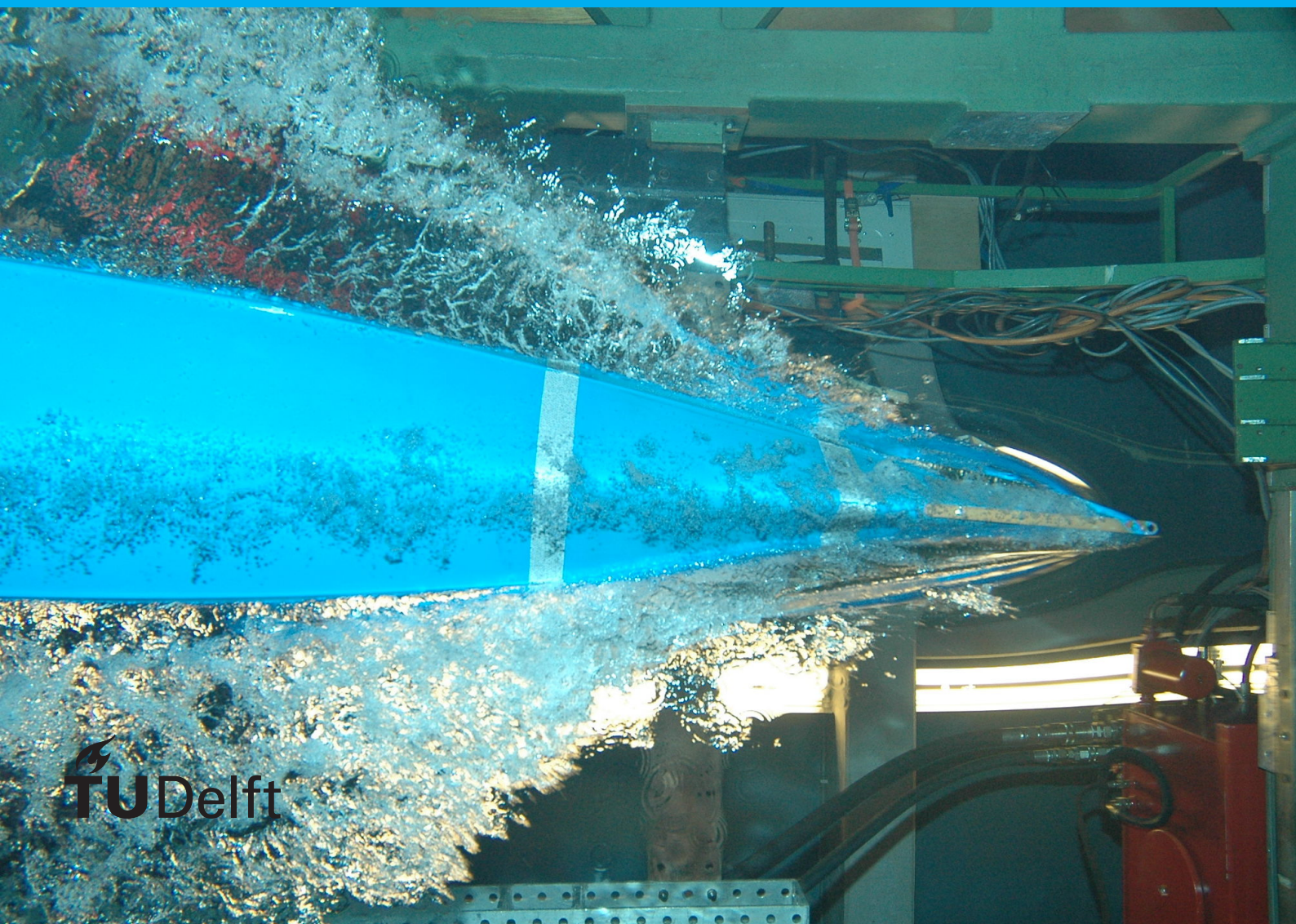
Characterizing the knowns and unknowns of text simplification models

Siwei. Wang



Characterizing the knowns and unknowns of text simplification models

by

Siwei. Wang

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday August 23rd, 2022 at 10:00 AM.

Student number:	5239982	
Project duration:	November 23rd, 2021 – August 23rd,2022	
Thesis committee:	Prof. dr. ir. G.J.P.M Houben,	TU Delft, Thesis advisor
	Dr. J. Yang,	TU Delft, Daily supervisor
	Dr. Luís. Cruz,	TU Delft

This thesis is confidential and cannot be made public until 23rd August.

TU Delft

Preface

Last year I enrolled in a course in Delft called Information Retrieval, which is a combination of two sections: information retrieval and natural language processing. Dr. Jie Yang is responsible for the NLP(natural language processing) part, and I became increasingly interested in such an area. Jie's teaching style helped me immensely through this course and sparked my curiosity. So I wrote an email to Dr. Jie for suggestions about my thesis; Jie provided me with an idea of interpreting an NLP-based model, which combined several subjects I have been interested in for a long time. So my journey with characterizing text simplification models knowns and unknowns started. This thesis project combined the local interoperability and human-in-the-loop method to interpret the text simplification model. It can identify model unknowns, and therefore we can predict when the text simplification model will fail.

The journey is combined with tears and joy; on the one hand, the depth of such a project is beyond my imagination. I had to read many papers and re-implement codes to understand and thus get inspired. Sometimes, deadlines and weekly meetings push me to work all night long; people who love me give me the strength not to give up and move forward. On the other hand, by solving one and another difficulty, I feel great joy. Also, Jie and Lorenzo's praise and affirmation give me much confidence and make me proud. Since this journey is about to end, I want to thank everyone who helped me.

Honestly, I want to thank my supervisor Jie and Lorenzo, for their excellent guidance and help during this period. Their advice helped me during all research time, implementing the research idea, writing the thesis, and preparing the final defense. I want to express my warm gratitude to my direct advisor Prof. dr. ir. Geert-Jan Houben for chairing the defense and helping with preparing various forms and guidance. I also want to thank Dr. Luís Cruz for being a committee member. Besides, I want to say thank you to my family, thank you for supporting me. Thanks to all my friends who have contributed their time to complete annotation work, mainly thanks to my boyfriend, who always encourages me to tackle the challenge.

Siwei. Wang
Delft, August 2022

Contents

1 Introduction

In the Netherlands, approximately 2.35 million inhabitants between the ages of 16 and 65 are people with low education levels and second language readers[1]. It is difficult for them to know what an advertisement is about, to gain knowledge about any subject or even to understand the meaning of posts on social media. Thus, the complex text is causing trouble in the lives of these people.

For example, the National Library of the Netherlands(KB), is one of the largest cultural heritage institutions and is responsible for providing data available to all Dutch citizens. There are millions of materials and nearly 120 kilometres of books. If the public cannot obtain these materials, KB will not be able to fulfil its responsibilities. Therefore, a simple text is needed to help everyone understand these materials and a simple text may improve their experience and quality of life[36]. In order to facilitate access, KB has tried to simplify the text manually. However, there are too much data in KB, so manual rewriting is laborious and costly. So, adopting a more efficient way to simplify text is urgent. Thus, we can ask Artificial intelligence for help.

In the field of natural language processing, there is a task named Text Simplification. Text simplification lessens the text's complexity to make it easier to read and comprehend while keeping the original information's content[2]. There are various algorithms to do so, however, these methods are not good enough to apply in the industry. Even the state-of-the-art methods make mistakes and tend to fail in the following situations: (1). Sometimes models delete important words which may change the original meaning of the sentence. (2). The model may replace a complex word with a simple word, however, the changed word may bring a different meaning to the original sentence. (3). Additional words may also be added to a simplified sentence, as to help readers to understand in a more easy way, but this may bring confusion to the sentence.

If we want to build better text simplification models, it is essential to first understand the situations in which state-of-the-art methods fail. Therefore, if we can figure out why the model failed, We can find ways to improve it to get a better model. we also find that interpretable machine learning is a way to address this problem. More specifically, it is a method that helps us know how the model gives this output. When applied to the text simplification task, it is a way to tell which words play a more important role in the model's simplification. Therefore, we can use the method to describe the interior of the black box model and to look for in what kind of situation the model fails.

Interpretable machine learning provides explanations based on some artificial intelligence methods. But some of its explanatory forms are not easy to understand by humans. In our task, the explanation form generated by Interpretable machine learning methods is complex-simple word pair. Hence, it is hard to infer all simplified operations from the generated word pairs since some operations involve multiple words. Here we would like to bring human intelligence into the loop of interpreting models. Several works have combined human-in-loop and interpretable machine learning methods to better interpret the model. Balayn et al[9] recently proposed a human-in-the-loop pipeline named SECA to interpret models in the image classification tasks, where human involvement provides a richer semantic interpretation. In addition, humans can help to contribute knowledge that models should learn.

Previous work has targeted the field of computer vision and focused on classification problems. Little work has been done on the language generation task. This research proposes a human-in-the-loop framework combined with a machine learning interpretability method, which can analyze what the black-box model knows and what the model should know and then extract the model-unknown knowledge. By gaining such knowledge, we can clearly understand in which circumstances our model fails and why our model fails.

1.1. Problem statement

Under the incentives of these emerging fields, some state-of-the-art text simplification methods do have good results by automatically evaluation metrics.

However, it is still not good for human evaluation, and these methods are not good enough to apply in the industry. We can take an example to explain this situation: The original sentence is **They are culturally akin to the coastal peoples of Papua New Guinea.**, and the model simplified it as **They are a lot like the coastal peoples of Papua New Guinea.**. This simplification gained a high score by automatically evaluation metrics. But the meaning of the original sentence is changed: it loses the sense of *culture*.

Therefore, we aim to know why our model fails and what kind of situation my model fails thus to gain a better text simplification model. With the deepening of the research, the critical problem we need to solve is **To what extent can our framework identify and characterize the Text simplification model's unknowns**. And for this, we can also study and discuss the following three aspects.

1. How can we know what the text simplification model really knows?
2. How can we know what the text simplification model should know?
3. how effective are rule-based unknowns for predicting model behavior on out-of-distribution data?

This research will generally explore why and under what circumstance the text simplification model fails. The process implemented by this pipeline is as follows: First, take the interpretable machine learning and a human-in-the-loop approach to extract what the model really knows. Then, human intelligence is used to gain what the model should know. Finally, we calculate what model does not know by the Really-Know part and the Should-Know part. Figure 1.1 displays the overall idea of our framework.

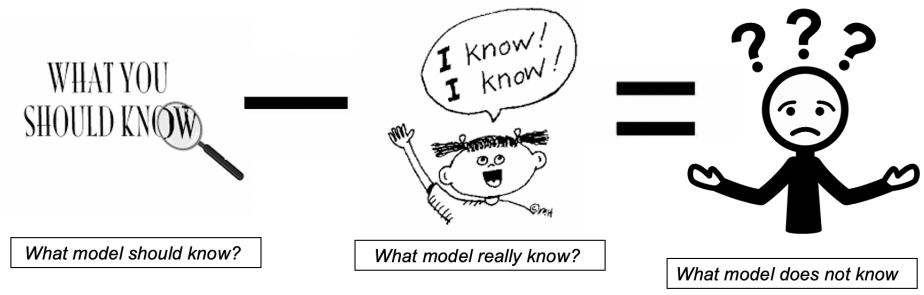


Figure 1.1: The proposed framework to collect what the model does not know

1.2. Challenges

Achieving the purpose of characterizing the unknowns of the text simplification model is challenging in several ways. The challenges are as follows:

1. **Hard to explain model behaviour only by Interpretable machine learning methods**

   Text simplification can be divided into two types: lexical simplification and syntactic simplification[2]. The most widely used is the interpretable machine learning methods, which operate mainly by exploiting its important input features[41]. In other words, it provides connections between the original and generated simple words. Thus these generated links can be used to specify lexical changes, but not enough to describe syntactic changes. And syntactic changes usually refer to the changes in syntactic structure, so this change often involves more than one word. However, relationships generated by interpretable machine learning methods are generally one-to-one correspondence, so We can not infer from one-to-one generated word pairs. Therefore, interpretable machine learning methods are not enough to explain the model behavior.

2. **Hard to explain model behaviour only by human intelligence**

   We will describe the model behavior by designing a human intelligence task. We all know that text simplification is the simplification of complex sentences through synonym replacement, syntactic structure transformation, and the removal of irrelevant information into simplified sentences. So even

though interpretable machine learning can provide information about simple words that are generated and their relevant original words, figuring out how sentences are still a huge burden for humans, the descriptions they offer have no exact format, so it is hard to extract their descriptions for the model behavior.

3. **Difficult to evaluate the result from the framework**

   So far, no framework has been proposed in this field, so it is hard to prove the effectiveness of the proposed method without comparative analysis with other approaches. In addition, the output of this framework is a set of rules representing the unknowns of the model, but it is hard for us to prove the credibility of the unknown rules. For example, there is some work done on image classification, but it is still challenging to apply it to our task. In this case, we can add some fake noise to see whether the framework can extract the added noise as a piece of unknown information. By analogy, the state-of-the-art method for our task is to adopt a pre-trained language model, but the technique requires a lot of time for training, so we can hardly apply this method in the field of computer vision. So the only way is that We have to come up with another effective way to evaluate our framework.

1.3. contributions

Current research mainly focuses on characterizing model knowns and unknowns in the classification task. The framework proposed in this research is one of the pioneering work to try its best to extract the unknowns in language generation tasks. Besides, the framework combines the interpretability method and human intelligence to explore what is inside the model and what the model should learn. Therefore, we can use our framework to predict what kind of situation the model will fail and to explain why the model fails. In addition, we also adopt knowledge from the framework to reduce human cognitive load and help them to provide better explanations. The contribution can be described in detail as follows:

1. **A novel human-in-the-loop interpretability framework to describe model knowledge and for specifying domain knowledge**

   In this thesis, we propose and design a novel framework to describe what the model knows and should know. It is a framework that combines human intelligence and the interpretable machine learning method to describe model behavior. It can specify both lexical and syntactic simplification operations. Thus, this method minimizes the deficiency of only using the machine learning interpretability method.

2. **A method to use rules from the knowledge base to better explain model behaviour**

   We adopted rules from the PPDB (a simple paraphrase database[32]) to reduce human burgeon and to help humans better describe how the model simplifies the sentences at both lexical and syntactic levels. The PPDB knowledge base[32] contains a set of lexical rules and syntactic rules which users can use to describe the changes as to reduce human cognitive load. In addition, the rules can also provide the same format for crowd workers to follow.

