us an overall look at how well the model can translate the individual tokens. It is very similar to FA, but by comparing the prediction f to the correct formula $\hat{f}$ on a token-to-token basis and dividing the result by the total of tokens T in the formula we can observe if the model is translating well, despite the total prediction is incorrect. If a model shows a high TA in contrast to a low FA, it tells us that the model is rather close, but has a single mistake or two in the sequence that creates the wrong result. Thus we can differentiate between models that do not perform well in varying levels of incorrectness despite showing similar FA scores.

$$\mathrm{FA}(\hat{\mathcal{F}}, \mathcal{F}) = \frac{\sum_{k=1}^{M} \sum_{j=1}^{T_{f^k}} \begin{cases} 1, \text{if} f_j^k \equiv \hat{f}_j^k \\ 0, \text{otherwise} \end{cases}}{\sum_{k=1}^{M} T_{f^k}} [20]$$

We can see how each of these evaluation metrics would score an example translation.

Example:
if-then statement: PersonX tries to help PersonY, xAttr, caring
correct rule:
$\forall$ x y z ((person (x) & person (y) & tries to help (x,z,y)) $\rightarrow$ caring (x))
predicted rule:
$\forall$ x y z ( ) person (x) & person (y) & tries help (x,z,y)) $\rightarrow$ caring (x))

In the example above, we have underlined the differences between the correct rule for the if-then statement as provided by the algorithm, and the predicted rule from the model. All our metrics are averages, but as we only have a single translation, i.e. $M = 1$, it means that the score for this example is equivalent to the overall score. Since there is a difference between the correct rule and the predicted one, it means that the FA score is 0. The incorrect tokens are a missing "to", and a parenthesis that has been flipped. To fix these, we would need to perform one addition of a token, and a substitution of another, and since each action has a cost of 1 we end up with an ED score of 2 for this translation. Finally, for the TA score, we count how many tokens at the same index are correct. Since we are missing a to in the prediction, it results in every following token being incorrect due to appearing earlier than they should. Therefore we only achieve a TA of 0.6, i.e. 60% accuracy, despite only two tokens being technically incorrect. This demonstrates how each metric gives us different information about the translation.

4.4 Results

In this section, we will look at the results of the four different types of experiments we perform. First, we will look at how the Transformer compares to DKET's [20] RNN approach in translating their datasets. Then, we investigate how well the model translates small datasets from Atomic, and how each category performed individually to see if certain parts of the data are easier than others. After this, we omit the quantification of variables from the rules, to observe if the accuracy increases on shorter, less complicated sequences. Finally, we go all out and let the model train on the entire dataset, to see if the model achieves even higher performance on.

4.4.1 DKET using RNN and Transformer

Model	FA	ED	TA
RNN-2k	61.1%	2.48	91.8%
TF-2k	0.0%	10.2	42.6%
RNN-5k	84.4%	0.6	97.5%
TF-5k	0.0%	9.25	51.3%
RNN-10k	88.8%	0.47	98.7 %
TF-10k	99.8%	0.007	99.9%
RNN-20k	81.7%	0.46	98.3%
TF-20k	**99.9%**	**0.000067**	**99.9%**

Table 4.2: Results comparing the RNN approach and Transformer (TF) on the DKET [20] datasets.

In Table 4.2 we can see the results that were achieved in the reference work when using their RNN approach [20] compared to our Transformer model on the same datasets. The first thing to note is that the RNN approach achieves much better results when training with very few samples. It achieves a FA score of over 60% with only 2k training examples and improves rapidly towards its peak accuracies when training with 5k. Also, the TA score is over 90% for all experiments, showing that it quickly picks up on the syntactic structure of the data. In fact, the RNN approach learns the structure much faster than the Transformer does, as we can see that the Transformer performs terribly when training on 2k as well as 5k examples. In neither case the model actually manages to correctly predict a full formula, meaning that the overall FA score is 0%. In addition to this, the TA scores are around $40 - 50\%$ meaning that the predictions are not even close, having to replace over 9 tokens to be correct on average.

