Implementing Natural Language Into Knowledge Base Generation

Jethro Dunn
dnnjet002@myuct.ac.za
University of Cape Town
Cape Town, South Africa

ABSTRACT

Knowledge Representation and Reasoning provides a framework for representing information in a form that artificial intelligence and computers can reason with. Using propositional logic, classical reasoning treats statements as strictly true or false, but real world concepts rarely are true under all circumstances. Defeasible reasoning, formalised by the KLM framework, can be used to allow statements to be true in certain contexts. Using this framework, statements can be put into a hierarchical ranked structure to form ranked knowledge bases. The knowledge base generators that create these tables use symbolic representation to create synthetic propositional statements that are ranked. Currently these knowledge base generators are mainly used to test defeasible entailment algorithms such as rational closure or lexicographic closure. These algorithms are used to query the knowledge base to check whether a query is true according to the information in the knowledge base. Since the knowledge is all symbolic, and synthetically created, it is impossible for a person to directly interpret them. This project addresses this gap by using a large language model to convert the symbolic propositions into coherent English sentences. The sentences created preserve their logical structure while making the underlying reasoning more accessible to human readers. This project contributes to bridging the gap between symbolic reasoning and human interpretability in explainable artificial intelligence.

KEYWORDS

Artificial Intelligence, Knowledge Base Generator, Knowledge Base, KLM Framework, Defeasible Reasoning, Rational Closure, Explainable Artificial Intelligence, Natural Language Processing

1 INTRODUCTION

Propositional logic is a foundational form of knowledge representation and reasoning (KRR), where information is expressed using structured logic to enable automation and decision making by algorithms [27]. In this framework, statements, called atoms, are assigned truth values and logical connectives are used to combine these into formulas [2, 9]. Collections of formulas are set as the rules for a domain of information, called knowledge bases (KBs). Formulas are placed into hierarchical tables which provide a structured representation that can be queried and reasoned with.

Classical reasoning is the primary form of logic which applies strict inference, where the consequence is always entailed by the knowledge base [19]. This does not allow for any exceptionality, making it unsuitable for real world reasoning. For instance, the following rule "birds can fly" or $b \rightarrow f$ would imply that either penguins are not birds or that they cannot exist. To handle situations like this, defeasible reasoning was formed and the KLM framework was created as a formal framework for it [23]. This framework allowed for exceptionality, using the symbol $\mid\sim$ to represent "typically implies". These formulas can be ranked according to their level of exceptionality by using various forms of inference such as rational closure[26].

The current implementation of a knowledge base generator (KBG) can produce tens of thousands of defeasible implications per second [32]. These KBGs are highly effective for testing inference algorithms such as rational closure and lexicographic closure, but the KBs generated are synthetically generated and unintelligible for humans. They lack interpretability and cannot be understood or validated easily even by experts.

This project aims to explore how large language models (LLMs) can be used to make these synthetically generated knowledge bases more interpretable. Gemini Flash 2.5 is used to allocate natural language phrases to atoms, producing English sentences that still preserve the logical structure of the knowledge base statements. These "generated sentences" will be compared to a baseline where random, but thematically consistent, phrases are allocated to atoms, referred to as "manual sentences". The three different types of models used to attempt to determine the coherence and understandability of these sentences were: SBERT, KenLM, and Gemini.

Improving the interpretability of KBs contributes to the wider field of explainable artificial intelligence (XAI). By making symbolic reasoning outputs more understandable, it can help support greater transparency and trust in AI systems. This is relevant not only for academics in reasoning, but in any field where decision making could be derived from knowledge bases.

2 BACKGROUND

This section explains how defeasible reasoning fits into the broader reasoning field. It explains the general syntax and rules used to apply defeasible reasoning and structure it so that it can be used in KRR. Rational and lexicographic closure are also presented as valuable forms of defeasible entailment that can use the results of this project.

2.1 Propositional Logic

Propositional logic is a formal system of reasoning with statements that are assigned truth values [2]. Propositions are created from atoms, which are pieces of information, that are combined with boolean and unary connectives [20]. By doing this, natural language statements can be expressed as logical formulas, which can be evaluated. Propositional logic is the foundation for the two forms of reasoning used in this project, classical and defeasible reasoning.

Propositional logic is a formalised language meaning that it uses specific notation. The following subsections describe the three main components used to represent propositional logic.

2.1.1 Atoms. Atoms are basic statements which can be assigned true or false values. They are units of information that can be combined into formulas using operators.

For example w which represents the statement "wings" and b which might represent "birds". More complex statements can also be encapsulated by an atom, such as "the tide is strong tonight" can be represented by t.

When considering a knowledge domain, the set of all atoms is finite and are referred to as P [20].

2.1.2 Operators. Boolean operators are used to join statements together. With the exception of negation ($\neg$), which is unary, they are binary operators [2]. These are used to form truth tables which result in a true or false outcome. The standard connectives are:

$\neg$ negation
$\wedge$ and
$\vee$ or (inclusive)
$\rightarrow$ if then
$\leftrightarrow$ if and only if

2.1.3 Formulas. The combination of atoms and operators creates formulas, which provide more complex combinations of statements. Lower case Greek symbols (e.g. α, β) are typically used to represent formulas that can be derived from a knowledge base. The symbol, $\mathcal{L}$, is used to denote the set of all these formulas [2]. Formally, the set of formulas in $\mathcal{L}$ is:

$$
\begin{aligned}
fml &::= p && \text{for some } p \in \mathcal{P} \\
fml &::= \neg fml \\
fml &::= fml\ op\ fml \\
op &::= \vee \mid \wedge \mid \rightarrow \mid \leftrightarrow
\end{aligned}
$$

For example, $\alpha \rightarrow \sigma$ is a formula where $\alpha, \sigma \in \mathcal{L}$.

2.2 Semantics

Semantics assign meaning to symbols of propositional logic. Object-level semantics describes truth within specific worlds. Meta-level is used to evaluate knowledge about a knowledge base [20].

2.2.1 Object-Level Semantics. Within propositional logic, the object-level refers to all parts of a language used to model knowledge, including formulas, operators and atoms [20].

Valuations, also called *interpretations* or *worlds*, determine how truth values are assigned to atoms. This is important, as it then allows for satisfaction and entailment [5]. Formally:

- Given set $\mathcal{P}$= p,q,r, ...
- Valuation u maps each atom p $\in \mathcal{P} \in$ T,F

Where $u(p) = T$ means p is true in u, while $u(q) = F$ means q is false. This can be written as $u = p\overline{q}$.

The truth of formulas are determined by standard truth tables:

α	β	$\alpha \vee \beta$	$\alpha \wedge \beta$	$\alpha \rightarrow \beta$	$\alpha \leftrightarrow \beta$
T	T	T	T	T	T
T	F	T	F	F	F
F	T	T	F	T	F
F	F	F	F	T	T

Figure 1: Truth Table for Boolean Operators

2.2.2 Meta-Level Semantics. Meta-level semantics are used to evaluate knowledge about some knowledge [20]. Entialment and satisfiability are core concepts of meta-level semantics.

Classical entailment is defined as knowledge base $\mathcal{K}$ entails an α if every model of $\mathcal{K}$ is a model of α [2, 20]. This can be shown as $\mathcal{K} \models b \rightarrow f$, which reads as: in $\mathcal{K}$ it follows that if it's a bird then it can fly.

A statement or formula can be said to be satisfiable if and only if there exists some valuation u which is true [2]. If there is no situation where a formula is true, then it is said to be unsatisfied. For example:

- satisfied: $u \Vdash q$ where $q = True$
- unsatisfied: $u \nVdash q$ where $q = False$

2.3 Defeasible Reasoning

Classical reasoning is monotonic, meaning that once a conclusion is drawn, new information cannot invalidate it [19, 34]. This is limiting in the real world, where there are often exceptions to typical behaviour and new information can invalidate conclusions. For example, the rule "birds fly" has been shown to be incorrect, since penguins exist. To accommodate these exceptions, the statement needs to change to be "birds **typically** fly" [5].