3. **A computational method to evaluate model unknowns**

   The method mentioned above is to extract the really know, and the should know of the model by combining the framework, then calculate the model unknowns by calculating the difference set of the Really-Know task and the Should-Know task. From this, we take an approach to verify whether our framework is practical. More specifically, the method is that we predict the failure cases in the test set from the unknowns extracted from the validation set. If we can predict most failure cases, we can know that our method is accurate and practical.

1.4. Organization

The following sections of the thesis report are organized: In Chapter 2, we will provide background information on the three main topics in our thesis: Text simplification, Interpretable machine learning, and Human-in -the-Loop Machine learning. Next, we will illustrate the main framework for extracting model unknowns. Then, we will describe the methodology of the first task in our paper: what the model really knows. Chapter 5 is going to explain what models should know. The experiment setup and the results will be presented in Chapter 6. Finally, conclusions are drawn, and future work has prospected.

2

Background Information

This report presents how model interpretability technology and human explanation can be used to explain what our text simplification model knows and collect information about what our model should know, and then extract the unknowns to predict when the model fails. In order to help readers have a more comprehensive understanding of our work, we research the background information of the following three research areas: text simplification, machine learning interpretability and the human-in-the-loop method.

Firstly, it will introduce the definition of text simplification and then the classification of text simplification should be summarized. After that, we are going to illustrate several typical text simplification methods. Then, some interpretability methods for interpreting language models discussed. The next step is to illustrate existing work on how human intelligence can help us to interpret language models.

2.1. Text simplification

Text Simplification(TS) is a task of reducing the linguistics of a sentence while preserving its original meaning[2], here is an example as shown in table 2.1. There are so many reasons why such a task is needed. Firstly, it can help people with low literacy, children, and non-native speakers to understand the text of a sentence. Besides, People with reading comprehension problems like autism, aphasia, dyslexia, and deaf people also benefit from simple text[2]. In addition, TS is a pre-task of several natural language processing tasks like question generation and information extraction[56]. Thus, these natural language processing tasks can also benefit from TS[45].

Examples of Text Simplification	
Example of a complex sentence	Grammarly provide assistance in order to optimize users' communication
Example of a simple sentence	Grammarly helps people communicate

Table 2.1: Examples of Text Simplification

TS can be divided into two sub-tasks, Lexical Simplification and Syntactic Simplification[2]. Lexical simplification focuses on the word level while syntactic simplification focuses on the sentence level. Lexical simplification is going to replace complex words with simple synonyms while Syntactic simplification will replace the complex syntactic with a simple structure[2].

Existing technologies can be divided into two groups: the first is rule-based method. It can simplify sentences by the extracted lexical and syntactic rules. Besides, the second is data-driven methods. Those approaches extract knowledge from large datasets. To comprehend and analyze data, it applies methods from numerous disciplines, including computer science, mathematics, and statistics [2].

2.1.1. Rule-based text TS methods

The first rule-based TS system was proposed in 1998[11]. This method is aimed to simplify English newspaper texts to assist aphasic readers. The system consists of a simplifier and an analyzer. The analyzer is used to analyse the syntactic information of the sentence and the simplifier aims to simplify the text using the information provided by the analyser[11]. The simpler simplified words by the following steps. It adopts WordNet

[28] to generate synonyms and then replaces the original word with the highest frequency of synonyms based on the Oxford Psycholinguistic Database[2]. However, the limitation of this approach is so obvious that sometimes will lose its original meaning.

While the method mentioned before pay more attention on lexical simplification, the following research presented on appositives, coordination clause separation, and structures like relational clauses and appositives[2]. The widely known method is to simplify the syntactic structure by manually formulationg rules. Consider the following rule: $X : NP, RELPRON\ Y, Z. X : NP\ Z. X : NP\ Y$. If a sentence begins with a noun phrase (X) followed by a relative pronoun (RELPRON) with the pattern Y followed by Z, where Y and Z are word sequences, then the embedded clause can be broken down into two sentences, namely the sequences X followed by Z and the X followed by Y[2]. Following this rule, the Sentences can be simplified. In actual use, the system is not always extremely effective. It sometimes fails in long-distance sentences and ambiguous sentence[2]. Biran et al [10]proposed YATs, which focused on both lexical simplification and syntactic simplification. For the lexical part, it adopts a vector space model to calculate the most possible meaning of the given word in the context and then rank the possible alternatives according to their word simplicity and word frequency. For the syntactic structure, it adopts part of the speech tagging and syntactic dependency tree to simplify the original complex sentence.

2.1.2. Data-driven TS methods

The traditional rule-based TS methods are using hand-crafted rules and then apply these rules to the TS corpora, while the data-driven methods extract knowledge from the aligned corpus[2]. Horn et al.[18] to learn simplification knowledge from the aligned Wikipedia datasets using GIZA++. They trained a feature-based ranker by utilizing SVM in order to choose the best candidate in a specific situation. A series of features are used to represent the best candidate words, such as candidate alignment likelihood, word frequency, language model, and context frequency. From this we can conclude that The proposed system produced positive outcomes.

The following talks about TS methods who solved the problem as mono-linguistic machine translation, Zhu et al[55] proposed a probabilistic, Tree-based Simplification model which firstly adopted statistical simplification models and simultaneously covered the four simplification operations: split, delete, reorder, and replacement. This paper gives a more comprehensive simplification steps that considered by researchers, but it still does not achieve a good result in terms of BLEU and NIST(evaluation metrics of TS). Sander et al.[50] follow Zhu et al [55] work which also treats the complex-simple sentence pairs as the input of a mono-linguistic machine translation tasks, but they do not explicitly take the syntactic information into account, and instead they learn the transformation of syntactic implicitly. Besides, this method pay more attention to the difference between two sentences than to the drop operation.

This paragraph describes a method of TS from sequence to sequence structure, which is widely used in machine translation. The first method was introduced by Sergiu et al [30] in 2017.They use sequence-to-sequence neural network to replace the complex words for TS [30], and it also adopts human evaluation to prove the effectiveness of the method.

After the introduction of the transformer-based architecture, Zhao et al [54] proposed a method that inherited from the machine-translation method and combines a transformer-based architecture. They demonstrated that the transformer-based architecture has the ability to understand the text by selecting the complex word in the original text and then replacing it with a simpler word, while also it shows the effectiveness of the database[54].

2.2. Interpretable machine learning

Even using the state of the art methods for TS, results are still not good. From the test set, we can find that model is prone to failure in some certain situations. Therefore, the existing state of the art methods of text simplification still needs to be refined[35]. Our solution is first to find out what model has learnt. So, here we need to take interpretable machine learning methods to show what is inside our model[12].

Interpretable machine learning can described as methods and models that make the behaviour and predictions of machine learning systems understandable to humans[23]. In other worlds, Interpretability refers to the extent to which a human can consistently predict the model's result[20]. The more interpretable a machine learning model is, the simpler it is to understand why particular judgments or predictions have been made[29]. Therefore, it can help us figure out what the model really know and then learn why the model fails.

Interpretable machine learning can be categorized into two groups: intrinsic interpretability method and

post-hoc interpretability method[15]. The difference between the two methods is the time when we use the interpretability method. The intrinsic interpretability method is like combined the interpretability structure with the model like the attention-based model and decision tree[15]. In contrast, Post-hoc interpretability method is like we need another method or model to provide an explanation for the existing model. In addition, baseing on the categorization provides here, each group still can be categorized into two types: global interpretability and local interpretability[49]. Global interpretability means that users can know how the model behalves on a global view while local interpretability means that users can understand why the model gives this prediction for one specific example[15]. Figure 2.1 displays the category of interpretable machine learning.

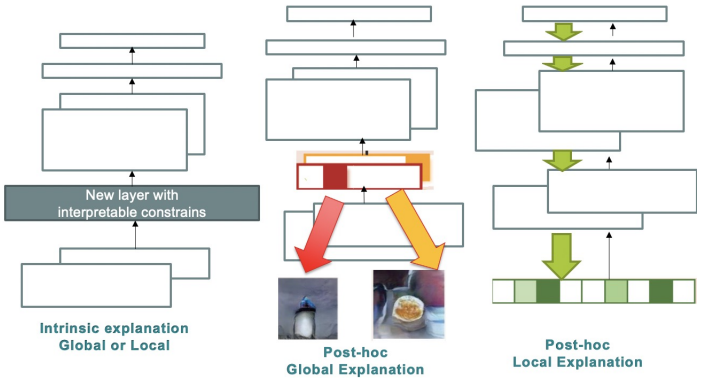


Figure 2.1: The classification of Interpretable machine learning method

2.2.1. Intrinsic interpretable method

Intrinstic interpretable method refers to machine learning models that are considered interpretable due to their simple and straightforward structure, such as short decision trees or sparse linear models[29]. More specifically, For linear regression models, the importance of each feature is easily known by the weights or the coefficients. Besides, a decision tree is another good example because we can access the partition of each feature. Moreover, it is easy for us to know how the decision is made.

There is another typical interpretability method: attention-based mechanism, which is widely used in the explanation of sequence to sequence pattern prediction mode(eg. recurrent neural network)[15]. The benefit of the attention mechanism is that it enables the users to understand which aspects of the input of the model focuses on by displaying the attention weight matrix for a specific predictions[15]. This mechanism can be used in the field of computer vision,Xu et al.[39] proposed a method to automatically generate descriptions for images, and when generating each word [39], the model shifts its attention to the relevant regions reflecting the image. In addition, the attention mechanism can be used in the field of machine translation[8]. More specifically, the attention mechanisms are used to improve the performance of Neural Machine Translation by selectively focusing on sub-parts of the sentence during translation[33].

2.2.2. Post-hoc Interpretability method

Post-hoc Interpretability method can also be used to describe what the input tokes model looks at when generating the output. The following is an example that shows when applying saliency score to the machine translation task, the darker colour means that the input token is more important when generating the specific output. The post-hoc interpretable method analyses the model after training [29]. It also can be divided into two categories: one is global model interpretable methods, and the other is local model interpretable methods[29].

The global model interpretable methods explain the general model behavior towards a given dataset. The working process of the global model interpretability method can be described as follows: Starting from training, the model can learn much knowledge from the data and store them in the parameters. Then the interpretable approach provides a global understanding of what the knowledge-based model has learned. There is another typical method named permutation importance[4], which is a method to calculate the importance of one feature by the degree of the accuracy of the model changed when the feature is removed. This method specifically measures the correlation between a set of features and the output result[4]. This algorithm is

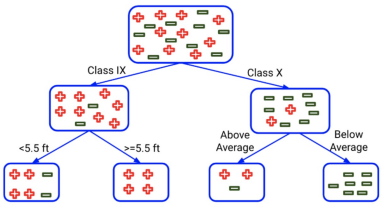


Figure 2.2: Decision tree

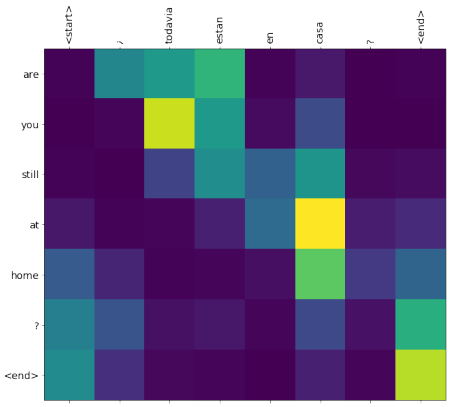


Figure 2.3: Word Alignment

done for each component, and the final prediction scores are obtained for each N feature, respectively. Then the importance of the N features is ranked according to the reductions of their score compared to baseline accuracy[15]. This method has several advantages:

1. It can preserve the relationship between each feature [4]
2. This method is unified and can be applied to any model as long as it can generate feature importance [4].

Post-hoc locally interpretable method emphasizes explaining individual predictions of Ml models. This type of interpretability method is what we focus on most in this paper. There are two types of post-hoc interpretable methods that are most commonly used:

1. **Counterfactual explanations**

   The counterfactual explanation is a way to see what would happen if the input changed in a particular way[46]. The working process can be described as follows: The counterfactual explanations adopt what-if Serino to give explanations. Here, we can tell the predicted outcome of an instance as an event, and then we describe the particular eigenvalues of this instance which caused the "event" as the “cause.” Both feed them into the model, and we can get a specific prediction. We then change the "causes" to analyze how the result of prediction changes[29]. The advantage of this method is that it provides good explanations since the explanations are contrastive and focus on a few reasons[29].

2. **Saliency map**

   It is an interpretable method, and saliency methods are widely used in language processing and computer vision. Usually, we use an image as input, then use the information to predict the output. For example, if we use an image to indicate the category of a bird, we do not need to consider all the pixels of the input. Therefore, we only need some necessary pixels from the original picture to give the output. The saliency map allows us to find the critical pixel from the input image. The following figure shows an example; Figure 2.4 shows the input picture, in which the model identified as 26% to be a sheep and 17% to be a cow, and other figures show what the model looks at when identifying it as a sheep or cow. Ways to implement a saliency map can be divided into two categories: Gradient-based and Perturbation-based. The following paragraphs briefly describe typical methods for implementing saliency maps.

   - **Gradient-based approach**

     The gradient-based methods compute the gradient of the prediction with respect to the input features[29]. Many types of gradient-based methods mainly differ in calculating gradient[29]. Then three gradient-based methods will be explained:

     - **Gradient**

       The idea of Vanilla Gradient was first proposed by Simonyan et al. [40], and the working process is the same as the backpropagation function. This function finally can provide a rank for the importance of the input features[29].