However, when the Transformer model gets enough data samples to train on, we see an incredible jump in performance. While the RNN approach improves with $\sim 4\%$ between 5k and 10k examples, we see a literal jump of over 99% for the Transformer. Suddenly, the Transformer outperforms the RNN's best performance and is translating the data almost perfectly. In fact, at its best when training on 20k examples it only missed a total of two translations out of the entire set of 30k. This shows that the syntactic structure of the DKET datasets is easier for the general semantically aware Transformer model to capture and translate than for the specified syntactic RNN approach when there are enough training examples to learn from.

If there is sparse data consisting of natural language sentences with equivalent ontologies in a target Description Logic language then we can use the RNN approach to achieve decent translations that are either correct or very close to being. On the other hand, if we have enough data, we can observe that the Transformer model is more than capable of producing correct ontologies that consist of the same words as the natural language in combination with a dozen logical symbols to capture the correct meaning.

4.4.2 Atomic on small datasets

Training Dataset	FA	ED	TA
2k-Persona	0.08%	2.83	84.4%
5k-Persona	12.9%	1.48	93.4%
10k-Persona	49.6%	0.65	97.0%
20k-Persona	78.3%	0.27	98.8%
2k-Mental	0.03%	3.82	84.8%
5k-Mental	14.2%	1.79	92.9%
10k-Mental	64.0%	0.48	98.1%
20k-Mental	**84.1%**	**0.19**	**99.3%**
2k-Event	0.0%	4.90	81.8%
5k-Event	3.13%	2.64	90.2%
10k-Event	55.0%	0.61	97.7%
20k-Event	78.9%	0.25	99.1%
2k-All	0.0%	4.51	82.6%
5k-All	4.82%	2.31	91.1%
10k-All	35.1%	1.01	96.1%
20k-All	76.3%	0.29	98.9%

Table 4.3: Results from Persona, Mental-State, Event, and All-included datasets.

The results of performing experiments on the Atomic datasets quickly show that the Transformer model has a harder time translating these datasets than it had with DKET. Whereas it was almost perfectly translating DKET when it had trained on enough data, we only ever see results on Atomic that are as good as the RNN approach was capable of doing on DKET. This suggests that capturing variables and atoms in first-order logic might be harder for the model than the DL language ALCQ, making Atomic a much more challenging translation task. However, we observe in Table 4.3 that the model much more quickly finds the underlying structure of the desired logical formulas compared to ALCQ as the TA score is over 80% for all experiments even when using the smallest training datasets. The reason

for this is most likely due to the vocabulary of Atomic being roughly three times the size of the one used in DKET, which results in the model spending more time needed to create a semantic understanding of each word that appears in the dataset.

Example:
oReact if-then relation: PersonX kills PersonY's father → grief
correct formula:
A x y z ((person (x) & person (y) & kills (x,z,y) & father (z)) → grief (y))
predicted formula:
A x y z ((person (x) & person (y) & kills (x,z,y) & father (z)) → grief (y))

Here we have an example of a perfectly translated formula, where everything has been captured as expected. All the variables are quantified correctly, the atoms are correct and everything is wrapped in closed parentheses.

Example:
xAttr if-then relation: PersonX loves PersonX'd husband → enamored
correct formula:
A x z ((person (x) & loves (x,z) & husband (z)) → enamored (x))
predicted formula:
A x z ((person (x) & loves (x,z) & husband (z)) → affectionate (x))

In this example, on the other hand, we see that the inference word has been switched to a different one, while all the syntactic pieces of the formula are otherwise correct. This might be caused by the fact that "enamored" is quite an uncommon word, and often appear in the same contexts as "affectionate" which is observed ten times as much in the data. We even see it appear twice just for this specific event, which might have contributed to the incorrect choice of word.