The Kraus, Lehmann and Magidor (KLM) framework is used to determine entailment in a defeasible context, using the concept of typicality [23]. Formally, typicality can be denoted by $\mid\sim$ and for $\alpha, \beta \in \mathcal{L}$, the defeasible implication $\alpha \mid\sim \beta$ is read as "typically, if α then β" [5]. This framework proposed six relations for preferential consequence relations (a formal framework for non-monotonic reasoning) that a nonmonotonic consequence relation should satisfy [20].

The KLM framework sets out structures to create ranked interpretations [26]. A ranked interpretation is a function $\mathcal{R}$ that maps statements to a rank from 0 to ∞ (infinity) [5]. Given some knowledge base $\mathcal{K}$, where P = b, f, p representing birds, fly and penguin, $\mathcal{R}$ can be created. To set up the ranking, 0 represents the most typical valuations, higher levels ($\mathbb{N}$) are less typical and rank ∞ contains impossible valuations.

The following ranked interpretation is one possible ranking that could be constructed:

∞	$a\overline{b}c\ a\overline{b}\overline{c}$
1	$\overline{a}bc\ ab\overline{c}\ abc$
0	$ab\overline{c}\ \overline{a}\overline{b}c\ \overline{a}bc$

Figure 2: Ranked Interpretation of $\mathcal{P}$

This is an arbitrary ranking, with no formulas dictating typicality. An example of what can be deduced from this ranking is that $a\bar{b}c$ is impossible to occur and $ab\bar{c}$ is one of the most typical cases.

2.3.1 Entailment. In this framework, entailment represents defeasible inference, where the notation $\approx$ is used to denote typicality. Rational closure and lexicographic closure are two examples of entailment under the KLM framework [25, 26].

The core of the KLM framework for non-monotonic reasoning shows how defeasible entailment and ranked interpretations are formed. Ranked entailment and minimal ranked entailment extend this to formalise how conclusions are drawn from defeasible knowledge bases.

2.3.2 Ranked Entailment. A knowledge base $\mathcal{K}$ rank entails a defeasible implication $\alpha \mid\sim \beta$ when, in all ranked interpretations that satisfy $\mathcal{K}$, typically if α is true, then β is true [20, 26]. Symbolically ranked entailment can be shown as $\alpha \mid\sim \beta, \mathcal{K} \models_R \alpha \mid\sim \beta$ [26]. This is still a monotonic relation, to make it non-monotonic the concept of minimal ranked entailment will be used [20].

2.3.3 Minimal Ranked Entailment. A partial ordering of all ranked models of $\mathcal{K}$ can be made where ranked interpretations are moved down the ranks by their valuations, for example $Cn_1(\mathcal{K}) \preceq CN_2(\mathcal{K})$ [26]. Using this, minimal ranked entailment can be defined where for every defeasible implication $\alpha \mid\sim \beta, \mathcal{K} \approx \alpha \mid\sim \beta$ if and only if $\alpha \mid\sim \beta$ satisfies the ranked interpretation [10].

2.4 Rational Closure

Rational closure is seen as the most conservative construction of non-monotonic closure [5]. It attempts to make as few assumptions as possible about atypical implications. Rational closure uses ranking with two formal terms; *between* ranked models is where lower ranked models are more typical and, conversely, higher ranked models are less typical [10]. Using the same intuition, the term *within* ranked models of $\mathcal{K}$, are where valuations further up are less typical and ones lower down are more typical [5].

The following notation is used for rational closure:

$\mathcal{R}$	ranked interpretation
$\mathcal{R}^{RC}_{\mathcal{K}}$	rational closure over a knowledge base
$\preceq_{\mathcal{K}}$	partial order

2.4.1 Base Ranks. The base rank of a propositional statement α, for some $\mathcal{K}$, is a measure of how exceptional, or atypical, α is in $\mathcal{K}$ [5, 10]. Base ranks are used to determine the level where α becomes consistent with the typicality of $\mathcal{K}$. It quantifies the minimum level of typicality at which α becomes consistent with the defeasible rules in $\mathcal{K}$. That is, the $br_{\mathcal{K}}(\alpha)$ is defined as the smallest r where α is not exceptional with respect to $\mathcal{E}^{\mathcal{K}}_r$[5]. The knowledge base generator used in this project uses the base rank approach to place implications into ranks. This can then be used by rational and lexicographic closure for querying implications.

2.4.2 Minimal Ranked Model. The minimal ranked model is a ranking of a knowledge base using minimal ranked entailment. Unlike ranked entailment, minimal ranked entailment is non-monotonic, so it follows that the minimal ranked model is also non-monotonic [20]. The following is an example of a minimal ranked model [5]:

Given $\mathcal{P} = b, f, p$, with statements:

(1) p $\rightarrow$ b
(2) b $\mid\sim$ f
(3) p $\mid\sim \neg$f

The minimal ranked model can be construed as follows:

∞	$\text{p}\bar{\text{b}}\text{f}$ $\text{p}\bar{\text{b}}\bar{\text{f}}$
2	pbf
1	$\bar{p}b\bar{f}$ $pb\bar{f}$
0	$\bar{p}\bar{b}\bar{f}$ $\bar{p}\bar{b}f$ $\bar{p}bf$

Figure 3: Ranked Interpretation of $\mathcal{P}$

It is assumed in the minimal ranked model that statements are typical until proven otherwise, so all propositions begin on level 0. The following describes how propositions are moved to create the ranked model in figure 4:

(1) $p \rightarrow b$: $\text{p}\bar{\text{b}}\text{f}$, $\text{p}\bar{\text{b}}\bar{\text{f}}$ are impossible as penguins have to be birds, so they are place in the ∞ level.
(2) $b \mid\sim f$: $\bar{p}b\bar{f}$, $pb\bar{f}$, pbf are push up to level 1 since birds typically do fly.
(3) $p \mid\sim \neg f$: pbf is pushed up another level, since $\bar{p}b\bar{f}$, $pb\bar{f}$, pbf are all on the same level, but p $\mid\sim \neg$f says that in the best bird worlds, penguins typically do not fly.

2.4.3 Rational Closure Algorithm. Rational closure uses the partitions made by base ranks for queries, checking if implications are true in a knowledge base. Given $\alpha \mid\sim \beta$ as an implication, first α is checked if it is exceptional in the lowest rank [5]. If α is exceptional, then the rank above is checked, repeating until α is not exceptional. When α is no longer exceptional, $\alpha \mid\sim \beta$ is checked if it holds at that rank. If it does hold then the implication is accepted by rational closure, otherwise if it does not hold or α reaches level ∞ then it is not accepted.

2.5 Lexicographic closure

Lexicographic closure is another form of non-monotonic defeasible entailment which uses a ranked model to represent a given knowledge base [4]. Lexicographic closure is a bolder form of closure, where statements inherit more from general statements than in rational closure.

Lexicographic closure was proposed by Lehmann and in defining it he provided four informal properties to guide this form of entailment: presumption of typicality, presumption of independence, priority of typicality, and respect for specificity [20, 25].

The presumption of typicality means that given two statements the one with stronger justification is preferred. For example, given $b \mid\sim f$ the rational entailment could be $\mathcal{K} \approx b \vee p \mid\sim f$, or it could be $\mathcal{K} \approx$ b $\mid\sim \neg p$. Both of these are feasible, but according to the first condition, the former is preferred [20]. The presumption of independence means that you assume typicality for a statement, unless there is a conflict that suggests it is not typical. The priority of typicality is used where there is a conflict between the first two guidelines. It resolves this conflict by choosing the presumption

of typicality as being more significant. The fourth guideline is used when any 2 inferences clash, here the implication with the most specific antecedent is preferred. This does not have a perfect formalisation and is specific to the knowledge base. An example of this is that, given two antecedents, *penguin* and $AfricanPenguin$, the $AfricanPenguin$ is a more specific antecedent than penguin. From this, the inference with $AfricanPenguin$ as the antecedent is the most preferred.

2.5.1 Ranked Lexicographic Closure. As with rational closure, interpretations must be ranked (R_{LC}) [20]. To rank the implications a level of "seriousness" is assigned to each implication and preference will be given to those that contradict less serious implications. To understand "seriousness", Lehmann defined two properties: set size and specificity of elements [25]. Set size is where violating a smaller set is less serious than a larger one. Specificity of elements is where less specific elements cause a lower level violation. When constructing the ranked lexicographic closure, the second rule (specificity) is applied first and then any further conflicts are settled using the first rule [20].