Figure 2.4: Sheep 26%,cow 17%

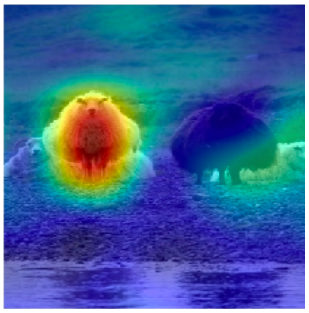


Figure 2.5: Saliency map of sheep

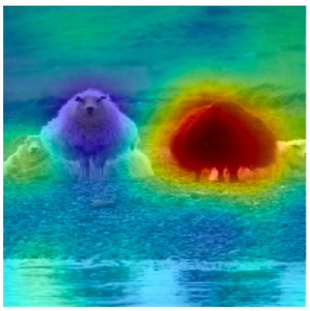


Figure 2.6: Saliency map of a cow

- ◇ **Integrated Gradient**
  Integrated Gradient was first introduced in 2017 from a paper named Axiomatic Attribution for Deep Networks [44]. Sometimes, we do not know that an incorrect attribution is due to an error in the model or the attribution method. But this method can solve this problem. It proves that it can provide reliable results because if an input feature changes the classification score, the attribution value for that input is not equal to 0[44].
- ◇ **SmoothGrad**
  SmoothGrad was proposed and tested by smilkov et al. [41],[7]. The goal of SmoothGrad is to average across these artificially noisy gradients and add noise to make gradient-based explanations less noisy[29]. SmoothGrad is an addition to any gradient-based explanation approach, rather than a stand-alone explanation method itself[29].

– **Perturbation-based approach**
  Perturbation-based methods seek to gain insight into how the model works by changing its inputs, such as the pixels in a picture, the words in a text, or similar components of other data types, and then tracking changes in the model output. There is one typical method (Shapley Values) can be introduced in the following:

- ◇ **Shapley Values**
  Shapley value is an original method from game theory[29]. It is a method to explain prediction by pretending that each feature value of the instance is a "player" in a game, where the prediction is the "payout"[29].

2.2.3. Interpretable machine learning tools

With the development of the field of interpretable machine learning, there are a number of toolboxes that integrate several interpretable methods. The following paragraph will list some of the widely used toolboxes, most of which are related to the natural language processing field.

- **LIME**

  LIME [34] is a toolbox that provides local model-agostic interpretations. The idea of LIME is to use a simple (for example, linear regression model) to approximate the behavior of the black model[34]. This idea was implemented by tweaking the feature value to see how it affected the output[34]. The advantage of LIME is that it can be used to interpret any model, but sometimes it is not enough to explain a complex model.

- **Allennlp**

  Allennlp [48] is not only a toolbox for interpreting models; it provides an NLP pipeline for users to train, test, and evaluate their models. The interpret part of Allennlp consists of two interpretation methods: • Gradient-based saliency map and counterfactual explanations[48]. Compared with lime, this toolbox can support the explanation of a more complex model: For example, it can illustrate why Bert makes the prediction for the mask. It also provides a more user-friendly interface.

- **Seq2seq-vis**

  Since the purpose of our research is to focus on the TS tasks, seq2seq-vis is a toolbox used for debugging sequence-to-sequence structured models. Seq2seq-vis provides opportunities for users to interact with

the model and explain each stage of the natural language processing task[42]. What's more, it allows users to use what-if science to explore the model[42]. For example, in the specific case of machine translation, users can change one word into another, and then the translation results can be displayed.

- **Bertviz**

  One way to interpret the model is to use the attention scores. Since the state-of-the-art method of TS uses a transformer-based architecture, an attention-based approach, Bertviz is an excellent tool to interpret[47]. This tool provides an easier way to decipher the complex model's attention score and supports two views to interpret the model: the high-level model view and the low-level neuron view[47].

- **Ecco**

  Compared with previously mentioned tools, Ecco is a more powerful toolbox. Ecco consists of a set of tools to analyze and visualize the different model types, allowing users to interact with the internal states of models[3]. There are three benefits of using this tool: The first is that it supports all the models built with Hugging face[3]. Besides, users can use this tool to explore model behavior through different types of feature attribution methods. Last, it contains rich visualization types for users to choose from.

2.3. Human-in-the-Loop Machine Learning

The previous section gives detailed information about our task: interpretable machine learning can serve as our solution. For the TS tasks, interpretable machine learning is not enough to describe the model behavior. So, humans can participate in the process, thus giving a more complete explanation, which also includes the syntactic simplification operations. Here, we are going to introduce two related works which can bring humans into the process of interpreting models.

2.3.1. What do You Mean?

Balan et al. [9] proposed a human-in-the-loop pipeline named SECA to interpret the model in the image classification task. The existing interpretability techniques mostly describe a model's behavior by finding salient visual patches, which the users must manually interpret because it supports the model validation with queries to verify many visual concepts[9]. To enable automatic statistical analysis of model activity, salient image regions detected by local interpretability methods are annotated with semantic concepts and then compiled into a tabular representation of images[9]. Here, the crowdsourcing task is to draw a bounding box of the highlighted area of the saliency map and then provide concepts of these circuited boxes. Doing so can provide a set of knowledge for developers to figure out what models really know. Figure 2.7 displays the whole annotation process:

2.3.2. What Should You Know?

Compared to the previous approach, which aims to explain what the model has learned, this paper adds one more task based on it[37]. It also adopts a human-in-the-loop method to analyze what the model should know, then combines the two tasks to characterize the model *unknown unknowns*. Unknown unknowns means that the model feels confident about its prediction while the prediction is wrong[37]. This framework is named Scalpel-HS and engages humans in two tasks: what a model should know and what it really knows. The task's objective is to identify the scene's elements that impact the machine learning model prediction and determine whether this corresponds to the human mental model[37]. Crowd workers categorize the items and connections found by the model and rank the importance of each item in the identification scenario[37]. Figure 2.8 shows the interface of the crowd task, which is described in detail as follows:

a. Draw bounding boxes according to the saliency map.

b. Label the objects and attributes.

c. Classify the relationships among the objects.

d. Concrete all items and relations together.

e. Establish a relevance score for relationships among object pairs.

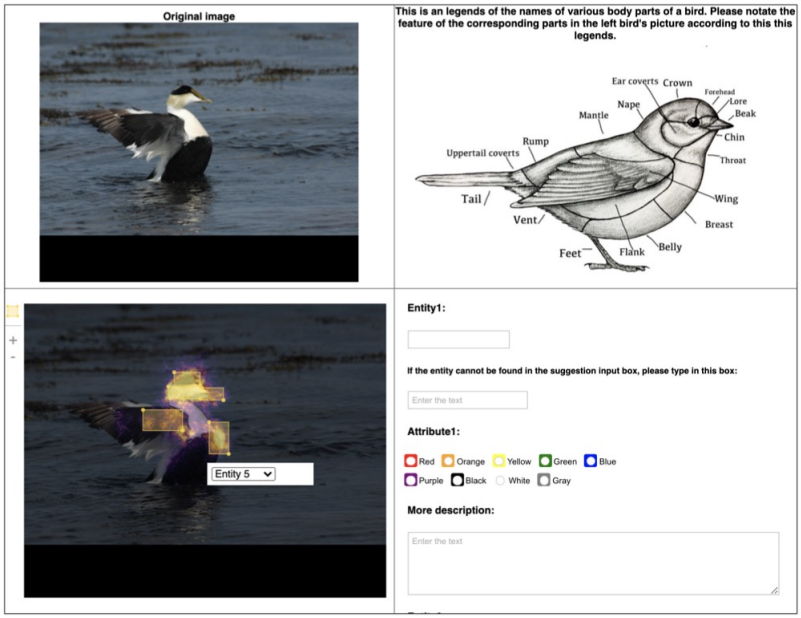


Figure 2.7: The crowdsourcing task of SECA

For the Should-know task, crowd workers must identify the objects and their relationships in the scene to predict the classification results[37]. Figure 2.9 shows the interface of the crowd task which is described in the following paragraph[37]:

a. Determine whether or not the relationship between two objects in the scene graph is right.

b. Given a relevance score for relationships among object paris.

c. Adding missing concepts.

d. Find the minimum objects that are needed to give the result of the classification result.

e. Find the minimum relation sets that are needed to give the result of the classification result.

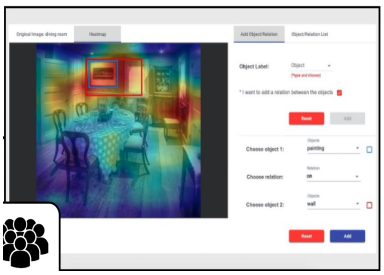


Figure 2.8: Really-Know task[37]

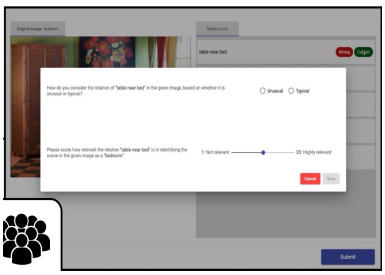


Figure 2.9: Should-Know task[37]

3

Framework

This chapter is going to talk about how our method works to extract the model unknowns. It is a human-in-the-loop method that can be divided into three parts: the first part is to extract what the model really knows, the second part is what the model should know, and finally, we extract the model's unknown by subtracting these two parts. This chapter displays an overview of how to characterize the model unknowns. Figure 3.1 shows the framework. In the following, we will describe each component in detail.

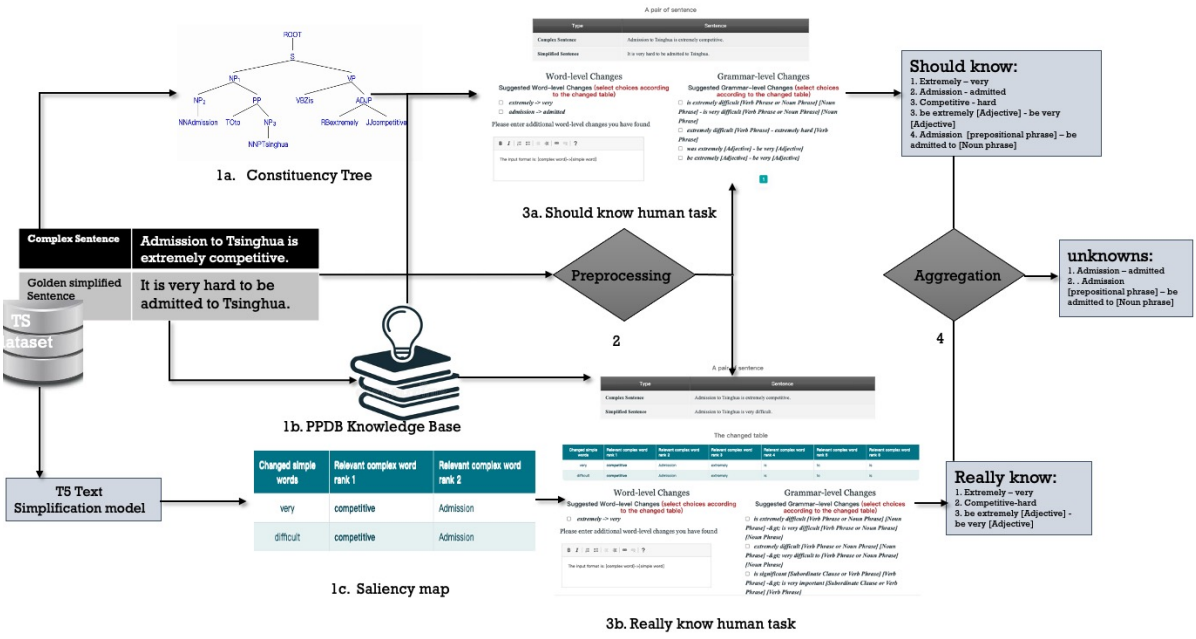


Figure 3.1: **Extracting the unknown unknowns framework**, which is taking a pair of complex and simple sentences with a fine-tuned text simplification model as input. Then it will produce a set of unknowns containing both lexical and syntactic ones as outputs. To do so, the (1a) constituency tree (1b) saliency map and a set of lexical and syntactic rules are extracted from the PPDB knowledge base(([32]). Only the sentence pairs with more than one simplification operation are selected(2.prepossessing); The preprocessed sentence, saliency map, and constituency tree are then fed into (3a) what should be known and (3b) what is really known. These two tasks are human computation tasks where crowded workers can participate in providing knowledge . The final step is to aggregate((4) the output of two tasks for unknown unknowns characterization.

1a. **Consituency tree extract**

Understanding how a complex sentence becomes a simple sentence requires an understanding of the syntactic information of the sentence. For example, a pair of sentences may turn a clause into two sentences. Among the tasks that are Really-Know and Should-Know, we should help crowd workers

figure out how the syntactic changes occur. The Constituency tree can display the syntactic information in the sentence. Thus we generate a Constituency tree for each sentence.

1b. **PPDB knowledge base**

Understanding how the sentences are simplified, we need external knowledge to describe these operations. Except to introduce human intelligence, a paraphrase database named PPDB([32]) can also help. It is a database consisting of paraphrase rules in both lexical and syntactic parts. Therefore, it can be used to help humans to find out simplified operations.

1c. **Saliency map extract**

Understanding model behavior is a machine learning interpretable problem. The widely used method is the saliency map. A saliency map is a post-hoc local interpretability method that can specify which words contribute most when generating new sentences.

2. **Text prepocessing**

For the validation set, this step is going to select informative and meaningful pairs. Only pairs have more than 10% modified words, and they have lexical changes, and syntactic changes can be chosen. For the test set, we will find failure cases selected by three metrics: Fluency, Simplicity, Adequacy, and human annotations.

3a. **The Should-Know task**

The goal of this task is to identify what the model should know. To be more specific, human annotators helped specify what kind of lexical and syntactic operations are needed to get a simplified golden sentence.

3b. **The Really-Know task**

In the Really-Know task, annotators will specify how the model simplifies the sentence by looking at the saliency map.

4. **Aggregation**

First, we rejected some meaningless annotations through three standards(finish time, multiple lexical rules, and more than one syntactic rules). Then, according to our experiment setup, we collect three annotations for each pair of sentences. Here, we will adopt the most vote algorithm for the three annotations, and for each manually input annotation, we add them into the result after a detailed review. After that, we computed the intersection set of the Should-Know and Really-Know sets. In the end, we subtract the intersection set from the Really-know set, and the Should-Know set separately to get a bunch of unknown rules.

4

What model really know

This chapter will describe the pipeline to characterize what the model really knows. First, we introduce how to select and test the local interpretability methods. After that, it will illustrate how human intelligence and knowledge from the knowledge base help interpret models.

4.1. Post-hoc interpretability

4.1.1. Attention-based

Even though the attention-based scores are associated with the model, in machine translation tasks, it is the most widely used method to see how the output is contributed to the input tokens in the task. Text simplification is often treated as the mono linguistic machine translation, So here we would like to see the effect of attention score as our local interpretability method.