Example:
xWant if-then relation:
PersonX is really sad → to speak with a friend or family member
correct formula:
A x z ((person (x) & is really (x,z) & sad (z)) → E a b (to speak with (x,a) & to speak with (x,b) & friend (a) & family member (b)))
predicted formula:
A x z ((person (x) & is really (x,z) & sad (z)) → E a b) to speak with (x,a) & to speak with (x,b) & friend (a) & family member (b)))

This final example demonstrates the last type of common issue, where a syntactic mistake has been made. In this case, a single parenthesis has been flipped, which in itself is the smallest type of mistake we can find in predictions, but is still considered incorrect like any other mistake. It might seem trivial compared to quantifying all the variables correctly and capturing all the atoms but it shows that sometimes keeping track of all the different parts of the formula patterns at the same time can be difficult.

Interestingly, the categories' results are very similar across the board, except that the mental state category has a noticeably better result than the other when trained on 20k examples. This was unexpected, as one would assume that the persona category would be the easiest to learn due to only consisting of one inference dimension that always centers around the subject of the event (PersonX), and has the shortest formula lengths on average, but this does not seem to be the case. In fact, persona achieves the same best accuracy at 78% as the event category, despite the event one having the largest set of examples to pick from and the most amount of relations. Thus it seems that the subtleties between the categories do not have a huge impact on the overall results, as all categories seem to improve similarly in all three evaluation metrics as the amount of training samples increases. When sampling from all categories, we also see the same results as doing them separately, showcasing that the model performs well across the board with an overall FA between 76% to 84% and a token accuracy at $\sim$ 99%, meaning that the incorrect translations are generally also really close to the correct formula.

This does lead to the overall interpretation of the results, that the biggest factor for performance is the amount of data that the model has to train on. We see that the model is not able to correctly translate at all when only having 2k samples to work with, despite

having a high TA, but it improves more and more as the training dataset size increases. From this, we assume that the Transformer's overall performance is hindered by the limitation of training data and that it will improve the more data it has available, an idea we will thus revisit later in the chapter.

4.4.3 Atomic without quantification

Training Dataset	FA	ED	TA
2k-Persona	0.06%	2.69	79.9%
5k-Persona	18.2%	1.32	90.1%
10k-Persona	46.1%	0.72	94.6%
20k-Persona	85.7%	0.16	98.8%
2k-Mental	0.17%	3.53	77.6%
5k-Mental	16.1%	1.67	89.2%
10k-Mental	61.3%	0.51	96.6%
20k-Mental	**86.7%**	**0.15**	**99.0%**
2k-Event	0.01%	4.32	74.4%
5k-Event	17.5%	1.59	90.4%
10k-Event	67.9%	0.41	97.5%
20k-Event	84.5%	0.19	98.9%
2k-All	0.003%	4.21	73.8%
5k-All	9.44%	1.97	87.9%
10k-All	68.6%	0.39	97.5%
20k-All	80.8%	0.23	98.6%

Table 4.4: Results from Persona, Mental-State, Event, and All-included datasets without quantification of variables.

Despite the Transformer model's ability to capture the atoms, and variables, and quantify them we do not however expect it to fully comprehend the complex meaning of it all. The formulas always follow the same pattern of universally quantifying any variable that is found in the body and potentially in the head, while any variable that may occur exclusively in the head is quantified existentially. This is a result of the algorithm created to make the formulas from the natural language, a deliberate design choice during our process. So, despite there being multiple equivalent ways of translating a sentence correctly, the model has only learned a specific pattern that applies to all examples and would therefore not have any reason to understand or produce alternate correct interpretations.

Thus, you could consider the quantification of the variables as only a matter of potential pattern recognition, the question then becomes how much this impacts the model's ability to translate and if we could achieve a better result if we omitted it from the formulas. The inclusion of them substantially increases the sequence lengths of the formulas and thus maybe we would see the model having an easier time correctly translating when the potential for mistakes is reduced as there are fewer total tokens to generate. If the results prove to increase dramatically, there might be a more viable solution to only train the model on formulas with the quantification omitted, and instead, run an algorithm on the results that finds all the variables and applies the same pattern as we used onto them instead as a postprocess.