2.5.2 Lexicographic Closure Algorithm. Lexicographic closure extends rational closure by splitting ($\mathcal{R}_{\mathcal{K}}^{RC}$) into sub-levels, which still maintain the initial ordering of interpretations [5]. Within each level, implications are ordered by their seriousness. When a conflict arises, lexicographic closure prefers conclusions that violate the less serious implications, while respecting $\mathcal{R}_{\mathcal{K}}^{RC}$.

This project does not integrate with a lexicographic algorithm, but it would be useful since it reflects a less cautious form of reasoning. It complements rational closure by offering alternative strategies to phrase allocation when generating defeasible implications.

3 RELATED WORK

This project extends previous research done on knowledge base generation. Aiden Bailey created the initial implementation of a knowledge base generator with parameters for the: number of defeasible implications, number of ranks, distribution used, whether only defeasible statements should be used and if conservative generation should be used [1]. Subsequent extensions to the project improved the knowledge base generators efficiency through multithreading and space-time techniques [24, 32]. The parameters used to create the knowledge base were also expanded, including options for transitivity and consequent reuse, extending the available distributions and being able to select the number of antecedents and consequence per implication.

3.0.1 Natural Language Processing. It is difficult to determine the understandability and coherence of sentences according to people for multiple reasons, such as differences in individuals knowledge and level of literacy. Research comparing the ability of different models to predict human relative sentence probability judgements on controversial sentence pairs, found that GPT-2 and RoSBERTa were the best at making these judgements [11]. They did conclude that although GPT-2 performed the best, but it still made decisions inconsistent with human judgement.

3.0.2 Rational Closure and User Interface. The knowledge base generator fits into a broader project which uses the knowledge bases created. The rational closure algorithm, which has been optimised to process knowledge bases can take in the knowledge bases created by this project and can process if a query about the knowledge base is true or false [16]. The user interface is used to show the knowledge base and result of a query and explains how the query reached its conclusion [7]. This was partially integrated so that the phrases correctly replaced atoms, however the knowledge base generator and user interface are autonomously connected and do not share the full sentences generated by Gemini.

4 METHODS AND MATERIALS

4.1 Hypothesis and Research Objectives

4.1.1 Hypothesis. The hypothesis for this project is that by using Gemini Flash 2.5, symbolic implications can be turned into English sentences which are more understandable than if random phrases are used to form sentences.

4.1.2 Aim. The aim of this project is to improve the understandability of knowledge bases created by an existing knowledge base generator. The generated sentences should preserve their logical correctness while also being more understandable and accessible to users.

4.1.3 Research Questions. This report addresses the following research questions:

(1) How can sentences from a knowledge base be generated using natural language phrases so that the sentences are understandable and logically correct?
(2) Are implication sentences generated by an LLM more understandable than random phrases allocated to implications?
(3) Can sentence understandability be effectively measured using a word embedding approach (SBERT), an N-gram model (KenLM), and an LLM approach (Gemini)?

4.2 Hardware and Software Required

4.2.1 Hardware. The experiment was conducted on a 2020 MacBook Air with an Apple M1 processor and 8GB of RAM.

4.2.2 Software and Libraries. Several key pieces of software were used to create the models used in this project. Two varieties of SBERT were used by this project, multi-qa-mpnet-base-dot-v1 and all-MiniLM-L6-v2 [17, 18]. ConceptNet was used in preliminary testing. It is a graph based natural language processing tool for common knowledge, but it was not used in the final experiment [33]. The KenLM model was used, which is a form of optimised N-gram model [21]. The final major software used was the Gemini API for Flash 2.5 [8].

4.2.3 Limitations. The project faced several main limitations. The first was that the available hardware limited the ability to train more sophisticated models. Both the ConceptNet and N-Gram models used would have benefited from improved hardware that facilitated more complex actions. The second limiting factor was that the highest freely available model of Gemini was Flash 2.5. This is not the latest LLM from Gemini for general reasoning and was not able to produce sentences as well as the newest models. Using this model

in the free tier also limited the ability to run more extensive tests since there are daily constraints on tokens and queries. The last major constraint was that limited time and resources prevented this project from conducting human trials to rate sentence understandability. This would have provided a "gold standard" to measure and compare sentence understandability to.

5 IMPLEMENTATION

To accomplish the goal of creating a more understandable and usable knowledge base, an LLM was used to attach phrases to atoms.

5.1 LLM implementation

When a knowledge base is generated multiple parameters are used to create the desired knowledge base. Originally, Latin and Greek symbols were used to represent atoms so that if Latin is selected it would use $a, b, c...$ and if Greek was selected it would use $\alpha, \beta, \gamma...$. This project introduced a new option, *phrases*. When this is used, the knowledge base is generated as it would have been before, always using the lower Latin alphabet. This ensures that no logical information is lost or corrupted. Once the knowledge base is correctly generated, each individual rank is separately sent to the GeminiAPI class to be processed by the LLM.

After setting up the connection with Gemini, the rank is processed [13]. This process checks each implication with its atoms to see if those atoms have previously been mapped to a phrase. If an atom has a phrase allocated to it, that phrase replaces the atom and the new list of statements is returned. Similarly, a list of all the atoms that have no mapping are also returned.

For example, this transforms a rank where some atoms have already been allocated from:

$$a \mid\sim b, c \mid\sim d$$
$$to:$$
$$a \mid\sim \text{birds},\ \text{cars} \mid\sim d$$

This information is crucial for prompting the LLM. By providing the implications with phrases already filled in and a list of unmapped atoms, it helps reduce hallucinations.

5.1.1 Prompt. The prompt provided to Gemini was created by following Google's guidance on prompt strategies [15]. Initially the prompt provides context and outlines the task which needs to be completed (forming English sentences given implications). The prompt is structured using headings to clearly separate topics. The symbols are explained to give the model context and ensure that it does not confuse them with other meanings, such as by confusing the defeasible and classical implication symbols. Ranks are explained since lower ranks should reflect more certain events and higher ranks reflect lower certainty events. This explanation is key since as a result of ranking statements, higher ranks are less certain; so this must be reflected in the sentences produced. General rules for how atoms should be mapped were also provided and a constrain of phrase length, from one to five words long, was stipulated.

In the initial implementation of this prompt, operators were always replaced with the same word or set of words. For example, $\rightarrow$ was always replaced with "implies". This was done to try stay inline with how statements and formulas are read logically. This also meant that Gemini's output could be parsed to extract the sentences and atoms. This provided a higher level of control over the outputs and allowed for less opportunities for the LLM to hallucinate. However, as a goal of the project was to create sentences that are both logical and coherent, this strict replacement was removed. Instead, the prompt provided a list of example words and phrases for the model to use to replace these operators. The model was not constricted to these, but they were provided to help ensure that it did not misunderstand and assign incorrect terms to those symbols.

Due to the randomness of sentences and the inability to parse out symbols, an output format was provided so that the sentences and the atoms with their mapped phrases could be provided by the LLM.

The prompt used a few shot approach, which is where examples of how actions should be completed were provided. This is another method to reduce hallucinations as a zero shot approach, where no examples are used, leaves actions up to the LLMs interpretation. The examples provided followed a positive pattern approach, encouraging correct responses rather than showing incorrect ones.

5.2 Manual Sentence Generation

To evaluate how much using an LLM to turn implications into sentences improves understandability, manual sentence generation was also required. A list of random phrases, mostly between three and six words in length was created [35]. These phrases were all within the same general topic of conversational phrases. The implications generated by the KBG were parsed, replacing atoms with one of these phrases. Symbols were also parsed and replaced with their associated words, such as $\rightarrow$ always being replaced with "implies". Atoms are stored with their associated phrase so that repeated atoms use the same phrase.

6 VERIFYING SENTENCE CONCEPTS

To test if replacing atoms with phrases using an LLM were more understandable compared to allocation of random phrases, tests needed to be done to calculate the coherence and understandability of sentences. Measuring this is a difficult task with many possible ways to test it [28]. Since there are multiple ways to test the understandability of sentences, the following models were applied.

6.1 SBERT

Sentence Bidirectional Encoder Representations from Transformers (SBERT) is a variant of BERT which is designed for sentence level embeddings, rather than word level embeddings [30]. This can be used to calculate similarity scores between phrases. It calculates a vector representation across a sentence and uses a cosine function to calculate similarity.