In the implementation, we use the cross-attention score of the T5. After an apparent reformation and squeezing, the attention score of all layers and heads is equally weighted. Finally, we use the attention score to select all the relevant complex words for each simple word. The precise algorithm is written as follows: ruled

Implement saliency methods 1 algorithm Attention score calculation

function ATTENTION SCORE CALCULATION($cross\ attention\ score\ torch(X)$)
    $layer \leftarrow 0$
    $head \leftarrow 0$
    $x = x.transpose(1,0)$
    $x = x.transpose(1,3)$
    $x = x.squeeze(x,4)$
    $x = x.transpose(2,1)$
    $x = x[layer,0,head]$
    for $i = 0 \rightarrow xNumber$ **do**
        $idx = np.argsort(att[i].cpu().numpy())[::-1][:6]$
        $score = np.sort(att[i].cpu().numpy())[::-1][:6]$
        $store[idx] = score$

    return $Store$

4.1.2. Gradient-based Pixel Attribution method

Here we also tested the effect of the saliency map. After trying all the toolboxes mentioned in Chapter 2, finally, we use Ecco [3] to generate a saliency map since it provides multiple ways to produce saliency maps. Given the characteristics of T5, we chose gradient-based ways to interpret it. And there are three typical gradient-based ways:

1. **Gradient**

As mentioned in the background, it is a method that relies on a backward pass. To be more specific, assuming that the classification results of a neural network are linearly dependent on each pixel or feature in the input image, the formula can be written as $y = xW + b$. The gradient of output y concerning input x can be measured by $w = \frac{\partial y}{\partial x}$, so it can be used to measure the importance of each input token to the final output.

2. **Intergrated Gradient**

   Ecco [3] also implemented the integrated gradient approach proposed by Denil et al. [13]. As mentioned in the background part, the Integrated gradient method is a method proposed for the gradient vanishing problem caused by the first traditional method. Its formula can be written as follows:

$$IntergratedGradients = (x_i - x_i^{('})) \times \int_{\alpha=0}^{1} \frac{\partial F(x_i^{('})+\alpha X(x_i - x_i^{('})}{\partial x_i^{('})} d\alpha$$

3. **Input X Gradient**

   This method, proposed here in 2020 [16], is also a gradient-based method but differs in that it performs pretty well in the transformer-architecture model, especially on the text classification tasks.

   This gradient can be calculated by this formula:

$$inputXGradient = || \bigtriangledown x_i f_c(X_(1:n) X_i ||^2$$

   Here x_i is the embedding vector of the input token x at the time, and the back-propagated gradient of the selected token's score is $\bigtriangledown x_i f_c(X_(1:n)$. The L2 norm is then used to aggregate the resultant vector into a score. Atanasova et al. [16]' does show the effectiveness of this method.

Here, We measure the effectiveness of the four interpretability methods based on intuition. Since each interpretability method can generate a relationship that displays which complex words are more relevant to the simple terms, we will select ten pairs of sentences from my validation data. Then for each changed simple word, we will choose one origin complex word more relevant to the simple one by our intuitions. In other words, we create the ground truth on our own, and then we use four methods to generate a complex word - a simple word relationship of the ten pairs of sentences. After that, we calculate to which degree the interpretability method covers the ground truth and then choose the interpretability method that can cover most ground truth. The results can be displayed in the table 4.1. According to the table, there are 72 word

Evaluation of four interpretability methods			
Method	**Total word pairs**	**Covered word pairs**	**Coverage**
Attention-based one	72	54	75%
Gradient	72	58	80%
Integrated gradient	72	59	82%
Input x gradient	72	63	88%

Table 4.1: Evaluation of four interpretability methods

pairs in total, and the input x saliency map gets the highest correct rate, so here this method is used as our local interpretability method.

4.2. Human-in-the-loop method

As Described in the background information, humans can be involved in the loops to provide more complete and understandable explanations. For example, syntactic simplifications made by the model can be explained by humans. But if humans were to explain all the tasks manually, it would be a huge burden. Moreover, everyone has their ideas, and the explanation format is not the same, so it is difficult for us to extract the explanations.

We proposed a rule-based method to help humans explain how models simplify sentences. Because the traditional text simplification is a rule-based approach, it simplifies corrections by hand-crafted rules. So,

we aimed to explain model behaviors by rules. Now our sentences have undergone multiple changes; the previous existing handcrafted rules were not enough. After a careful search, we found a simple paraphrase knowledge base named PPDB knowledge base([32]). It contains over 100 million paraphrases. Besides, the paraphrase type can be divided into three types: lexical, syntactic, and phrasal, and examples of these different types of rules can be described in table 4.2. Moreover, this knowledge base is available in 21 different languages, and here we primarily focus on English. This knowledge base is constructed by extracting pairs from a bilingual parallel corpus of nearly 10 million sentences[17]. In addition, each rule in the knowledge base has a score that indicates the confidence score for the complex words or structures that are paraphrased into a single structure. Here, we decide to use the lexical and syntactic parts in the PPDB knowledge base([32]) to describe our model behavior because traditional rule-based text simplification also selects rules from the lexical and syntactic parts.

Examples of three different types of paraphrase rules	
Lexical	required - must
Syntactic	is required to [Verb phrase] [Noun phrase] - must [Verb phrase] [Noun phrase]
Phrasal	is required to - must

Table 4.2: Examples of three different types of paraphrase rules

4.2.1. Annotation phase

To better apply the crowdsourcing method to help describe model behavior, we developed a web application to involve humans. This paragraph describes the most important part: the structure of the annotation phase. Figure 4.1 shows the annotation interface. Part 1 is the sentences table(complex and simple tables), and part 2 is the table that shows the relationship from the saliency map produced by Ecco[3]. In addition, part 3 is the first task for humans to select lexical changes for the application. Section 3a displays the suggested word level changes(automatically retrieved), and part 3b shows an input box for users to enter the additional lexical changes that the users deem reasonable but do not appear in the original sentence. What's more, part 4 focuses on the syntactic level. The next part is to illustrate the implementation details of each part.

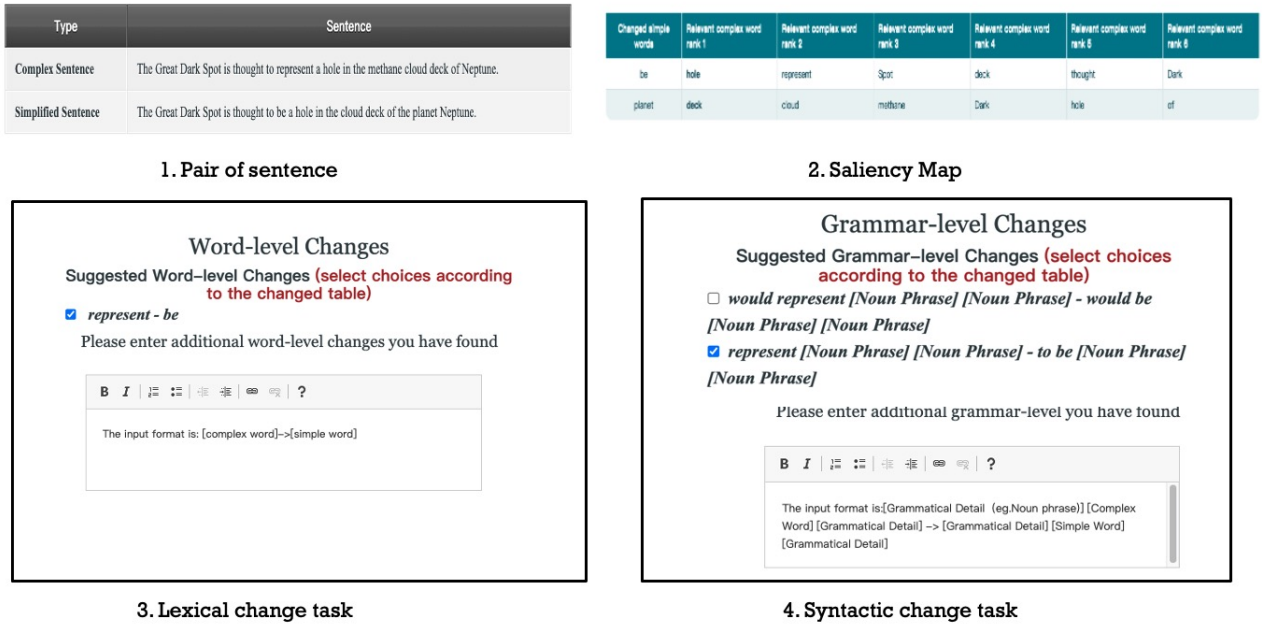


Figure 4.1: The overview of a user interface for Really-Know task

1. **Pair of sentence** The first part is a table which displays the original complex sentence and simple sentence.

2. **Saliency map**

   The second part is the saliency map which shows simplified words and their relevant original words. We have implemented three ways to display this information. All the three methods use the same information generated by Ecco, and only differs in their presentation ways.

   1 . It is an interactive picture. When the user clicks one simple word, its corresponding complex words can be shown. And the darker color is a complex word; it contributes more to generating a simple word.

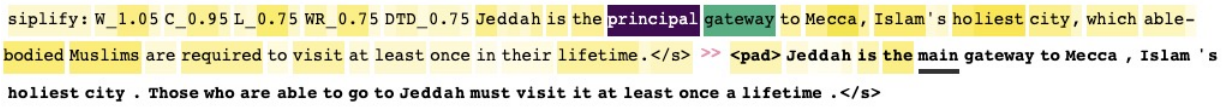


Figure 4.2: It is an interactive visualization map, if we click the **main** in the simple sentence, then the most relevant word **principal** in the complex sentence can be labelled with a dark colour

   2. The second method is done by HeatmapVisual(a python library). The difference is that when users click on the words in the simple sentence, all relevant words in the complex sentence will appear in the head of the simple words and ranked one by one. From my point of view, this way of displaying information is much easier to understand. [3.] We also proposed a more simple way

Figure 4.3: It is also an interactive visualization tool but only plays the simple sentence and the related complex word. For example, if we click the word **main**, then six complex related words like **principal** appear at the head of the simple sentence.

   to display information. It is formatted like a table, with the first columns being words that appear in simple sentences but not in complex sentences. What's more, each of those simple words is paired with the words from the complex sentence that significantly impact simplification.

Changed simple words	Relevant complex word rank 1	Relevant complex word rank 2	Relevant complex word rank 3	Relevant complex word rank 4	Relevant complex word rank 5	Relevant complex word rank 6
must	**required**	ablebodied	visit	once	least	lifetime
Those	**ablebodied**	Muslims	which	required	visit	lifetime
who	**ablebodied**	Muslims	which	Islams	required	visit
it	**visit**	Mecca	gateway	once	ablebodied	lifetime
main	**principal**	gateway	Mecca	holiest	Islams	ablebodied
go	**ablebodied**	Muslims	visit	which	gateway	Mecca

Figure 4.4: The changing table displays simple words and the corresponding six complex words.

   The point of crowdsourcing is that we hire people from different backgrounds. As a result, some do not have a computer science background. After displaying three visualization methods to my colleagues, they all agreed that the third way is easier to be understood. Therefore, we adopted the third way to illustrate the saliency map.

3. **lexical changes**

   Simplification can be divided into the lexical part and the syntactic part. This paragraph is mainly focused on lexical changes. It is quite difficult for humans to describe the simplified operations by themselves. Here, we automatically select the appropriate lexical rules from the PPDB knowledge base([32]),

and crowd workers can choose the reasonable rules aligned with their idea. The automatic process of retrieving rules can be described as follows:

- Find all the words that appeared in the simple text but not the complex text. The Tokenizer of T5 tends to detokenize words into tokes which is not easy to manipulate, so here we recover all the tokens to their original word forms.
- According to the saliency map, we have got a relationship list of which complex words can contribute most to the simple word, and here we choose six relevant complex words for each word we found in the first set according to the relationship list.
- Then, all the combinations are searched in the PPDB knowledge base([32]). Here, the exact match has been executed(rules can only be added to the output if both complex and simple words are equal to the complex-simple pairs in the knowledge base.

In addition, the knowledge base can not include all the lexical changes, so there may exist some other lexical rules that do not appear in the knowledge base but appear in the sentences. Therefore, in part 3b, we provided an input box for crowd workers to enter additional rules they have found.

4. **syntactic changes**

   Compared with lexical changes, syntactic changes are a more complex task. Rules can also be used to describe syntactic changes, but crowd workers have a hard time understanding the meaning of the rules without a linguistic background.

   Here, we aim to use more comprehensive and easily understood rules to explain the syntactic changes that the model has made. As mentioned in the previous paragraph, there is also a syntactic version of the PPDB knowledge base([32]), which we can apply here. The format of syntactic rules in PPDB knowledge base([32]) is as follows:

$$[POS\ taggers]\ Complex\ phrases\ [POS\ taggers] -> \\ [POS\ taggers]\ Complex\ phrases\ [POS\ taggers]$$

   Part of speeding taggers(POS taggers) mark words in a text (corpus) corresponding to a particular part of speech based on its definition and context. For example, an *apple* can be marked as a *noun* in a sentence: *My favourite fruit is apple.* These pos taggers help us specify the syntactic structures in the sentences, and we can figure variations in words and structures with these pos taggers. For defining these syntactic structures, we also applied a consistency phrase tree which can split the sentences into constituents. The explicit steps on how to retrieve syntactic rules are as follows:

   1. Gain a set where all the words appear in the simple text but not in the complex text.
   2. According to the saliency map, we have got a relationship list of which complex words can contribute most to the simple words, and here we choose six relevant complex words for each word we found in the set according to the relationship list.
   3. Using the generated complex-simple pair to find the rules in the database, but here, due to the reason that a phrase may contain multiple words, we can add the rule to the output list if the complex structure contains the complex words, and if the simple structure contains the simple words.
   4. 4. Then, we use the constituency tree of the pair of sentences to select all the rules with the same syntactic structure as the pair of the sentence.
   5. Since we want to reduce the cognitive load of crowd workers, we do not wish to display all these rules. Therefore, we used the paraphrase score to rank all the rules in the previous steps and showed the top five rules to the crowd workers.