In Table 4.4 we see the results of performing the same experiments on the Atomic dataset where the quantification has been removed from the logical formulas, and a first glance they look very similar. The model performs extremely poorly when trained using the smallest datasets. We can even observe that the overall TA score is worse on the 2k datasets compared to the experiments with the quantification included. This is likely due to the fact that there are fewer "pattern" tokens to translate, such as parentheses and $\forall$ that are always a part of the sequence's variable quantification. So, when a model correctly finds these structures and still predicts the formula wrong it can potentially achieve a higher TA than a model that lacks them. This is supported by the fact that the TA score is in many cases lower in these experiments compared to the equivalent experience with quantification, despite achieving a better score both in terms of FA and ED.

Interestingly, we notice that the models achieve an overall improvement of $\sim 5\%$ FA in all categories at their peak when trained on 20k training examples. The exception to this case is actually the mental state category, which was exceptionally good in the quantification experiment as well. Here we only saw a small improvement by a couple of percent in terms of FA, but this is now the new highest score across any experiment. Another interesting observation is that the event category has a much better result in the 5k training dataset in terms of FA, with an increased score of 14% despite having the exact same TA as in the equivalent experiment. In the 10k training dataset consisting of all types of relations we also see an FA increase of 33%, meaning that it can now capture 10k of the 30k validation examples that it was not able to when it had to quantify as well. This was also the worst-scoring experiment using 10k examples with quantification, suggesting that figuring out how to quantify with examples from the different categories was the biggest struggle for the model.

The biggest changes are noticeable in the 5k and 10k datasets, while the differences are much smaller in the 2k and 20k. This suggests that 2k is not enough data to capture the structure of the formulas at all, while at 20k examples the model has enough to memorize very well, regardless of quantification included or not. It matters more in the middle ground, where the model shows more prowess at capturing the atoms when they are its only focus, and it has a harder time when it needs to focus and quantify the variables at the same time. However, this does indeed support the idea that the difficulty of translation is to capture and split up the atoms correctly, as omitting the quantification did not suddenly cause the model to achieve near-perfect accuracy. Instead, we see an overall improvement, suggesting that shorter sequences and more ability to focus on capturing the atoms create better performance.

However, once again, we notice that the overall most important factor for the model's success lies in the amount of training data. All the experiments, except the variations mentioned, seem to follow the same pattern of improving roughly the same based on the amount of data it learns from in terms of both FA and ED. This, once again, reaffirms the idea that not hindering the amount of data the model has to train on will cause an even larger improvement in the model's performance.

4.4.4 Atomic with all the data for training

Training Dataset	FA	ED	TA
Persona	93.6%	0.07	99.69%
Mental	93.7%	0.07	99.73%
Event	93.1%	0.07	99.72%
All	**93.8%**	**0.06**	**99.74%**
Persona-No-Quantification	94.8%	0.05	99.59%
Mental-No-Quantification	94.7%	0.06	99.60%
Event-No-Quantification	93.3%	0.07	99.57%
All-No-Quantification	93.8%	0.06	99.58

Table 4.5: Results using all the data available, comparing with and without quantification of variables.

Since we observed that the models always performed the best when they trained on the largest dataset of 20k training examples, we decided to instead change up the training setup

to allow the model to learn as much as it wants from the data that we have available. The reasoning for using the datasets seen in the previous experiments was to compare them to the reference work using the RNN approach, where achieving good results on a few training samples was something they valued highly. However, due to our semantic approach and a larger vocabulary, allowing the model more data to see the words used in more contexts to create a better semantic representation for them could potentially lead to a better overall performance of the model. So, we decided to let the model use the entirety of the dataset and use an 85/15 split, where the model trains on 85% of the examples and is evaluated on 15% of them. We performed the experiments on all categories separately as well as the entirety of the dataset with quantification of the variables both included and omitted.