6.1.1 All-mini-6L-V2. The first approach taken at using SBERT was with the all-miniLM-L6-V2 via Hugging Face which is a fine tuned variant of Reimers and Gurevych's Sentence-BERT [17, 31]. This model is designed to run on sentences and short paragraphs where it maps text to a 384 dimensional vector space. The largest

contribution to its training was comments from Reddit between 2015 and 2018.

6.1.2 multi-qa-mpnet-base-dot-v1. The second model used, and the one which was used in the experimental testing was the multi-qa-mpnet-base-dot-v1 [18]. This model is used for sentences and short paragraphs and maps to a 768 dimensional vector space. It was trained on 216 million question-answer pairs. This training tended to capture the multi-part nature of the sentences being produced by Gemini.

6.2 Universal Sentence Encoder

The Universal Sentence Encoder from Google is used to get sentence level embeddings which can then be used to get sentence level meaning [6, 12]. The model used was version 2 of the universal sentence encoder model published by Google [14].

Since USE and SBERT both use sentence level encoding, preliminary testing was done to select one of them to conduct the full tests on. From this testing, SBERT was selected since it was able to capture a higher level of semantic similarity in the sentences used.

6.3 ConceptNet

ConceptNet is a knowledge graph built to link related words and phrases together [33]. It uses labels and weights to connect these words and phrases. As the graph provided words and phrases that related to one another, an attempt was made to use the weights and labels to provide a "relatedness" score. Two different varieties were used to test this, a two hop and a three hop approach, where the number of hops indicates the number of outward edges that would be searched. While the three hop approach was much more successful in finding related words and phrases, it took several minutes for a single short sentence to run. Since three hop took too long, it meant that most testing was done using the two hop method.

The first approach taken was to identify the number of related words and phrases between the different parts of sentences. This yielded poor results as two hops tended to only yield one or two relationships even in longer sentences. The second approach attempted to weight the relationship scores using both the edge weighting and the relationship label. Different multipliers were attached to the different relationship types to encourage certain relationships. By using these as multipliers, it aimed to improve scores which used relevant relations. As these scores had to be manually assigned, sentences scores were sporadic and persisting low numbers of related words hindered performance.

Due to this sporadic scoring and manual assignments of weightings, this algorithm was not used in the final testing.

6.4 KenLM

An N-gram model is one which uses the N-1 words to predict the probability of the next word [3]. This type of model was selected due to its ability to give a score for how likely the next word is which provides a measure of coherence of a sentence. The second reason was that it is a deterministic model which is aligned with the field of explainable artificial intelligence.

KenLM is a variety of N-gram models that is optimised for efficiency [21]. Once trained, the model provided a perplexity score for a sentence. This measured the probability of a sentence, with higher perplexity indicating low probability sentences and vice versa.

To train the KenLM the one billion words dataset was first considered, as it is used as a benchmark for statistical language modelling[22]. The contents of this data was taken from the WMT 2011 news crawl and mostly contained sentences with different sentence structure and language. The second dataset used to train this model was an English subsection of Wikipedia from 2022 [29, 36]. This dataset comprised around 100 million sentences. The Wikipedia trained model proved to be better at determining the perplexity of the generated sentences in preliminary testing, so was using in final testing.

Using this data the model was trained with both a 5 and 3-gram approach. Due to the size of the dataset and resource constraints, it was only possible to train the 5-gram model on a subset of 10 million sentences, and the 3-gram model on 25 million sentences. In preliminary testing it was found that the 5-gram model produced scores that were generally more accurate, even though it had been trained on less data. Due to this the 5-gram model was used in the final testing.

Due to the large perplexity scores produced by the model, typically from 60 to 100 000, the scores were normalised. Log min-max normalisation was used since the scores were so large. When the experiment was run, the minimum and maximum scores were taken from across the whole set run. For example, it was the minimum score produced across all defeasible implication sizes for the simple set (the sets will be explained in more detail in the next section). This was to ensure that the scores could be graphed together and compared since they been normalised with the same minimum and maximums.

6.5 Gemini

Due to the nuanced nature of determining coherence and common sense, Gemini was used to test sentences. In the absence of human testing and with the resources available for this project it was useful to use an LLM for testing.

Using the API setup from creating the sentences, Gemini was prompted to provide a score for each sentence. The prompt provided followed similar design strategies to the original prompt to generate sentences.

In the prompt four questions were asked and a score from 0 to 1 was provided by the LLM. The reason why four questions were used, was to ensure that the LLM considered multiple relevant factors. This maintained consistency across evaluations, since if the model were simply prompted to provided a measure of understandability, it could use different measurements on different knowledge bases. The first question was about grammatical correctness which tested if the model created sentences that followed English grammar. The second question was to check for sentence coherence. Although the sentence by default was logically correct, this evaluated that the two phrases put together were coherent. The third question was about the sentence usability. This is important as a sentence can be logically and grammatically correct, but the aim of the project is to create understandable sentences. The measure of if a person could typically say or use that sentence tried to score more usable sentences higher. The final question was about how related the

parts of the sentence were. This is similar to the second question, but gives a measure of how similar the content of the different parts of a sentence are. This rewards sentences which are more easily understood since they stay on topic.

These four scores were then averaged to provide the final score for each sentence.

7 TESTING AND RESULTS

Tests were conducted on knowledge bases of varying size and complexity. For each knowledge base, both Gemini generated sentences and random phrase allocated manual sentences were created.

The generated and manual sentences were evaluated using SBERT, KenLM and Gemini. These models produced scores for each sentence, which were averaged per rank and used to assess the understandability of the sentences. While the scores are graphed together, the scores from each model represent slightly different meanings, explained in the sections above. The scores for each model also are on different scales, with Gemini between 0 and 1, while SBERT and KenLM could possibly exceed 1 (although in the experiment they do not). For this reason, the results and discussion focus on changes in score, difference between generated and manual sentence scores and variability, rather than directly comparing numerical scores between each model.

7.1 Experiment Setup

7.1.1 Parameter Sets. To generate the knowledge bases needed to run the experiment, two different sets of parameters were used. The first was the simple set, which produced a smaller $\mathcal{K}$ with shorter sentences comprised of only one antecedent and one consequent. There was no transitivity or reuse of the consequence and the sentences only used defeasible and classical entailment as operators. The second parameter set was the complex set. This set used transitivity and consequence reuse and all operators were available for use (and, or, not, if and only if, defeasible and classical implications). It also produced a maximum of two antecedents and two consequents.

The simple set used six ranks and the complex set used eleven sets. The number of ranks was chosen because when complex knowledge bases are generated the final rank often contains short implications or partial formulas (for example $a\&b$). By using eleven ranks, there are ten ranks of standard sentences and one rank of partial sentences.

The two sets of parameters were both used to create five different sized knowledge bases. The simple parameter set generated datasets of 25, 50, 75, 100 and 125 defeasible implications. The complex parameter set generated datasets of 75, 100, 125, 150 and 175 defeasible implications. When creating the knowledge base, the number of defeasible implications is taken as input, but the total number of implications generated could be greater than this. For example, with 75 defeasible implications, 75 defeasible implications are created and 10 classical implications could be generated.

When $\mathcal{K}$ is created, a minimum number of defeasible implications is required. Since the aim of this project is to create understandable knowledge bases, smaller knowledge bases were be tested as these are the most readable for a person. The minimum number of defeasible implications in a knowledge base is determined by the complexity of the parameter set, this is why the simple set starts at 25 and the complex set starts at 75.

A normal distribution was used for both datasets as this produces both large and small ranks, while also replicating how some information in the world is distributed.

Each knowledge base and model evaluation was run three times per parameter set and the results per rank were averaged. These results were used to generated graphs to display changes by rank.

A third parameter set was also used to compare the complex parameter set to the simple one. The only changes to each parameter set was that they both produced 100 defeasible implications and created eight ranks. 100 defeasible implications was chosen since it is within the tested ranges of both parameter sets and eight ranks was chosen as a halfway point between the original number of ranks generated.

The experiment produced ten graphs with scores for the SBERT, KenLM and Gemini models, which evaluated both generated and manual sentences. These graphs are presented in the appendix (Figures 8-17). To analyse these graphs, the following results and discussion use the average of these graphs to evaluate the sentences created.