4.2.2. Training phase

After presenting our crowd task to supervisors, they thought it difficult for crowd workers with no computer science or linguistic background to understand. After an initial pilot study, We found that the poor performance of our application could be due to the lack of linguistic knowledge and not having proficient task processing abilities(they do not know how it works). Thus, in this phase, we adopt an explicit training method:

workers are asked to finish training tasks. Once they have completed these tasks and labeled each training task perfectly, they can process the annotation task.

We have added a training phase which consists of two tasks, each of which takes about 3 minutes, as a longer training period may discourage crowd workers from joining. The crowd workers have to finish all the training tasks with 90% accuracy before they can start the main tasks, and during the training process, of course, they can get hints if their answers are wrong. When they submit the training task, the correct choice will be labeled in green, and the wrong choice will be marked in red. After that, the crowd workers can change their answers. If they are confused about what to do, there is a hints button explaining why the answers are correct.

90% was set as the passing score because there are no absolutely correct answers for the linguistic changes, and it's unrealistic for the crowd workers to choose the exact choice as the ground truth. So except for the training part, We have also added a syntactic information table in the instruction part to help crowd workers to finish our tasks better. As mentioned above, syntactic rules may contain information about the syntactic structure, which is hard for crowd workers without a linguistic background to understand. To solve this problem, the syntactic information table gives information about the syntactic structure and provides a specific example of how to apply syntactic structure. Table 4.3shows the format of the syntactic structure.

Example of syntactic information table		
Syntactic structure	**Explanation**	**Example**
Adverb Phrase	Phrasal category headed by an adverb (including comparative and superlative adverbs)	rather timidly, very well indeed

Table 4.3: Example of syntactic information table

4.2.3. Pilot study

This section will examine the performance of our application, and we will focus on two parts: the first is what the results of our application's collection are, and the second is the efficiency of our application. In this pilot study, we focused on investigating two phases: the training and annotation phases.

a **Training phase**

In this phase, we will study three aspects: how long it took them to finish the training task, how many attempts they tried to complete it, and whether they think the training is helpful for them in understanding the annotation task. To know how the crowd worker understands the task, we collected data from five users with no computer science or linguistic background. None of the Five users all are not speaking English, and they came from different majors like microelectronics, civil engineering, and embedded systems. Table 4.4 shows the finish time for each task, the attempts they made to finish the training task, and some other information.

Pilot study(Training phase)						
	Crowd worker1	Crowd worker2	Crowd worker3	Crowd worker4	Crowd worker5	Average
Training task 1 finish time	3min	2min14s	5min28s	3min27s	2min13s	3min16s
Training task 2 finish time	4min28s	3min13s	6min11s	4min28s	3min 27s	3min45w
All training cost time	7min28s	5min27s	11min39s	7min55s	5min40s	7min1s
Attempts to finish training task1	1	1	2	2	1	1.4
Attempts to finish training task2	3	3	4	5	2	3.4
Helpful or not	Yes	Yes	Yes	Yes	Yes	Yes
Need hints of not	No	No	No	No	No	No

Table 4.4: Pilot study of Training phase

Table 4.5 shows the results of our training pilot study. From my point of view, training task 1 is easier than training task 2 since there are only two changes from the original sentence to the simple sentence. And the results indicate that crowd workers spent an average of 3 minutes on a simple training task and 4 minutes on a more challenging training task. So for each task, they try three times on average to get 90% accuracy. Thus, we can find that our job makes it difficult for them to understand what to do,

and sometimes it takes time for second language readers to describe specific changes. However, none of them use the hints to see the correct answer, which means the task can still be completed even if it takes time. In addition, our study shows that the training phase is essential for them to understand how the process works.

b **annotation phase**

There were ten tasks for the annotators to finish, and we studied how long it took for them to complete all of them, whether they understood our task or not, and whether their answers were of quality. For the third question, one of my friends who studied English education and I annotated all ten tasks, and we made a golden answer. Then, we compare the user's response with the golden answer(if their answer covered all our golden answers, the score is one. Also, the score can be higher than one because sometimes crowd workers can find additional information). The calculation formula can be written as follows:

$$Accuracy = \frac{Labelled\ accurate\ rules}{Golden\ accurate\ rules} \tag{4.1}$$

This formula can be used to evaluate both lexical parts and syntactic parts.And table 4.5 showed the experiment results: This table indicates the average task completion time is 2min20s, and based on

Pilot study(Annotation phase)						
	Crowd worker1	Crowd worker2	Crowd worker3	Crowd worker4	Crowd worker5	Average
All task finish time	28min	17min26s	23min	29min15s	20min24s	25min
Average task finish time	2min54s	2min8s	2min18s	3min9s	2min2s	2min30s
Understand the task or not (0-10(understand very well))	8	9	8	7	9	8.2
Lexical accuracy	82%	88%	76%	82%	84%	82%
Syntactic accuracy	75%	80%	68%	79%	77%	75.6%

Table 4.5: Results of the annotation phase

my findings, each task's completion time decreased with the crowd workers' completed task. Besides, there were five tasks in the pilot study, which only consisted of several simplification operations, so the average finish time for the accurate tasks may be longer. And people all understand the task well because they can recite the purpose of our application. In addition, we also found that the lexical accuracy was 6.4 percent higher than the syntactic part since the lexical task were more straightforward. Last but not least, we did find the accuracy and the time to complete our task is relevant to the English level of crowd workers. One of the crowd workers has an excellent English level and the complete time of each task is 28s shorter than the average completion time. So here, in our real experiment setting, we aimed to search for crowd workers with high English proficiency to do those annotation tasks.

Except for the numerical findings, this pilot study also aimed to collect feedback about our application. When we get users' feedback, we can know the real needs of the users. Therefore, we can make it easier for them to use our application. The Suggestions can be divided into two parts: one on the format of the user interface and the other on the lack of clarity of information.

4.2.4. Budget calculation

This section is going to introduce how we calculate the budget. With the growing development of crowdsourcing, there are several crowd platforms now like Amazon Mechanical Turk1 (mTurk), politics, and CrowdFlower[22]. The workflow of these platforms is like this: Programmers designed and posted their tasks, and then these tasks are visible to the crowd workers, who can choose which task they want to complete. And the average hourly wage depends on where the crowd workers are located [22]. For example, in the United States, An ethical minimum hourly wage for crowd-sourcing work is $8.50/hour, which is the national average minimum wage based on the population distribution. For our research, we consider hiring crowd workers from European countries, and the average salary is 9.31 euros per task.

Here, we also want a high quality of the final results, so multiple labels are needed because sometimes crowd workers tend to make mistakes. So we adopt a majority vote algorithm to ensure a more accurate result: each task requires three annotations, and then we choose to mark the answers which are more than or equal to twice.

For Really-Know task, there are 809 complex-simple sentence pairs in our task. As mentioned in the previous part, each sentence needs three annotations, and in our pilot study, each task needs 3mins to complete. Besides, we are willing to pay 9.31 euros for one hour of work for a crowd worker. So our budget should be:

$$Budget = \frac{Number\,of\,task * Number\,of\,annotations * Minutes\,per\,Task}{hour} * Wage \tag{4.2}$$

$$1129.77 = \frac{809 * 3 * 3}{60} * 9.31 \tag{4.3}$$

5

What model should know

This chapter will cover the application to collect what our T5 model should know. It will first introduce the user interface of what the model should know. Since some application parts were discussed in Chapter 4, here we will only focus on the different parts compared with the Really-Know application.

5.0.1. Annotation phase

This paragraph discusses how the annotation phase works. It can be structured as follows: 1a is the introduction which gives the overall image of the task. 2a is the pair of sentences (a complex sentence and a golden answer from the Asset data set). 3 is the lexical task. 3a provides automated retrieval rules from PPDB lexical database ([32]), and 3b is an input box for crowd workers to submit rules which describe the model behavior, but this does not appear in 3a. In addition, part 4 is the syntactic task. 4a displays the automated generate pairs from the PPDB syntactic database([32]). 4b is also an input box for the additional rules. The interface is almost the same as the Really-Know application, but there are some key differences listed as followings:

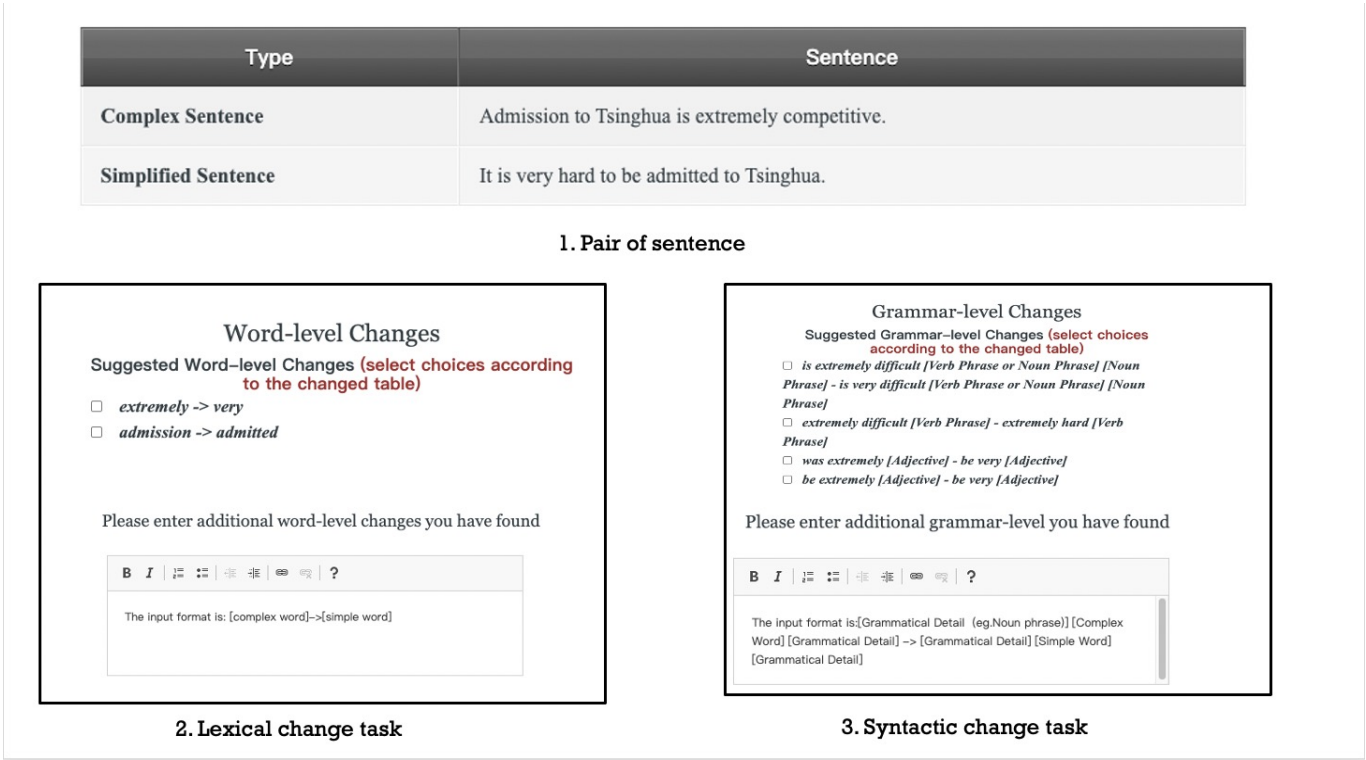


Figure 5.1: The overview of the annotation task for what model should know

1. **Pair of Sentence**

   The origin and golden sentences are all from the Asset dataset[6]. The golden answer sentences are generated by a reasonable process and the details of which will be talked in the following paragraph:

   - These golden simplified sentences are generated by crowd workers. First, these crowd workers are hired with strict requirements: they should be native English speakers and finish more than 1000 human intelligence tasks; in other words: they should be professional crowd workers[6]. In addition, their task acceptance rate is more than 80%[6]. These strict requirements do ensure the quality of the results[6].
   - Then, they are provided with a set of instructions about how to simplify the sentence[6]. Examples of phase splitting, compression, and lexical paraphrasing (lexical simplification and reordering) (deleting unimportant information) are given[6].
   - There is a training task that needs every crowd worker to pass[6].
   - After that, they can move to the annotation task; they are asked to give their simplification of the original sentence[6]. Besides, they have to enter their confidence score about their simplification using a 5-point scale[6].
   - Their answers are justified by linguistic experts[6].

   The strict work process proved the quality of the golden sentences. Therefore this dataset is adopted as data for our Should-Know task.

2. **Lexical task**

   For the lexical task, we also adopt automatically retrieved rules from the PPDB knowledge base(([32]) to describe simplified operations. But without a saliency map, the process of retrieving the lexical rules is changed, and the new method is described in the following paragraphs.

   - We first find out the generated simple words, meaning that these words do not exist in the complex sentence but in the simple one. Here, we describe this as words set A.
   - Then, we subtract complex sentences with words in the simple sentence, and the remaining words are called word set B. Words set B contains the words most likely to be replaced by words set A.
   - We generate word pairs by fully matching the word set A and word set B. The formula can be written as: $\forall a \in A, \forall b \in B, -> pairs\ (b-a)$
   - The rules search process is the same as the Really-Know application, and it is going to retrieve exact match rules.

3. **Syntactic task**

   The difference for the syntactic tasks is how we generate the word pairs, which we have already illustrated in the last part.

5.0.2. Pilot study

This pilot study is going to explore how my application works. Then, when we get feedback, we can iterate and reiterate our application to improve the user experience and thus the quality of results. Our pilot study will be divided into two parts:

a **training phase**

We found five crowd workers who had not seen the model Really-know application before and that they had no computer science background or linguistic background. They are all highly proficient in English and study at Tu Delft. Here, we aim to check whether our training task is easy for them to understand and complete. Besides, we explore whether they can benefit from our training tasks. Also, to estimate how long it took them to finish the training task. According to the result displayed below, it is easy to notice that the training time is longer than required for the Really-know task. It shows that without the saliency map, it is more difficult for crowd workers to find the simplification operations. Besides, crowd workers have tried more attempts to reach 90 % accuracy. What's more, almost all of the workers finished the tasks without hints, and they did agree that training is helpful for them to complete the annotations.