From Table 4.5 we can immediately observe that the model performs better across the board than any of the experiments on the smaller datasets did. The variation is also incredibly low, where we see a difference of less than 2% from best to worst in terms of FA. The ED is also ridiculously low, averaging less than half of the best score from the smaller dataset experiments, where the mental state on 20k training examples had the best of 0.15 while now all are at 0.07 and below. The TA score is now also always above 99.5% meaning that only 1 out of 200 tokens are incorrect in the translation, suggesting that even the incorrect formulas are extremely close to the correct.

Surprisingly, we see that now that the model has such a large amount of data to train on, it no longer matters if the quantification is omitted or not. The edge of the omitted results is insignificant as the overall result is extremely good for every experiment. It shows that with enough data the Transformer model is able to learn better how to both capture the atoms as well as to quantify the variables correctly. The overall results suggest that omitting the variables when you have a smaller amount of data available will lead to better performance, but when you have a lot there is no need to do so.

Chapter 5

Conclusion

In this chapter, we will summarize the work, what contributions we have brought and the results from our endeavors, as well as explore where the road goes next in terms of potential future work.

5.1 Summary

We have investigated the potential of using the Transformer architecture as the basis for performing neural machine translation of natural language into logical rules. When comparing the results of the experiments performed on the DKET datasets where all the words are known with the RNN approach, we can observe that the Transformer outperforms the RNN model when it has enough data to train on. This showcases that the Transformer architecture is promising, while the RNN model has still displayed abilities that we have not explored, in terms of translating definitions containing unknown words.

By designing an algorithm that transforms the Atomic if-then statements into logical rules, we were able to create a dataset that allowed us to observe how well the Transformer could learn the relationship between natural language statements and their corresponding rules. By using this algorithm were are able to create datasets of the categories independently, persona, mental state, and event, as well as for all combined. In addition, we created alternative datasets where the quantification of variables was omitted, resulting in us being

able to compare the translation proficiency of just learning atoms compared to the entire rule.

We achieved a high score across the board and witnessed that the main factor driving the results was the amount of data the model trained on. Omitting the quantification of the variables from the rules, which shortens the sequence lengths and simplifies the translation slightly, has a positive effect on the results when working on smaller datasets. The difference becomes minuscule when training on large amounts of data, as we observed a less than $\sim 2\%$ spread when using the entirety of the dataset. The experiments also showed that the differences between the categories did not cause huge variance in terms of performance, suggesting that the overall task of capturing atoms and quantifying variables are the main challenges of the task. The Persona category contains just a single inferential dimension and has the shortest inference length on average, but does not display the peak performance.

However, issues related to the RNN code being too old with outdated packages made it unfeasible for us to use their model to test Atomic for more comparisons. This would have given us further insight into how well the Transformer performs on the data relative to the RNN would, and how difficult the Atomic dataset is compared to DKET. The design of dataset was designed to allow for this, and we could observe if the syntactic approach would be capable of capturing atoms and quantifying variables without the use of semantic comprehension of the statements.

Overall, we see great potential for the Transformer to be used in this type of task. It displays strong capabilities of performing one-to-one translations of different subsets of natural language into different logical languages. We believe that this supports the possibility to use it for other translations, and the architecture can potentially allow for the generation of alternate, correct logical rules due to its semantic understanding.

5.2 Future work

Using a model that utilizes both syntactic and semantic reasoning to perform the task of neural machine translation from natural language to logical formula introduces a ton of new avenues to explore. In our work, the Transformer model learns to mimic the algorithm that created the dataset, essentially following the rules set up for atoms and variables decided by

it. However, this is only a single use case of the model, and the Transformer has also shown the ability to translate other subsets of natural language to other logical languages as shown in the DKET [20] experiments. So, the Transformer's shown ability to perform this task ends up opening the floodgates to explore more extensive and novel approaches to perform logical rule learning from natural language.