For the simple and complex set graphs, the legend distinguished between the model used and the type of sentences. The first part of each label indicates the model (SBERT, KenLM, Gemini) and the second part shows the sentence type. For example, SBERTGen is the results for the SBERT model run on generated sentences and GemMan uses the Gemini model to evaluate manual sentences. For the comparison graphs, the first part of the label indicates the parameter set (complex or simple) and the second part indicates the sentence type.

7.2 Simple Parameter Set

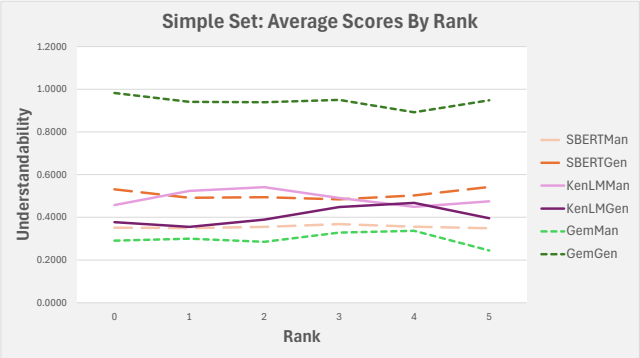

Figure 4: Average Score for Ranks of Simple Parameter Set

Figure 4 shows the average score for each model per rank, run on the simple parameter set. Overall in two of the three models, the generated sentences scored higher than manual sentences.

The generated sentences evaluated by Gemini scored significantly higher than the manual sentences. Starting at a peak of

0.9830, over the next four ranks it gradually decreases, with the largest decrease in the fourth rank to a minimum of 0.8926, before slightly recovering in the last rank. The manual sentences consistently scored the lowest. From rank zero to two it remained relatively stable, slightly increasing to a peak of 0.3369 at rank four before decreasing to its minimum of 0.2444 at rank five.

The generated sentences evaluated by SBERT formed a slight u-shaped curve, with the minimum of 0.4843 at rank three and a maximum of 0.5317 at rank zero. The manual sentences were nearly flat with only a difference of 0.0200 between the minimum at rank five and the maximum at rank four.

The generated sentences evaluated by the KenLM model gently sloped to its minimum at rank one with a score of 0.3550. It then rose to its peak of 0.4681 at rank four, before dipping slightly in rank five. The manual sentences initially rose to a height of 0.5416 at rank two, then slowly dropped to its minimum of 0.4500 at rank four, before recovering slightly in the last rank.

7.3 Complex Set Average

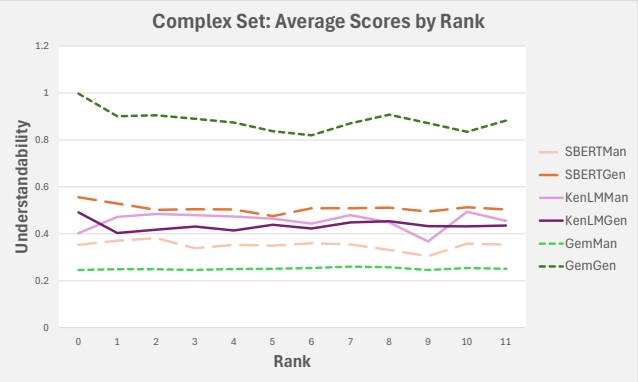

Figure 5: Average Score for Ranks of Complex Parameter Set.

Figure 5 shows averages scores for each rank produced by the evaluation models on the complex parameter set. Overall across the majority of ranks, two out of three of the top scoring models were for the generated sentences.

The generated sentences scored by Gemini dropped from the start moving from a maximum of 0.9972 to 0.8200 at rank six. It then increased over two ranks, decreased for two ranks, before slightly increasing in the last rank. The evaluation of manual sentences by Gemini produced scores which formed a nearly flat line. From the minimum of 0.2458 at the initial rank, there was an incredibly small rise to the peak of 0.2600 at rank seven, before slightly decreasing towards the last rank.

The generated sentences evaluated by SBERT started at a peak of 0.5555 before declining to a trough of 0.4751 at rank five. It then rose slightly in the next rank and remained constant, only dipping slightly at rank nine before recovering. The manual sentences score slightly increased from the start to the second rank with a score of 0.3817. It then dropped in the next rank before stabilising until rank nine where it reached the minimum of 0.3044. It then recovered in the tenth rank and remained stable in the last rank.

The KenLM manual and generated sentences intersected three times. The generated sentences stared at a peak of 0.4917 before dropping to a minimum of 0.4030 in the first rank. It mostly increased until rank eight before gradually decreasing until rank ten before there was a very slight increase in rank eleven. The manual sentences increased slightly in the first rank, then generally decreased to a minimum of 0.3677 at rank nine. It then increased to a peak of 0.4942 in rank ten, before slightly decreasing in the final rank.

7.4 Simple vs Complex Comparison

The following is a comparison of the simple parameter set with the complex parameter set. For a fair comparison, knowledge bases with eight ranks and 100 defeasible implications were created for the evaluation models to score.

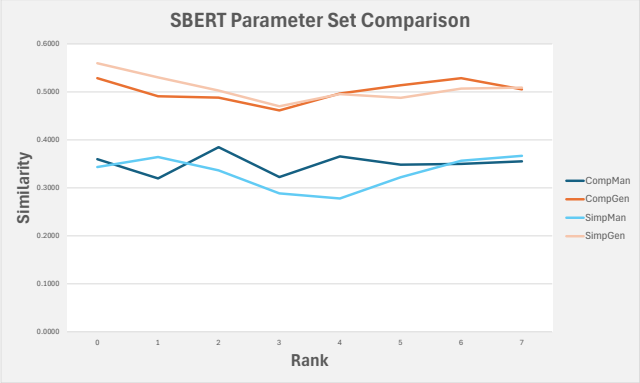

Figure 6: SBERT comparison of Simple vs. Complex parameter sets with manual and generated sentences.

Figure 6 shows that the SBERT evaluated generated sentences from both the simple and complex parameter set out scored the manual sentences.

The generated sentences generally followed the same trajectory in the simple and complex set, starting at a peak initially, before falling to a trough at rank three. They then increased until the seventh rank. The only major difference is that the complex parameter set dipped from the sixth to the seventh rank, where as the simple set rose slightly.

The manual sentences from the simple and complex sets performed similarly on average, but they had different trajectories. The complex set had some volatility, dipping and spiking, before smoothing out across the fifth to seventh rank. The simple data set initially rose slightly in the first rank, before dropping to a trough at the fourth rank. It then recovered and reached a peak in the seventh rank.

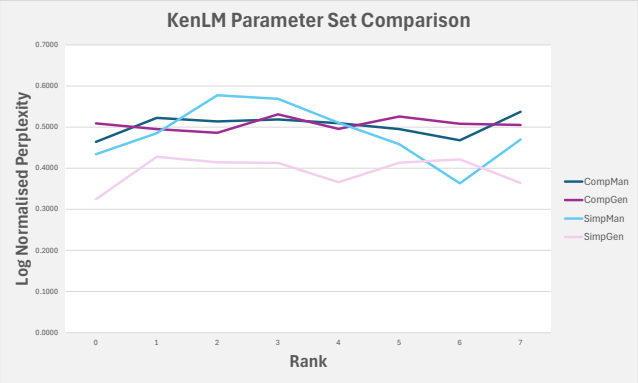

Figure 7: KenLM comparison of Simple vs. Complex parameter sets with manual and generated sentences.

From figure 7 displaying the KenLM evaluation, the perplexity scores calculated show that the complex data set for both manual and generated sentences were very similar, while the simple dataset produced very different scores between the manual and generated sentences.

The generated sentences on the complex parameter set consistently outperformed the simple set. The simple set created an "m" shaped curve, with an initial increase followed by a decrease until rank four. It then increased again until rank six before decreasing in the final rank.

The manual sentences intersected twice, but on average the complex set out scored the simple set. The simple set initially rose steeply until rank two, before steeply falling until rank six. It then recovered in the final rank. The complex set increased slightly, then slowly decreased until rank six, before increasing in the last rank.