Pilot study(Training phase)						
	Crowd worker1	Crowd worker2	Crowd worker3	Crowd worker4	Crowd worker5	Average
Training task 1 finish time	4min28s	4min21s	4min1s	3min15s	2min19s	3min45s
Training task 2 finish time	5min33s	3min27s	4min42s	3mins9s	3min24s	4min3s
All training cost time	10min1s	7min48s	8min43s	6min24s	5min43s	7min48s
Attempts to finish training task 1	2	1	2	3	1	1.8
Attempts to finish training task 2	3	4	2	2	1	2.4
Helpful or not	Yes	Yes	Yes	Yes	Yes	Yes
Need hints of not	Yes	No	No	No	Yes	NA

Table 5.1: Pilot study of the training phase

b **Anotation phase**

Regarding the annotation phase, we randomly select ten tasks. Each of the tasks has one more simplification operation. In addition to studying how long a task takes, we also want to check how much they understand our mission. In addition, we aimed to find the accuracy of their annotations (the golden answer pairs were also made by a friend who studies English Education).

Pilot study(Annotation phase)						
	Crowd worker1	Crowd worker2	Crowd worker3	Crowd worker4	Crowd worker5	Average
All task finish time	30min14s	22min15s	27min	18min15s	25min24s	24min22s
Average task finish time	3min1s	2min27s	2min42s	2mins9s	2min54s	2mins44s
Understand the task or not (0-10(understand very well))	9	10	8	9	9	9.2
Lexical accuracy	85%	88%	70%	80%	84%	81.4%
Syntactic accuracy	72%	76%	61%	77%	77%	72.6%

Table 5.2: Pilot study of the annotation phase

Table 5.2 shows the average finish time per task, which costs 2min44s per task, so it is longer than the Really-Know annotation tasks. However, without the saliency map, our Should-Know application is easier for them to understand than the Really-know application. Refers to the accuracy, both lexical accuracy and syntactic accuracy are calculated by the following formula:

$$Accuracy = \frac{Labelled\ accurate\ rules}{Golden\ accurate\ rules} \tag{5.1}$$

If the crowd workers label more reasonably simplified operations, the accuracy can be higher than one. As you can see From the table, 5.2, the accuracy of the two tasks that should be known is lower than the accuracy of the actually known tasks. Therefore, without the help of a saliency map, it is more challenging to figure out the relationship between a generated simple word and a complex word.

5.0.3. Budget calculation

This paragraph will calculate the budget we need to collect the suitable data. As mentioned in the previous section, we would like to collect three annotations for one task, and 9.5 euros is the average wage for a crowd worker for an hour. From the pilot study, on average, it costs 2min44s for one crowd worker to finish one task. And We considered the task's difficulty, so we assumed it would take three minutes to complete one task. Then, the final cost can be calculated by the following formula:

$$Budget = \frac{Number\ of\ task * Number\ of\ annotations * Minutes\ per\ Task}{hour} * Wage \tag{5.2}$$

$$1129.77 = \frac{809 * 3 * 3}{60} * 9.31 \tag{5.3}$$

6

Implementation Details

This chapter introduces the implementation details of the Really-know application and Should-Know application. It first introduces the technology stack and libraries we use to implement the web application and then talks about the web application's user interface. After that, the storage of data is also talked about.

6.1. Technology Stack

To better apply the crowdsourcing method to help describe model behavior, we developed a web application to involve humans. Python-Flask is adopted as our main framework for a more lightweight development environment and a more compatible setting with our model output. For accessibility, I use SURF Research Cloud to deploy my application so that every crowd worker can access my application easily. In addition, Docker is used for composing my application, so it is easy to version iterations.

6.2. User interface of Web application

This section talks about the user interface of my web application. Since Really-Know and Should-know applications are almost identical, we illustrate the interface together. Figure 6.1 displays how the application works, with the following details for each page.

- **Introduction page** This is the home page of our application, which provides explicit information about the task. Besides, there is a straightforward tutorial to teach crowd workers how to finish the annotation task step by step. In addition, due to the specification of our task, linguistic knowledge is required to complete the annotation task. On this page, there is a table showing the explanations of each specific term.
- **Register page** This page is designed for each crowd worker to register, and each crowd worker should have a unique username.
- **Login page** After the crowd workers are successfully registered, they can move to the login page. Only by logging into the system can they move to the training task.
- **training page** Training page is designed to help crowd workers understand how our application works.
- **Annotation page** When the user completes the training task, they can start performing the annotation task. The implementation details of the annotation task will be discussed in the next paragraph.

6.3. Data storage

In this part, we will illustrate how we store these data and how to process them. As discussed in the last part, our annotation task contains two sub-tasks: the lexical change task and the syntactic change task. We provided a set of automated retrieved rules and input boxes for each task for crowd workers to enter additional rules. Therefore, the data we collected is a rule-based format(*Complex word - simple word* or *[pos taggers][complex phrase][pos taggers]-[pos taggers][simple phrase][pos taggers]*. These data are stored in the database whenever a task is submitted.

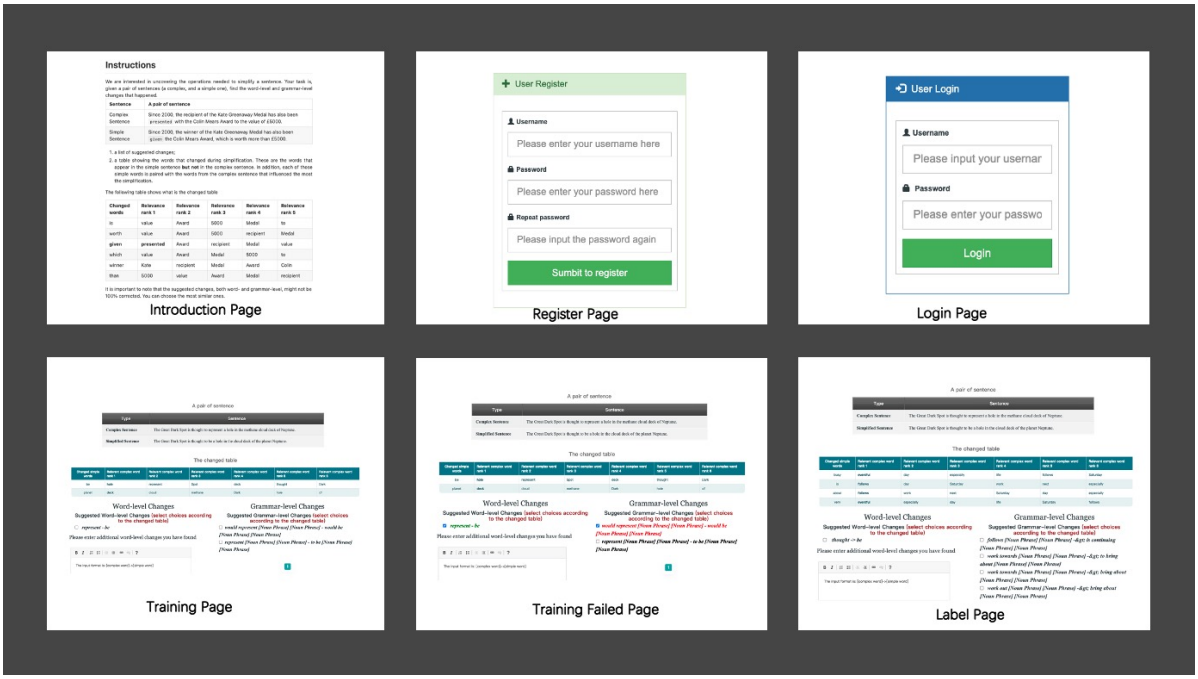


Figure 6.1: **Work process of web application** Users first log into the introduction page to get the essential information about how this application works, then they can go to the registration page to register an account; after that, they can log in and begin to finish the training task. When they finish their training task, they can start annotating.

For an easier way to get our results, we use SQLite as our central database. Since it is a lightweight database and all its data is stored in a .db file, it is easier for us to process these ppdb format rules submitted by crowd workers.

7

Experiment Set-up and Results

In this chapter, we will revise how different experiments are set up, present the results, then conduct an in-depth analysis of the results. First, let's start with an overview of the experiments. Then, you will be introduced to choosing the state art of text simplification model. After that, the experiments on human intelligence tasks will be presented. The final step is how we characterize the unknown, where the goal is to prove the effectiveness of our framework.

7.1. Overview of the experiment

In order to prove the effectiveness of our framework, we have to show that the unknowns we extracted can indeed predict the failure cases that the model will face in the future. So the overall experiment set-up can be described as follows: First, run the framework on the validation set, and then we get a set of rules that should be known, a set of rules that model really know and then extract a set of unknowns extracted from the Should-Know set and the Really-Know set. After that, we run the framework on the failure cases of the test set and get a bunch of unknowns from the test set. Finally, we override the unknowns from the failed cases in the test set by calculating the unknowns from the validation set. If the unknowns cover the failure cases, our framework proves its effectiveness in extracting the unknowns that predict future failures. In addition, we also calculate how part of our framework covers the unknowns, proving part of our framework is still adequate to predict the failure cases.

7.2. Text simplification

As discussed in the previous sections, the state-of-the-art TS method is not so good because it makes mistakes in several situations. We are trying to figure out why it was flawed. Furthermore, we can understand why our model made such simplifications because The increased understanding of the inside can make the users more convinced with the model. Therefore, we aim to analyze the state-of-the-art text simplification models. So, first of all, we will study which dataset we will use.

7.2.1. Datasets

After searching the keywords**Text simplification dataset** by Google Scholar and Tu Delft library, there are three widely used textual simplification datasets:

- **Newsela**

  Before the Newsela dataset was proposed, Simple Wiki data sets were mainly used for the Text simplification tasks. Due to the fact that Simple Wikipedia was created for children and adults who are new beginners to English[27]. There are several problems in the dataset:1) It's prone to automated sentence alignment errors; 2) A high proportion of insufficient simplifications. 3) It doesn't translate well to other text genres[51]. Therefore, the Newsela dataset[51] was created to solve this problem. Newsela is a new simplification corpus of news articles rewritten by expert editors to fulfill the readability standards for children of various grades[51]. This parallel corpus contains 1,130 news articles, so it is the size of the simple Wikipedia dataset[51].

- **TurkCorpus**[52]

  Data-driven text simplification models are difficult to evaluate on the complex, simply aligned datasets, especially if there is only one complex sentence with one paired simple sentence. To solve this problem, Xu et al. [52] conducted an in-depth application of statistical machine translation, which combined both advantages of large-scale paraphrases learned from bilingual texts with simplification rules from the PPDB knowledge base([32])[52]. They built a dataset named TurkCorpus, which consists of 2,359 original sentences from the large bilingually paired dataset English Wikipedia. Besides, each complex sentence has 8 manual reference simplifications. The dataset is divided into two subsets: 2,000 sentences for validation and 359 sentences for testing simplification models[24]. The advantage of this dataset is that the quality of the golden simplified sentences is quite good.

- **ASSET**

  Asset is a simplification dataset focused on multiple transformations: deletion, paraphrase(including syntactic structure and lexical words), and splitting(decomposing the long sentence into sub sentences), while other datasets only focus on one or two transformations[6]. The method uses crowdsourcing to create ten simplifications referencing each complex sentence from the TurkCorpus dataset and perform multiple transformations for each created simple sentence. So the quality of these simplified sentences is better than the original Wikipedia data set. It consists of 23,590 human simplifications paired with 2,359 authentic TurkCorpus sentences in total[6].

After clearly reading the paper and the dataset itself, Newsela contains multiple transformations, while its sentence alignments are automatically generated and thus not very good. Moreover, it is not easy to obtain its date due to the needs of the signed agreements. As for Turkcorpus, it mainly focuses on the transformations of lexical paraphrasing like reducing the number of a difficult word and does not consider some modifications, such as deleting and reordering[6]. Simplifying phrases prevents evaluating a model's capacity to perform a more diverse collection of rewriting operations[6]. So, here we would like to evaluate models on a dataset asset that targets multiple transformations and is more accessible.

7.2.2. Models

Since our goal is to work with Asset datasets, I am committed to finding the state-of-the-art models that achieve the best performance on this dataset. We search on the websiteand found there are the ten most popular models. After careful analysis, I picked 5 of them for in-depth research:

- **Dress-LS**

  Zhang et al. [53] proposed a model named Dress(Deep Reinforcement sentence simplification), combining the encoder-decoder structure with Deep Reinforcement Learning. It shows the possibility of combining a reward function, encouraging simple and fluency output. This model is trained on a simple wiki dataset, and the experiments showed the effectiveness of deep reinforcement learning. framework[53].

- **DMASS+DCSS**

  This method adopts a transformer-based architecture to build a sequence-to-sequence model. Its novelty lies in the fact that it combines rules from the Simple PPDB knowledge base (A Paraphrase Database for Simplification)[54]. In addition, it adopts two ways to incorporate rules: (1) DMASS: it uses the additional component to identify the context of these simplified rules and outputs [54]. (2). DCSS: it encodes the context and the output of the simplification rules into the model share parameters[54]. The experiments reveal that the integration component has two key advantages: [1]. The evaluation metrics indicate that the combined model outperforms some state-of-the-art methods in Sentence simplification[54]. The model aims to pick more accurate simplification rules by analyzing rules usage so that the simplified sentence has better readability[54].

- **Access**

  Martin et al. [25] aims to solve the text simplification task by a standard Sequence-to-Sequence structure. Besides, a discrete parametrization scheme is designed to satisfy the users' different targets. As a result, users can control the output by restricting the length of the output and the number of paraphrasing operations. They named this method ACCESS(Audience-CEntric Sentence Simplification) and conducted experiments to demonstrate that the method was more effective than the out-of-box sequence

to sequence models. Also, they explicitly analyzed the effects of several controlled attributes and then applied the controlled attributes with the best performance.

- **Muss**

  In 2020, a simplification method named Muss was proposed[26]. This method adopts the control mechanism from Access([25]) to generate simple text. Besides, instead of using the traditional sequence-to-sequence model, it adopts a pertained sequence-to-sequence model with a denoising auto-encoder named Bart([21]).

- **T5**

  T5 (A Unified Text-to-Text Transfer Transformer) is a sizeable pre-trained language model that has recently obtained best-in-class performance in various NLP tasks. Sheang et al. explored the possibility of combining the powerful T5 model and control mechanisms together[38]. This combined method has two advantages: first, T5 is a compelling model, and it is pretrained on several supervised and unsupervised NLP tasks such as machine translation and document summarization. Besides, it employs the best style of token and masking strategies. Compared with ACCESS, this method explores the effectiveness of adding more different tokens, such as word length(controlling the length of the generated word).