5.2.1 New version of Atomic

As mentioned when discussing the Atomic KB, there exists a newer and even more comprehensive version known as $Atomic^{20}_{20}$ [10]. Introducing two new categories describing inferential knowledge about common entities and objects, as well as common events that are related to one another, we see new dimensions of typed if-then relational knowledge. Expanding the work done to also cover these types of inferences could result in capturing an even greater proportion of natural language, and provide a wider coverage of if-then rules for use in language models. Due to the nature of these new dimensions being less similar than all three categories found in the Social-Interaction subset, which is equivalent to the original Atomic KB, we would need a more sophisticated way of creating logical formulas to learn from than the algorithm used in our work.

5.2.2 Pre-trained embeddings

During our experiments, we have used a model that starts its learning from scratch, i.e. has no previous knowledge and uses the Atomic dataset exclusively to learn from. This drastically limits the possibilities the model has to grasp the finer details of words as the number of contexts it observes them are limited in the if-then relations compared to large corpora of natural language.

There exist many pre-trained embeddings, vector representations of words and phrases, that are learned from large amounts of text data using unsupervised machine learning algorithms. Algorithms such as Word2Vec [13] and GloVe [14] have been immensely popular this last decade for injecting word representations into language models. Despite recently being considered outdated [27] due to the state-of-the-art transformers' excellent performance, they are still interesting to apply in our use case, where we do not train on large-scale corpora.

There is potential that the use of pre-trained embeddings will allow the model to achieve better semantic understanding between words and achieve the ability to translate into *better* logical formulas than what the input text provides. We might discover that the model makes "incorrect" predictions where instead of copying the word found in the original sentence it picks a synonymous word that is more applicable in the given context than a word chosen by the creator for the if-then relation. Experiments using different pre-trained embeddings could prove interesting insight into the potential of improving semantic understanding as a part of the translation process.

5.2.3 Testing for substitutivety

Relating strongly to the idea of introducing pre-trained embeddings, there is a potential to consider the possibility of testing for alternate correct logical formulas produced by the model. In our work and previous work [20], we strictly look for one-to-one translations where each generated formula is directly compared to the predetermined golden truth. In the RNN approach, this was the only logical approach as the model could not introduce words not found in the original sentence, but that is not the case for the Transformer model which also introduces semantic understanding. There is potential for our model to be "incorrect", but only due to using a synonymous word that means the same thing in the context and would be considered a completely fine formula.

There are many ways of expressing an idea in natural language, and the way we choose to do so is heavily reliant on our vocabulary. There are a lot of words in English that are extremely general that can be applied to a multitude of different contexts, such as "work" and "make". There are similar words to these types of words that express the same sentiment but are more specific. If we had an event such as "six months of hard **work**", then we would consider the statement of "six months of hard **labor**" as an adequate alternative and potentially even preferable due to the less general term of labor being used. If the Transformer decided to translate one of these sentences into a rule that ended up swapping these two terms, we would not consider this a failed translation, but rather an alternative one.

We do not test for this in our experiments, but it could prove significant if sometimes the model is correct in a different way than expected. Therefore, creating an algorithm

that compares the predicted formula to the expected result to find out if they contain the same meaning would be a step forward in terms of evaluating this type of task. It would be extremely important with the potential of introduction of pre-trained embeddings, and a heavier focus on utilizing the semantic understanding to create even better translations.

5.2.4 Testing for unknown words

One of the better features, and most important of the syntactic approach using a syntactic RNN model to perform translations is its ability to deal with unknown words. Since it always treats any word in a formula as an index in its input, it quickly achieves a powerful ability to capture the underlying syntactic structure that shapes definitory sentences. This means that when some of the content words of the sentence are replaced with "unknown" tokens it performs equally well.