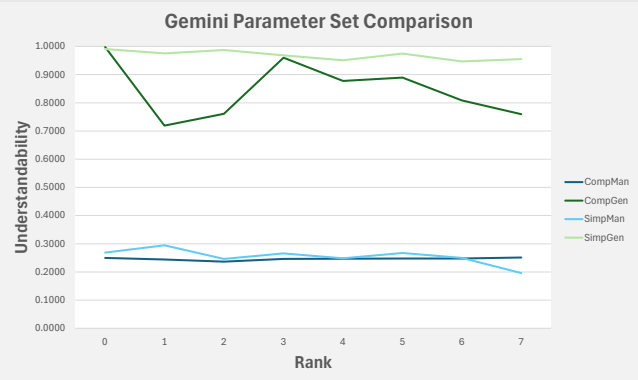

Figure 8: Gemini Evaluation comparison of Simple vs. Complex parameter sets with manual and generated sentences.

In figure 8, showing the evaluation by Gemini, the Generated sentences on both the simple and complex knowledge base scored higher than the manual sentences across all ranks.

The generated sentences on the complex set dropped from a peak of 0.998, to a trough of 0.7610 at rank one. It then almost recovered to its initial score at rank three, before generally decreasing until the final rank. The simple set produced fairly constant scores, forming a generally slightly decreasing line until the final rank.

For the manual sentences, both the simple and complex sets were relatively stable. The complex set was consistent across all ranks with no significant increases or decreases. The simple set initially increased in the first rank, before generally decreasing until rank seven where it dropped below a score of 0.2. The simple set scored higher than the complex set in seven out of eight ranks.

8 DISCUSSION

In both the average rank data and the individual rank comparison, SBERT scored the generated sentences higher than the manual sentences. When looking at the SBERT evaluation of generated sentences for the simple set in figure 4, the data forms a slight u-shape and in figure 6 where the knowledge bases are compared, all the datasets, except for manual sentences on a complex knowledge base, also form general u-shaped curves. This suggests that, due to the implications being normally distributed, when more sentences have to be created at a time the similarity of the sentences decreases. This could be due to longer, more complex sentences using a higher variety or words and phrases, which reduces the similarity. For the manual sentences, when looking at figure 6, it is shown that in the simple knowledge base, similarity scores follow this u-shaped scoring, but the complex set does not. This is an interesting result as it would be expected that the simple and complex sets would follow each other in a similar manner to how the generated sentences follow each other. The complex set likely remains on a similar average across ranks due to reuse of consequence and transitivity. Sentences generated re-use the same phrases and these phrases are used more often due to transitivity, so the scores remain similar no matter the number of implications in a rank. The simple set forming a u-shaped curve is likely due to the wider variety of sentences used in the larger ranks. Since the simple set only has one antecedent and one consequent, and there is no reuse or transitivity, far more different phrases are used to create sentences, causing a lower similarity score.

Overall the KenLM appeared to generally score the manual sentences higher than the generated sentences, meaning that it found that the probability of the manual sentences existing was higher than the generated ones. The manual sentences out scored the generated sentences in almost every rank for the simple knowledge base. Counter to this, on the complex knowledge base, the perplexity scores were very similar, with both generated and manual sentences scoring higher on half the ranks. This may suggest that on larger, even more complex knowledge bases, the generated sentences could start scoring higher than manual ones. However it could also suggest that the generated sentences would score equally to the manual sentences.

The reason for the manual sentences generally scoring higher, is likely due to how the KenLM was trained. Due to resource constrains the largest dataset able to run a 5-gram model was only about ten million sentences. The random sentences used to create the manual sentences were all very simple phrases, often with a casual conversational or writing tone. It is likely that the KenLM was able to recognise most of the words and phrases used in the manual

sentences, so could provide more accurate scores. The generated sentences were made from a larger variety of topics and sentence types, so the KenLM might not have been trained in all the topics used, resulting in lower scores. This can be seen partially in the data in the appendix, where the global minimum scores for manual and generated sentences were comparably similar but the global maximums for the generated sentences were often far lager. These large scores from topics the model was not trained on are likely to have offset the average for the generated sentences.

From the experiment results it shows that Gemini consistently evaluated the generated sentences as being more understandable than the manually created sentences. It was expected that Gemini would score the generated sentences highly, since in the knowledge base generation it was used to form these sentences. The important observation is that it consistently scored the manual sentences much lower than the generated sentences. This strongly supports the hypothesis that Gemini is able to form more understandable sentences than the allocation of random phrases. From both figure 4 and 5, it is shown that the understandability of sentences generally decreases from its initial score, except for in the final rank. This is likely due to the way that the sentences are generated. Each batch is sent separately to ensure logical correctness is kept and to reduce the chance of hallucination from receiving too many implications to change. Even though the rank is provided, for example rank 1 of 7, Gemini flash 2.5 is not able to forward think to be able to accommodate for the next ranks. This means that it is likely to use a greedy technique, where simple, understandable phrases are used first. This means that over the ranks, less common phrases are used and sentences have to be created from these phrases. This causes a decline in sentence quality. In the complex set, the rise in the last rank is likely caused by phrases from earlier being reused to make the partial implications, meaning that the simpler phrases are being used again as starting points for sentence creation. In the simple parameter set, the increase in the last rank is unexpected but it could be caused by there being fewer implications for Gemini to generate sentences from, so it can use more common phrases again.

In the individual graphs (found in the appendix figures 8-17), several anomalies were produced by the KenLM and Gemini models. In contrast, SBERT generally produced reasonable scores, showing lower variation than the other two models. This could be due to SBERT being trained on a larger dataset, so it is less volatile. It could also be from SBERT using a narrower band of scoring, which meant that any anomalies were averaged back to the median.

In Figure 10 for the KenLM with 50 defeasible implications, both generated and manual sentence scores spiked at rank two. While the generated sentences score remained within a normal range, the manual sentences produced the highest perplexity score across all graphs for simple and complex parameter set. The sentences created at this rank were not meaningfully more coherent than at other ranks, so the spike is likely due to instability in the model's scoring, rather than a genuine improvement in sentence quality.

There were several instances where Gemini produced unexpectedly high or low scores, such as in Figure 8 rank four for generated sentences and Figure 9 rank three for manual sentences. In both cases, the sentences produced were not significantly different from those in surrounding ranks. This suggests that these fluctuations in score were due to hallucinations, where Gemini misunderstood sentences and incorrectly scored them.

9 CONCLUSIONS

In conclusion, using the structure of implications from the previous knowledge base generator, this project was able to form sentences using an LLM that were both logically correct and more understandable.

The sentences generated were shown to be much easier to understand than manually allocated sentences. Gemini's evaluation of generated sentences across simple and complex implications scored on average 3.4 times the manual sentences' understandability. SBERT also consistently evaluated generated sentences to be more similar than manual sentences. The KenLM was the only model that did not consistently show the generated sentences were better than the manual sentences.

Although there was no human evaluation in this project which meant that there was no "gold standard" to compare the models with, the convergence of the Gemini and SBERT models supports the conclusion that generated sentences are more understandable than those created with random phrases. This agreement across two models, shows that multiple automated models can provided reasonable estimates of the understandability of sentences.

Using the SBERT and Gemini evaluators, the English sentences were shown to be understandable using both simple and complex implications as well as being understandable across small and large knowledge base sizes.

There are several extensions and improvements that can be made to this project. The first improvement is that sentence generation can be improved as newer models are released. In preliminary testing, newer models, which currently have no free tier, performed significantly better at creating sentences that made sense. Over the next few years it is likely that new, improved, free tier models will be produced which can replace the model used in this project. The second improvement could be where instead of general sentences created, specific topics can be selected. This is more limiting but it would allow evaluating models to be trained and instructed better, since the topics it needs to be trained on and understand are more specific. The third improvement could be for testing, where, beyond human trials, a hybrid approach could be taken. This would take the form of combining multiple model scores to give a single "understandability score".

REFERENCES

[1] Bailey, A. Scalable Defeasible Reasoning, 2021.

[2] Ben-Ari, M. *Mathematical logic for computer science*. Springer Science & Business Media, 2012.

[3] Brown F, deSouza P, M. R. P. V. L. J. Class-Based n-gram Models of Natural Language. *ACM Digital Library 18*, 4 (1992).