As we are going to interpret the best simplification model, we reimplement the above five methods. Then, we ran an experiment on the Asset dataset and used three metrics(SARI, BLEU, FKGl) to evaluate the model performance. The following part describes the details of the three mentioned text simplification evaluation metrics:

- **SARI**

  The system outputs are compared to the references and the original text using SARI[52]. It assesses the effectiveness of text simplification on the lexical level by evaluating the quality of words added, eliminated, and retained. It is currently the most widely used metric, and we use it as the primary evaluation metric[38].

- **BLEU**

  BLEU[31] was designed for Machine Translation and has been widely utilized. Due to its low correlation with human judgments and frequent penalization of smaller phrases, BLEU has lost favor in Text Simplification[43].

- **FKGL**

  FKGL was designed to measure the text's readability level, and a lower score means that it is more accessible to be understood[19]. However, these metrics do not consider grammatically and meaning preservation, which means that sometimes even the sentences lose their original meaning, but still can gain a good FKGL score[38].

Evaluation of existing text simplification methods				
Model	**Dataset**	**Evaluation metrics**		
		SARI	BLEU	FKGL
Dress-LS	WikiLarge	36.59	86.29	7.66
DMASS+DCSS	WikiLarge	38.76	70.44	7.36
ACCESS	WikiLarge	40.12	74.05	7.01
Muss	WikiLarge	43.01	75.21	6.24
T5	WikiLarge	44.62	70.08	5.84

Table 7.1: Evaluation of existing text simplification methods

As you can see from Table 7.2, T5 does perform better than the other four sequence-to-sequence models in terms of the score of SARI(the primary evaluation metric). Therefore, we decide to interpret the T5 model.

7.3. Data Preprocessing

As mentioned before, Asset is adopted as our main dataset. For the validation set, there are sentence pairs that do not have multiple simplified operations. And there are also pairs of sentences that do not have automatically retrieved rules. So here, even though there are transformations between sentences, it is difficult for crowd workers to describe the changes and post-process the collected rules. Therefore, We need to find a pair of sentences with more than one simplified operation to gather more quality and valuable results. The following algorithm chooses these sentences:

Select informative sentence 2 algorithm Select informative sentence

```
1: function SELECT INFORMATIVE SENTENCE(Modifiedwords, LexicalRules, Syntacticrules, Sentence)
2:     Sentence ← False
3:     if Modifiedwords > 0 then
4:         if lexicalRules > 0 then
5:             if Syntacticrules > 0 then
6:                 Sentence = True
7:     return Sentence
```

According to the algorithm, it is easy to find our goal is to select pairs with one more lexical operation and one more syntactic operation. Therefore, the size of the validation set changed from 2000 to 649 pairs.

For the test set, we aimed to select the failure cases. The failed cases are chosen by the two criteria:

* **Text Simplification evaluation metrics**

  There are three frequently used metrics in the area of text simplification: SARI[52], BLEU[31], and FKGL[19]. We use Easse[5](a text simplification evaluation library that consists of various evaluation metrics) to compute these three scores for each pair of sentences. Then, we calculate the three average scores for all sentences, and if two of the three sentences are lower than the average score, the three sentences are considered to have failed. In this step, 127 pairs of sentences are chosen as failure cases.

* **human evaluation**

  As mentioned in previous parts, these three automatic evaluation metrics are not absolutely correct. For example, BLEU is mainly used in the area of machine translation, so it is concerned with how to obtain the meaning of a sentence while ignoring whether the changed structure or words are simpler. FKGL is a measure of whether this sentence is easy to read or not, but it tends to give a high score to a short sentence while ignoring its meaning. So, even if some of the pairs of sentences are deemed correct by automatic evaluation metrics, human intelligence is adopted to evaluate these sentences.

  Following the previous work([14],[38]), we developed a crowd-sourcing task to collect human evaluations of these sentences. And we required our colleagues who participated in this task with a high level of English proficiency and to read a clear task guide about the task. They will then score the remaining pair of sentences using a 5-point Likert scale in three aspects[38]:

  – **Fluency** This indicates whether the sentence is fluent or not and whether it is easy to understand its meaning. Besides, it also investigates whether it has any grammar mistakes.

  – **Simplificity** This metric measures whether the simplified sentence is simpler than the original sentence.

  – **Adequacy** It shows the extent to which the simplified sentence preserves the meaning of the original sentence.

  Figure 7.1 shows the interface of the designed interface and averages the scores for each sentence. We didn't choose all the sentences, but we only chose the ones with a score below 3. Finally, the 33 samples in the "correct" test set were failed.

After combining these two methods, 160 failure cases were selected from the total 359 test cases. The extract failure cases can be divided into three types:

1. **Poor choices of what to leave out** Sometimes, models tend to leave words behind in order to simplify the sentences. However, the words they leave behind may hinder the original meaning of the sentence.

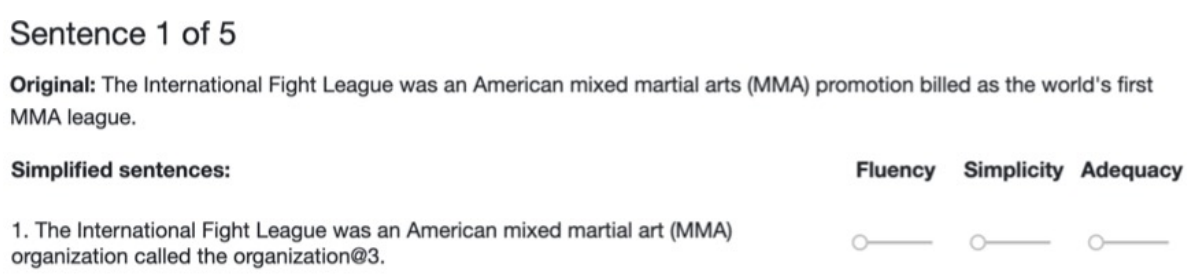


Figure 7.1: Human evaluation of TS user interface[38]

2. **faulty use of synonyms** The model tends to change one complex word to another. These transformations are prone to errors. Even though the transformation of the phrase seems correct, the meaning of the sentence changes after the transformation.

3. **Introduce 'knowledge' from outside of the original text** When the model tends to simplify the sentences, they will most likely add some words to explain the sentence. Instead, some of the added words are irrelevant because they change the meaning of the sentences.

Table 7.2 shows examples of the three mentioned failure cases.

Examples of Failure cases	
Poor choices of what to leave out	
Original Sentence	In 1990, she was the only female entertainer **allowed to** perform in Saudi Arabia.
Simplified Sentence	In 1990, she was the only female **to** perform in the country of Saudi Arabia.
Faulty use of synonyms	
Original Sentence	She **performed** for President Reagan in 1988's Great Performances at the White House series
Simplified Sentence	She **sang** for President Ronald Reagan in 1988's Great Performances at the White House series
Introduce 'knowledge' from outside of the original text	
Original Sentence	For Rowling, this scene is important because it shows Harry's **bravery** in finding Cedric's corpse
Simplified	For Rowling, this scene is important because it shows Harry's bravery. When he finds Cedric's body,**he shows that he cares**.

Table 7.2: Examples of Failure cases

7.4. Human intelligence task

7.4.1. Task overview

For the crowdsourcing tasks that capture Really-Know and Should-Know, we conducted a pilot study (discussed in section 4.2.3 and 5.0.2) to investigate whether the estimated time and our task design for each task were precise. Based on the feedback from the pilot study, we have made the following changes to make it more understandable. Followed our pilot study setup, there are a total of 809 tasks (649 validation sets and 160 failure cases) for both Really-Know and Should-Know tasks. Three annotations are required for each annotation tasks in order to obtain high-quality results.

7.4.2. Quality control

In order to carry out quality control, we mainly look for those crowd workers that were not computer science majors but had high English proficiency. Because the professional English proficiency ensures that the crowd workers can more easily find the appropriate rules and also choose the rules more accurately in this case, it is also worth noting that the task only allows them to work on their personal computers, as the rules selection may not work well on devices with smaller screens such as smartphones, so the effect is not very good. Apart from this, we also require crowd workers to have an education level equal to or higher than that of an undergraduate. In addition, they must complete the training task with more than 90% accuracy and read the task description to understand how the application works clearly enough.

7.4.3. Acceptance criteria

This paragraph talks about our acceptance criteria, during which the crowd workers must complete the task for more than a minute and a half unless we will reject this annotation. Except for this, responses are manually checked and accepted to avoid meaningless annotations, the acceptance criteria with the following acceptance criteria.

1. At least one lexical rule should be selected.
2. At least one syntactic rule should be selected.
3. For a Really-Know task, rules match with the modified part in the sentence.
4. For the Should-Know task, rules match with the modified part in the sentence.

7.5. Data Aggregation

As mentioned in the previous section, we collect three annotations for each pair of sentences. We now process them with two criteria: the first is that as our extracted rules all take the same complex word, the majority vote algorithm is applied. Another criterion is that if the first few parts are different words, we combine these rules in a join set since every crowd worker can identify transformations that others have not found.

7.5.1. Results of Really-Know task and Should-Know test on validation set

For the validation set, we have got 649 pairs of sentences, 1947 annotations for Really-Know tasks, and 1947 annotations for Should-Know tasks. Table 7.4 shows details about the Really-Know rules we collected for the Really-Know task. Through experimentation, we ended up with 1528 rules, including 869 lexical rules and 659 syntactic rules. We found that the number of syntactic rules is smaller than that of lexical rules. From my point of view, this situation may be caused by the following two reasons: one is that the sentence simplification operation itself has more lexical transformations. Second, if no retrieved rules exist, it is difficult to describe the syntactic changes. As a result, sometimes crowd workers tend to ignore them. However, since we are only using the large size of the knowledge base, there may still be some existing rules that may not be matched, and human annotators may not be able to describe these simplified operations. Therefore, we believe that the size of the knowledge base is related to the number of rules; that is to say, through the above analysis, it is not difficult to find that if the size of the knowledge base increase, the collected syntactic rules may also increase.

Results for the Really-Know task on the validation set	
	Aggregated result
Number of all rules	1528
Number of All lexical rules	869
Number of All Syntactic rules	659
Number of Average rules	2.25
Number of Average lexical rules	1.27
Number of Average syntactic rules	0.79
Number of tasks	649

Table 7.3: Results of the Really-Know task on the validation set

Table 7.4 shows the rules we collected for the Should-Know task. According to Table 7.4, the number of all rules we collected is less than the number of all collected rules for a Really-Know task. This can happen for two reasons, through preliminary research has found that it is difficult for crowd workers to describe the simplified operations in the Should-Know task. Especially in the absence of saliency maps, crowd workers have to pay more attention to discovering the relationships between words. And as a result, sometimes they may overlook some simplified operations. Moreover, the study also points out that the way we currently search for retrieving rules may not be able to find all the possible rules.

Results for the Should-Know task on the validation set	
	Aggregated result
Number of All rules	1302
Number of All lexical rules	781
Number of All Syntactic rules	521
Number of Average rules	1.92
Number of Average lexical rules	1.15
Number of Average syntactic rules	0.77
Number of Tasks	649

Table 7.4: Results of the Should-Know task on the validation set

Results for the Really-Know task on the test set	
	Aggregated result
Number of All rules	476
Number of All lexical rules	296
Number of All Syntactic rules	170
Number of Average rules	2.98
Number of Average lexical rules	1.85
Number of Average syntactic rules	1.06
Number of Tasks	160

Table 7.5: Results of the Really-Know task on the test set

7.5.2. Results of Really-Know task and Should-Know test on test set

Results for the Should-Know task on the test set	
	Aggregated result
Number of all rules	366
Number of All lexical rules	235
Number of All Syntactic rules	131
Number of Average rules	2.29
Number of Average lexical rules	1.47
Number of Average syntactic rules	0.82
Number of tasks	160

Table 7.6: Results of the Should-Know task on the test set

This paragraph will explain the results of the two tasks we obtained on the test set. As mentioned in the previous part, here, we only need to focus on the failure cases of the test set. So we labeled 160 pairs of sentences as a failure, and then for each couple of sentences, we collected three annotations. While for the Really-Know tasks, we have collected 476 rules. And The number of lexical rules is larger than the number of syntactic rules, which is the same as that in the validation set. Thus, syntactic rules are more complex than lexical rules, and in the process of simplification, models tend to be more inclined to simplify words than to change the syntactic structure.

For the Should-Know task, we collected 366 rules, which contain 235 lexical and 131 syntactic rules. The number of rules we collected for the Should-Know task is smaller than the number of rules we gathered for the Really-Know task. In addition, the average number of rules we collected is less than one. Two reasons can be caused: the first is that the number of syntactic operations is smaller, and the other is that it is difficult for crowd workers to describe these changes.

7.6. Extract the unknowns

Before discussing how we extract the known rules, we first give a clear definition of these extracted rules. In our framework, we classified the information into two categories: known knowns and unknowns. Known knowns is a set of rules that the model has learned, and it is also the rules that human annotators thought it should learn. While Unknowns can be categorized into two types: (1) **known shouldn't know**: It is a set of rules that model learns, but we do not want them to learn. (2)**unknown should know**: It is a set of rules the model does not learn, but human annotators thought they should learn this. Figure 7.2 displays the classification of knowns and unknowns. After setting the definitions of the knowns, we simply extract the

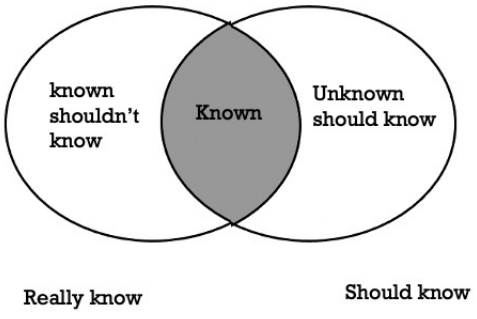


Figure 7.2: Classification of knowns

unknowns by the following steps:

1. The data we aggregated for the Really-Know and Should-Know task is collected from our web application and in the form of **sentence ID, Lexical rules, Syntactical rules**.

2. We then find the intersection of the set that the Really-Know set and the Should-know set. Then, if the rule is in the Really-Know set but not in the intersection set, we treat it as a model known but shouldn't know. Besides, if the rule is in the set that should be known but not in the intersection set, it is classified as the model unknown but should know.