We did not pursue this avenue in our experiments due to the limited time, but it is still something that has to be explored in the future. It is possible that the semantic prowesses of the Transformer can be a hindrance when dealing with incomplete sentences that uses the same token in way too many different contexts, or it might help it focus more on the syntactical strategy of copying the words seen in the sentence instead of looking for the underlying meaning. No matter what, it can help determine if the Transformer can be the ultimate all-purpose architecture for this task or if a syntactic RNN approach or another type of architecture may reign supreme in some use cases.

5.2.5 Out-of-distribution if-then statements

Every single relation from the Atomic dataset starts with the word "PersonX", which potentially affects the model's ability to learn to translate Atomic much faster than it should. The pattern of always including $\forall x$ and $Person(x)$ in every rule makes it easier to discover the general structure of how the rules are structured, therefore making it uncertain how well the model performs outside of its domain of use.

Since the model is trained on a dataset created from an algorithm that is specifically designed to create rules from the Atomic knowledge bank, it is fair to state that this is

a very specific type of if-then statement being represented and that there is no guarantee that datasets consisting of other types of syntactically structured statements would fare as well. How much the regularity of certain patterns affects the overall performance is therefore unclear, and would need to be investigated further. On the other hand, this is part of the larger issue in exploring the limits of what subsets of natural language the Transformer can learn to translate effectively into logical rules.

5.2.6 Other expressions of natural language to logic

Naturally, the holy grail of this neural translation task would be to be able to translate any natural language text into logical formulas defined in any logical language desired. So far we, have seen both translations of definitory sentences as well as if-then relations, but this is only a tiny subset of the endless possibilities to express knowledge. Performing more work involving: more specified semantic language models, and text consisting of compound and complex sentences, that result in rules that include words not found in the input sentences would all be huge leaps forward for this type of task. Previous results have shown that syntax is important, but using the Transformer we can see quality results being achieved using an approach that does not omit semantics, which means that we have more tools to achieve such advances.

5.2.7 Injecting rules into LMs

A future prospect could be to combine the best of two worlds, the LMs from the machine learning community and the logic-based reasoning systems from the automated reasoning community. The LMs could create rules from a natural language using neural machine translation, and the rules could be evaluated by the reasoners to generalize knowledge, which could be injected back into other LMs to improve their generalized training. Thus, the reasoning could be used to help mitigate the issues with logic-based reasoning in LMs.

The potential gain would be that when a LM is trained on a large-scale corpora they could also be fed logical rules that relate to the words found in the context, which would help to generalize the factual relations and contradicts the learning of biases and illogical statements. Thus, both parties could perform what they are best at, LMs capturing patterns

and regularities from the data and the reasoners performing logical evaluations of rules, where the latter becomes a course-corrected for the former. How feasible or effective this could be is uncertain, but the idea in and of itself spurs the effort to explore methods of creating, learning, and extracting logical rules to be used for such reasoning in an as accurate and effective way as possible to be able to combine the two communities strengths.

Bibliography

[1] *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2 edition, 2007. doi: 10.1017/CBO9780511711787.

[2] Smith R. Aristotle. *Prior Analytics*. HPC Classics Series. Hackett, 1989. ISBN 9780872200647.
URL: https://books.google.no/books?id=A751t2hTxn8C.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[4] Edward Loper Bird, Steven and Ewan Klein. Natural language processing with python, 2009.

[5] Myles. Burnyeat, M. J. Levett, and Plato. *The Theaetetus of Plato / Myles Burnyeat; with a translation of Plato's Theaetetus by M.J. Levett, revised by Myles Burnyeat*. Hackett Indianapolis, 1990. ISBN 0915144816 0915144824.

[6] Zhifang Fan, Zhen Wu, Xin-Yu Dai, Shujian Huang, and Jiajun Chen. Target-oriented opinion words extraction with target-fused neural sequence labeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2509–2518, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1259.
URL: https://aclanthology.org/N19-1259.

[7] The Linux Foundation. Pytorch library transformer class. https://pytorch.org/docs/stable/generated/torch.nn.Transformer.html, 2017. [Online; accessed 29-March-2023].