[4] Casini, G., Harrison, M., Meyer, T., and Swan, R. Arbitrary ranking of defeasible subsumption. *CEUR* (2019).

[5] Casini, G., Meyer, T., and Varzinczak, I. Taking defeasible entailment beyond rational closure. In *Logics in Artificial Intelligence: 16th European Conference, JELIA 2019, Rende, Italy, May 7–11, 2019, Proceedings 16* (2019), Springer, pp. 182–197.

[6] Cer D, Yang Y, Kong S, Hua N, Limtiaco N, John R, et al. Universal sentence encoder. In *EMNLP demonstration* (Brussels, Belgium, 2018).

[7] Cotterrell, J. A User Interface to Aid in the Understanding of Rational Closure. Honours Project, University of Cape Town, 2025.

[8] DeepMind, G. Gemini Models. https://ai.google.dev/gemini-api/docs/models#gemini-2.5-flash, 2025. Accessed: July 2025.

[9] Gallier, J. H. *Logic for Computer Science Foundations of Automatic Theorem Proving*. Universtity of Pennsylvania, 2003.

[10] Giordano, L., Gliozzi, V., Olivetti, N., and Pozzato, G. L. Semantic characterization of rational closure: From propositional logic to description logics. *Artificial Intelligence 226* (2015), 1–33.

[11] Golan T, Siegelman M, K. N. B. C. Testing the limits of natural language models for predicting human language judgements. *Nat Mach Intell 5* (2023), 952–964.

[12] Google. universal sentence encoder. https://www.tensorflow.org/hub/tutorials/semantic_similarity_with_tf_hub_universal_encoder, 2024. Accessed: July 2025.

[13] Google. Gemini API Quickstart. https://ai.google.dev/gemini-api/docs/quickstart, 2025. Accessed: July 2025.

[14] Google. universal sentence encoder. https://www.kaggle.com/models/google/universal-sentence-encoder/tensorFlow2/universal-sentence-encoder/2?tfhub-redirect=true, 2025. Accessed: July 2025.

[15] Google AI. Prompt Design Strategies. https://ai.google.dev/gemini-api/docs/prompting-strategies, 2025. Accessed: July 2025.

[16] Gounden, N. Investigation of Optimisation Techniques for Rational Closure in Defeasible Reasoning. Honours Project, University of Cape Town, 2025.

[17] HuggingFace. sentence-transformers/all-minilm-v6-v2. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2, 2025. Accessed: July 2025.

[18] HuggingFace. sentence-transformers/multi-qa-mpnet-base-dot-v1. https://huggingface.co/sentence-transformers/multi-qa-mpnet-base-dot-v1, 2025. Accessed: July 2025.

[19] J.D, M. The contributions of Alfred Tarski to algebraic logic. *Journal of Symbolic Logic 51*, 4 (1986), 899–906.

[20] Kaliski, A. An overview of KLM-style defeasible entailment. *NA* (2020).

[21] Kenneth H, Pouzyrevsky I, C. J. K. P. Scalable Modified Kneser-Ney Language Model Estimation. 2013.

[22] Koehn P, Robinson T, C. C. M. T. S. M. G. Q. e. a. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. *Google Research* (2013).

[23] Kraus, S., Lehmann, D., and Magidor, M. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial intelligence 44*, 1-2 (1990), 167–207.

[24] Lang, A. Extending Defeasible Reasoning Beyond Rational Closure, 2022.

[25] Lehmann, D. Another perspective on default reasoning. *Annals of mathematics and artificial intelligence 15* (1995), 61–82.

[26] Lehmann, D., and Magidor, M. What does a conditional knowledge base entail? *Artificial intelligence 55*, 1 (1992), 1–60.

[27] Levesque, H. J. Knowledge representation and reasoning. *Annual review of computer science 1*, 1 (1986), 255–287.

[28] Lie S, Zeng S, L. S. Evaluating Text Coherence at Sentence and Paragraph Levels. *arxiv* (2020).

[29] May, P. Wikipedia 2 Corpus. https://github.com/GermanT5/wikipedia2corpus/blob/main/README.md, 2022. Accessed: August 2025.

[30] Reimers, N., and Gurevych, I. Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing* (2020).

[31] Reimers N, G. I. *Sentence-BERT: Sentence Embeddings using Siamese BERT Networks*, 2019.

[32] Sadiki, M. A Study of Parameters and Optimization Strategies in Defeasible Knowledge Base Generatio. University of Cape Town, 2024.

[33] Speer R, Chin J, H. C. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In *AAAI31* (2017).

[34] Tarski, A. On the concept of logical consequence. Logic, semantics, metamathematics. In *Logic, Semantics, Metamathematics*. Hackett Publishing, 1956.

[35] The Word Finder. Random Sentence Generator. https://www.thewordfinder.com/random-sentence-generator/, 2025. Accessed: July 2025.

[36] Wikimedia. data Downloads. https://dumps.wikimedia.org, 2025. Accessed: Auguest 2025.

A RESULTS TABLES

Table 1: Average Scores from Simple Parameter Set

Rank	SBERTMan	SBERTGen	KenMan	KenGen	GemMan	GemGen
0	0.3516	0.5317	0.4572	0.3772	0.2907	0.9830
1	0.3500	0.4916	0.5240	0.3551	0.2998	0.9412
2	0.3552	0.4946	0.5416	0.3898	0.2853	0.9392
3	0.3682	0.4843	0.4910	0.4481	0.3282	0.9508
4	0.3559	0.5025	0.4496	0.4681	0.3369	0.8926
5	0.3488	0.5426	0.4752	0.3960	0.2444	0.9482

Table 2: Average Scores from Complex Parameter Set

Rank	SBERTMan	SBERTGen	KenMan	KenGen	GemMan	GemGen
0	0.35305333	0.55548	0.40250241	0.49167705	0.24583333	0.99722222
1	0.37074	0.52896	0.47229504	0.40299945	0.24888889	0.9014127
2	0.38166667	0.50183333	0.48501403	0.4177067	0.2492601	0.9048532
3	0.33847333	0.50461333	0.47994466	0.43064834	0.24563528	0.89031729
4	0.35303334	0.50413333	0.47411961	0.41422088	0.25019162	0.87434008
5	0.34932667	0.47513333	0.46454302	0.43874077	0.25048181	0.83705753
6	0.36017334	0.50909333	0.4431519	0.42286629	0.25399702	0.81979555
7	0.35489333	0.50924	0.47954729	0.44875752	0.25994643	0.87106833
8	0.33122667	0.51148667	0.44850143	0.45339563	0.25766667	0.90741865
9	0.30442667	0.49504	0.36767749	0.43227175	0.24583333	0.87138889
10	0.35794	0.51315333	0.49422275	0.43146333	0.25459524	0.83470873
11	0.35386667	0.50373333	0.45559269	0.43497707	0.25112088	0.88268938

Table 3: Comparison Between Simple and Complex Parameter Set

	SBERT				KenLm				Gemini			
Rank	CompMan	CompGen	SimpMan	SimpGen	CompMan	CompGen	SimpMam	SimpGen	CompMan	CompGen	SimpMan	SimpGen
0	0.3601	0.5289	0.3434	0.5601	0.4640	0.5090	0.4340	0.3247	0.2500	0.9979	0.2688	0.9906
1	0.3197	0.4911	0.3645	0.5306	0.5225	0.4951	0.4851	0.4277	0.2444	0.7190	0.2944	0.9755
2	0.3848	0.4884	0.3367	0.5030	0.5141	0.4861	0.5777	0.4143	0.2365	0.7609	0.2461	0.9875
3	0.3225	0.4615	0.2888	0.4703	0.5187	0.5310	0.5690	0.4130	0.2462	0.9602	0.2661	0.9684
4	0.3655	0.4967	0.2781	0.4956	0.5098	0.4958	0.5104	0.3663	0.2472	0.8778	0.2483	0.9506
5	0.3487	0.5143	0.3223	0.4880	0.4954	0.5258	0.4584	0.4133	0.2479	0.8896	0.2672	0.9745
6	0.3499	0.5288	0.3568	0.5069	0.4678	0.5082	0.3633	0.4215	0.2479	0.8083	0.2500	0.9469
7	0.3553	0.5055	0.3667	0.5092	0.5375	0.5054	0.4700	0.3641	0.2513	0.7596	0.1960	0.9552