[8] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, Nov 2020. ISSN 2522-5839. doi: 10.1038/s42256-020-00257-z.
URL: https://doi.org/10.1038/s42256-020-00257-z.

[9] Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? (extended abstract). In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 5065–5069. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/708.
URL: https://doi.org/10.24963/ijcai.2020/708. Journal track.

[10] Jena Hwang, Chandra Bhagavatula, Ronan Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Choi Yejin. (comet-) atomic 2020: On symbolic and neural commonsense knowledge graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35: 6384–6392, 05 2021. doi: 10.1609/aaai.v35i7.16792.

[11] C. G. Jung. *Two Essays On Analytical Psychology*, volume 7 of *Collecteed Works of C.G. Jung*. Princeton University Press, 1967.

[12] Dan Jurafsky and James H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 3rd ed. draft*. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2009.

[13] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.
URL: http://arxiv.org/abs/1301.3781.

[14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.
URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[15] J.R. Norris. *Markov Chains*. Cambridge University Press, 1997.

[16] Sharon O'Brien. Controlling controlled english - an analysis of several controlled language rule sets. *Joint Conference Combining the 8th International Workshop for the European Association for Machine Translation and the 4th Controlled Language Applications Workshop*, 2003.
URL: https://aclanthology.org/2003.eamt-1.12.pdf.

[17] OpenAI. Chatgpt: Optimizing language models for dialogue. https://openai.com/blog/chatgpt/, 2022. [Online; accessed 3-February-2023].

[18] Kishore Papineni, Salim Roukos, Todd Ward, and Wei Jing Zhu. Bleu: a method for automatic evaluation of machine translation. 10 2002. doi: 10.3115/1073083.1073135.

[19] Giulio Petrucci, Chiara Ghidini, and Marco Rospocher. Ontology learning in the deep. In *International Conference Knowledge Engineering and Knowledge Management*, 2016.

[20] Giulio Petrucci, Marco Rospocher, and Chiara Ghidini. Expressive ontology learning as neural machine translation. *Journal of Web Semantics*, 52-53:66–82, 2018. ISSN 1570-8268. doi: https://doi.org/10.1016/j.websem.2018.10.002.
URL: https://www.sciencedirect.com/science/article/pii/S1570826818300507.

[21] Jorge Pérez, Javier Marinković, and Pablo Barceló. On the turing completeness of modern neural network architectures. In *International Conference on Learning Representations*, 2019.
URL: https://openreview.net/forum?id=HyGBdo0qFm.

[22] Bertrand Russell. *The Problems of Philosophy*. Barnes & Noble, 1912.

[23] Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. Atomic: An atlas of machine commonsense for if-then reasoning. AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. ISBN 978-1-57735-809-1. doi: 10.1609/aaai.v33i01.33013027.
URL: https://doi.org/10.1609/aaai.v33i01.33013027.

[24] Robyn Speer and Joanna Lowry-Duda. ConceptNet at SemEval-2017 task 2: Extending word embeddings with multilingual relational knowledge. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 85–89, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi:

10.18653/v1/S17-2008.
URL: https://aclanthology.org/S17-2008.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[26] Johanna Völker, Pascal Hitzler, and Philipp Cimiano. Acquisition of owl dl axioms from lexical resources. In Enrico Franconi, Michael Kifer, and Wolfgang May, editors, *The Semantic Web: Research and Applications*, pages 670–685, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[27] J. von der Mosel, A. Trautsch, and S. Herbold. On the validity of pre-trained transformers for natural language processing in the software engineering domain. *IEEE Transactions on Software Engineering*, 49(04):1487–1507, apr 2023. ISSN 1939-3520. doi: 10.1109/TSE.2022.3178469.

[28] Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang, and Guy Van den Broeck. On the Paradox of Learning to Reason from Data, May 2022.
URL: http://arxiv.org/abs/2205.11502. arXiv:2205.11502 [cs].