B SIMPLE PARAMETER SET GRAPHS

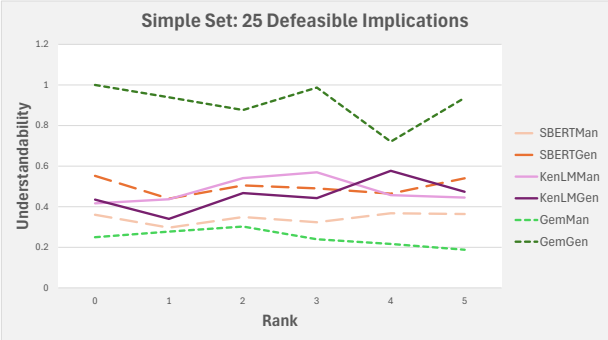

Figure 9: Simple parameter set with 25 defeasible implications

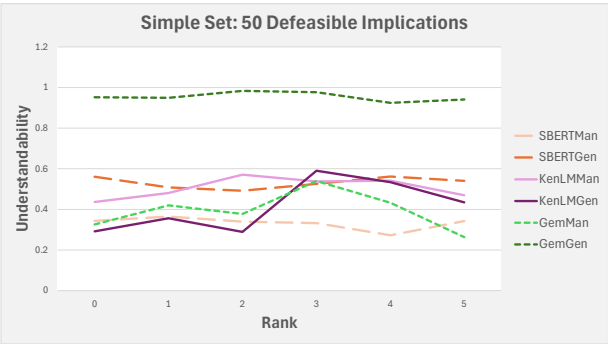

Figure 10: Simple parameter set with 50 defeasible implications

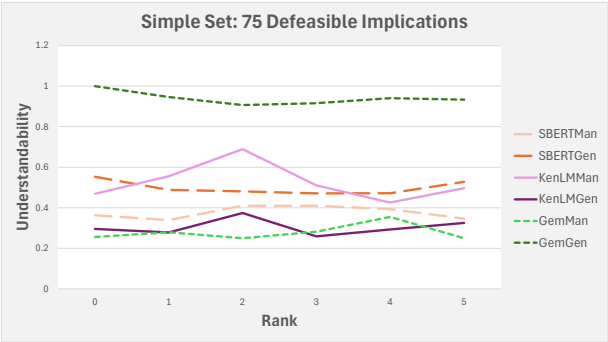

Figure 11: Simple parameter set with 75 defeasible implications

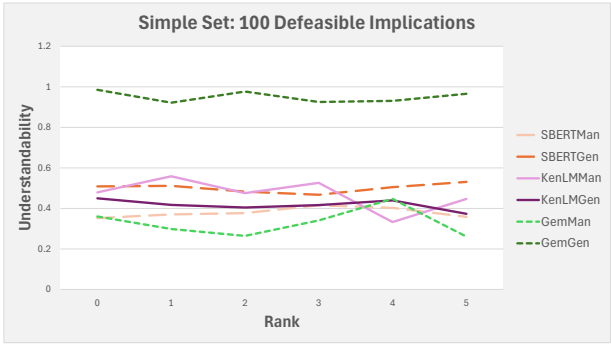

Figure 12: Simple parameter set with 100 defeasible implications

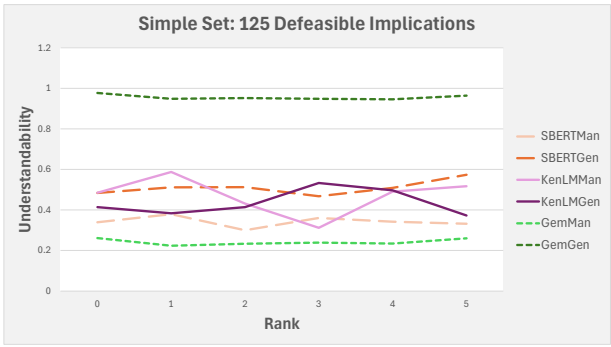

Figure 13: Simple parameter set with 125 defeasible implications

C COMPLEX PARAMETER SET GRAPHS

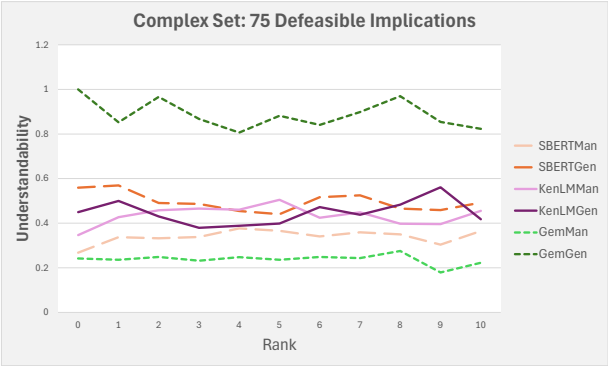

Figure 14: Complex parameter set with 75 defeasible implications

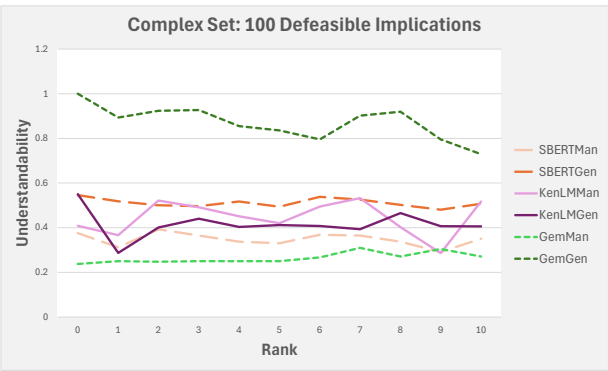

Figure 15: Complex parameter set with 100 defeasible implications

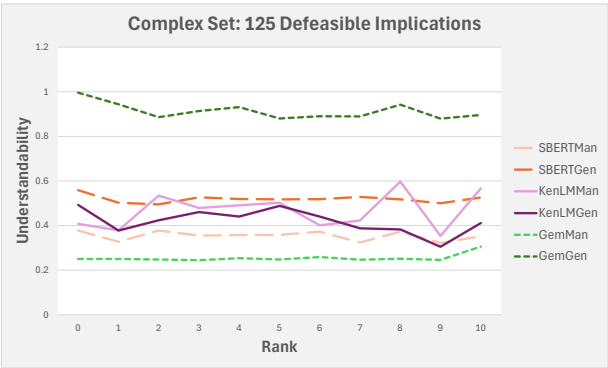

Figure 16: Complex parameter set with 125 defeasible implications

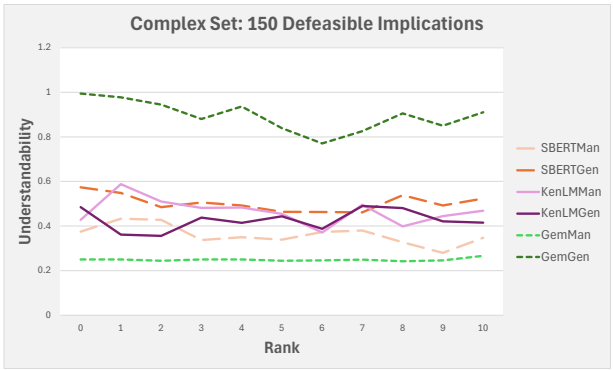

Figure 17: Complex parameter set with 150 defeasible implications

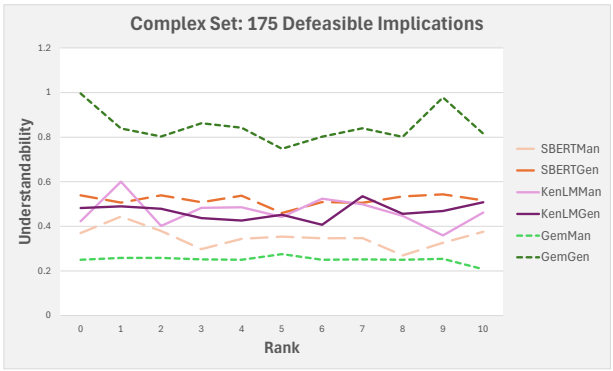

Figure 18: Complex parameter set with 175 defeasible implications