The extracted unknowns for the failure cases in the test set are shown in Table 7.8. We have extracted 672 unknown rules consisting of 388 knowns that shouldn't know and 384 unknown should know rules. The model knows but shouldn't know more rules than the model should know but not know, so in most cases, the model may make mistakes because they know some incorrect knowledge. In contrast, in some other situations, the model may fail because it lacks the proper knowledge of paraphrase.

Results for the unknowns on the test set	
	Aggregated result
Number of all unknown rules	672
Number of all known shouldn't know rules	388
Number of lexical known shouldn't know rules	239
Number of syntactic known shouldn't know rules	149
Number of all unknown should know rules	284
Number of lexical unknown should know rules	169
Number of syntactic unknown should know rules	115

Table 7.7: Results for the unknowns on the test set

7.7. Prediction results

This section talks about how to evaluate the effectiveness of our framework through three different experiments to show whether our framework can be used to predict future failure situations.

7.7.1. Really-Know prediction

Our first experiment is discussed in this subsection. To demonstrate that part of our framework can still predict the failure cases, we evaluate well the Really-Know rules extracted from the validation set cover partial failure cases. So we then compute the coverage by exact matching, and of course we only count rules in the Really-Know set and rules predicted in exactly the same way as the rules from the unknown set. Therefore, The coverage rate can be calculated according to the following formula:

$$coverage = \frac{predicted\ unknown\ rules}{All\ unknown\ rules} \tag{7.1}$$

Table 7.8 shows the Really-Know set coverage for unknown rules in the failure cases. As mentioned above, the unknowns can be divided into two types: known shouldn't know, and unknown should know. And since we only use our Really-Know rule set, we will measure how our Really-Know rule set overrides part of the unknowns(known shouldn't know). Finally, The experimental results showed that the Really-Know set we extracted covers 40.21% of the set of known shouldn't know rules, which also proved that the validity of our partial framework. In addition, the number of syntactic unknowns covered is greater than the number of lexical unknowns covered. The following reasons may cause the previous situation: Lexical changes replace one word with another word, while syntactic changes replace a syntactic structure with another syntactic structure(involving multiple words). There are more lexical substitutions than syntactic structures substitutions, so it is more difficult to predict them.

Results of the Really-Know rules predicts unknowns			
	Known shouldn't know		
Dataset	**Lexical cover rate**	**Syntactic cover rate**	**All cover rate**
Test set	36.40%	46.31%	40.21%

Table 7.8: Results of the Really-Know rules predict unknowns

7.7.2. Should-Know prediction

This part will illustrate the second experiment, which is how to predict the partial unknown by extracting a set of rules that should be known from the validation set. And Table 7.9 shows the exactly calculated coverage rate, which is also calculated from formula 7.1. Compared with the previous experiment, the prediction coverage of the proposed method for unknown rules is lower(failure rule coverage is 34.86%). From my point of view, the Should-Know task differs from the Really-Know task. Really-Know task reflect the simplification done by the model, while the Should-Know tasks reflect the simplification made by crowd workers. Crowd workers provide good quality simplified sentences, while different crowd workers may have different ideas for simplifying the sentences. For example, **propose** can be replaced by **suggest** and **offer**. Both substitutions are correct, but two crowd workers may tend to choose different substitutions. Therefore, it is harder to predict unknown should know.

Results of the Should-Know rules predicts unknowns			
	Unknown should know		
Dataset	**Lexical cover rate**	**Syntactic cover rate**	**All cover rate**
Test set	34.32%	35.65%	34.86%

Table 7.9: Results of the Should-Know rules predict unknowns

7.7.3. Unknowns prediction

This paragraph talks about the most important experiments. Here, we will measure the effectiveness of the entire framework. First, we follow the steps mentioned in section 6.7 to compute the unknowns from the validation set, and then we calculate how the extracted unknowns predict the unknowns of failure cases. And the coverage rate is also calculated by formula 7.1. The results are displayed in Table 7.10. As you can see from the table, a total of 35.27% of the rules are predicted. After explicit analysis, we categorize failure cases into three types: (1). Poor choices of what to leave out. (2).faulty use of synonyms (3). Introduce 'knowledge' from outside of the original text. Based on our research, our framework makes it easier to predict the first(Poor choices of what to leave out) and second failure situation (faulty use of synonyms) since the first and second situation is more likely to be carried out as a rule and is easier to describe by crowd workers. For classification tasks, we can describe the global behavior of each class. In contrast, it is difficult for our task to describe global behavior. Here, I will only describe some examples of local prediction about the first and the second types of failure to show what kind of unknowns types are predicted by my framework. For example, from the failure cases we mentioned earlier, the model tends to ignore **allowed to** which changes the sentence's meaning. This kind of failure also occurs in the validation set. Refers to the second kind of failure, the original sentence is **They are culturally akin to the coastal peoples of Papua New Guinea.** and model simplified it as **They are a lot like the coastal peoples of Papua New Guinea.**. Model changes the word **culturally** to **a lot** while it refers to **culture**. Therefore, the unknown is that culturally - a lot and **culturally - culture**. This kind of mistake is also found in the validation set.

There are still a large number of unknown rules our framework can not predict. And as mentioned before, the third type of unknown is difficult to predict because it is hard to find rules to describe the added external change. For example, according to Table 7.2, the third example adds one sub-sentence **he shows that he cases**. This is described by the rule **bravery-cares**, so it is hard to find the same one in the validation set. Besides, crowd workers tend to input some rules which are not retrieved by the knowledge base. Such a rule is hard to match because other crowdsourcing workers may not find it even if it exists in the sentence. In addition, the number of predicted failure cases was evaluated. Because even though there are multiple

Results of the unknowns predicts unknowns							
Cover rate							
	Known shouldn't know			**Unknown should know**			**All**
Dataset	**Lexical cover rate**	**Syntactic cover rate**	**All**	**Lexical cover rate**	**Syntactic cover rate**	**All**	35.27%
Test set	33.05%	43.62%	37.11%	31.95%	33.91%	32.74%	

Table 7.10: Results of the unknowns predict unknowns

unknown rules in a failure case, we consider that if an unknown rule is included, the failure case is predicted

since some unknown rules are not the cause of the failure, we want to follow this setting to reduce the work burgeon. The formula is shown as follows:

$$predicted\ rate = \frac{Predicted\ cases}{All\ test\ cases} \tag{7.2}$$

Our framework predicted 105 failure cases, the our final predict case rate is **65.6%**.

8

Conclusion and Future work

In this chapter, we will summarize the results of the previous three experiments to answer our research questions. We will then draw conclusions based on the experiment results. In addition, we make a discussion about the limitation and have a vista for the future

8.1. Conclusion

This section will illustrate our findings: we first start with results we extracted to describe what the model really knows, where we employed a human-in-the-loop combined with an interpretable machine learning method(RQ1). Then we talk about that result of the model should learn extracted by applying a human in the loop method and also applying the knowledge base to reduce human cognitive burgeon(RQ2). Next, we talked about the results of the extracted model unknowns in combination with the first two results(RQ3).

8.1.1. Really-know task

Our proposed approach, human-in-the-loop with machine learning interpretable method, compensated for the limitations of the saliency map. We designed the task to efficiently extract knowledge to describe model behavior since we extracted an average of 3 rules for each pair of sentences to explain its changes. In addition, we also applied the format of syntactic rules from the PPDB knowledge base([32]) to describe model syntactic variations. Therefore, our rules are meaningful and easy to understand.

8.1.2. Should-Know task

In our research, Here we also proposed that human intelligence could be applied to describe what the model should learn. For the Should-Know task, we also found an average of three rules (both lexical rules and syntactic rules) for each pair of sentences. Therefore, to sum up, our rule-based applications described what the model should learn.

8.1.3. Predict the unknowns

Here we combine the Really-Know tasks and the Should-Know tasks, and then we can get a set of unknown rules. Based on our findings, models tend to be error-prone in the following three situations:(1). poor choices of what to leave out (2). Faulty use of synonyms (3). Introduce 'knowledge' from outside of the original text. However, our proposed model is more inclined to predict first two failures, while the third type of failure is more difficult to be predicted. Our evaluation shows that our framework can predict 35.27% of the unknowns in the test set and 65.6% of failure cases. Thus, it proves that our framework can predict when and under what circumstance our model will fail.

8.2. Limitations and future work

There are several limitations we want to address, so our future work can be done in the following directions:

1. Text simplification can be divided into the following operations: adding, deleting, and instituting. Even though our application can describe all these simplification operations, it performs well in terms of

instituting. So in the future, we consider that we can continue to add sub-tasks to describe adding and deleting operations continuously.

2. In the case of classification, Our Really-Know and Should-Know application is still complex and time-consuming compared to human intelligence tasks. Therefore, we can further study how to improve the crowd-sourcing interface and reduce human cognitive load in the future.

3. In terms of annotation quality, we can continue to improve its quality. For example, nowadays, we only collect three annotations for each task. Then in the future, we can collect more annotations for it, which will help us to describe the model behavior better and more completely.

4. We can also migrate our framework to other language generation tasks on the application of the framework since now we only apply our framework towards text simplification tasks. So in the future, we can evaluate the performance of our framework in some other language generation tasks like question answering.

5. As for failure prediction, since the way we currently evaluate our method is to see whether our framework can predict future failure work. So same as this, our future work can continue to work in this direction, continuously improving our models' ability to correct these unknowns.

Bibliography

[1] 2.5 million people in NL can't read or write properly, costing society over €1bn - DutchNews.nl — dutchnews.nl. https://www.dutchnews.nl/news/2018/04/2-5-million-people-in-nl-cant-read-or-write-property-costing-society-over-e1bn/. [Accessed 11-Aug-2022].

[2] Suha S Al-Thanyyan and Aqil M Azmi. Automated text simplification: A survey. *ACM Computing Surveys (CSUR)*, 54(2):1–36, 2021.

[3] J Alammar. Ecco: An open source library for the explainability of transformer language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 249–257, 2021.

[4] André Altmann, Laura Toloşi, Oliver Sander, and Thomas Lengauer. Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347, 2010.

[5] Fernando Alva-Manchego, Louis Martin, Carolina Scarton, and Lucia Specia. Easse: Easier automatic sentence simplification evaluation. *arXiv preprint arXiv:1908.04567*, 2019.

[6] Fernando Alva-Manchego, Louis Martin, Antoine Bordes, Carolina Scarton, Benoît Sagot, and Lucia Specia. Asset: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations. *arXiv preprint arXiv:2005.00481*, 2020.

[7] Meghna P Ayyar, Jenny Benois-Pineau, and Akka Zemmari. Review of white box methods for explanations of convolutional neural networks in image classification tasks. *Journal of Electronic Imaging*, 30 (5):050901, 2021.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[9] Agathe Balayn, Panagiotis Soilis, Christoph Lofi, Jie Yang, and Alessandro Bozzon. What do you mean? interpreting image classification with crowdsourced concept extraction and analysis. In *Proceedings of the Web Conference 2021*, pages 1937–1948, 2021.

[10] Or Biran, Samuel Brody, and Noémie Elhadad. Putting it simply: a context-aware approach to lexical simplification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 496–501, 2011.

[11] John Carroll, Guido Minnen, Yvonne Canning, Siobhan Devlin, and John Tait. Practical simplification of english newspaper text to assist aphasic readers. In *Proceedings of the AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, pages 7–10. Citeseer, 1998.

[12] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832, 2019.

[13] Misha Denil, Alban Demiraj, and Nando De Freitas. Extraction of salient sentences from labelled documents. *arXiv preprint arXiv:1412.6815*, 2014.

[14] Yue Dong, Zichao Li, Mehdi Rezagholizadeh, and Jackie Chi Kit Cheung. Editnts: An neural programmer-interpreter model for sentence simplification through explicit editing. *arXiv preprint arXiv:1906.08104*, 2019.

[15] Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77, 2019.

[16] Spencer Frei, Yuan Cao, and Quanquan Gu. Agnostic learning of a single neuron with gradient descent. *Advances in Neural Information Processing Systems*, 33:5417–5428, 2020.

[17] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, 2013.

[18] Colby Horn, Cathryn Manduca, and David Kauchak. Learning a lexical simplifier using wikipedia. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 458–463, 2014.

[19] J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch, 1975.

[20] Daniel D Lee, P Pham, Y Largman, and A Ng. Advances in neural information processing systems 22. Technical report, Tech. Rep., Tech. Rep, 2009.

[21] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[22] Qi Li, Fenglong Ma, Jing Gao, Lu Su, and Christopher J Quinn. Crowdsourcing high quality labels with a tight budget. In *Proceedings of the ninth acm international conference on web search and data mining*, pages 237–246, 2016.

[23] Kuo-Yi Lin, Yuguang Liu, Li Li, and Runliang Dou. A review of explainable artificial intelligence. In *IFIP International Conference on Advances in Production Management Systems*, pages 574–584. Springer, 2021.

[24] Louis Martin. *Automatic sentence simplification using controllable and unsupervised methods*. PhD thesis, Sorbonne Université, 2021.

[25] Louis Martin, Benoît Sagot, Eric de la Clergerie, and Antoine Bordes. Controllable sentence simplification. *arXiv preprint arXiv:1910.02677*, 2019.

[26] Louis Martin, Angela Fan, Éric de la Clergerie, Antoine Bordes, and Benoît Sagot. Muss: multilingual unsupervised sentence simplification by mining paraphrases. *arXiv preprint arXiv:2005.00352*, 2020.

[27] Matej Martinc, Senja Pollak, and Marko Robnik-Šikonja. Supervised and unsupervised neural approaches to text readability. *Computational Linguistics*, 47(1):141–179, 2021.

[28] George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244, 1990.

[29] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[30] Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P Dinu. Exploring neural text simplification models. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 2: Short papers)*, pages 85–91, 2017.

[31] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[32] Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, 2015.

[33] Dejan Radovanovic. Introducing natural language interface to databases for data-driven small and medium enterprises. *Data Science–Analytics and Applications*, pages 11–15, 2021.

[34] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[35] Victor Henrique Alves Ribeiro, Paulo Cavalin, and Edmilson Morais. A dynamic multi-criteria multi-engine approach for text simplification. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[36] Matthew Shardlow. A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications*, 4(1):58–70, 2014.

[37] Shahin Sharifi Noorian, Sihang Qiu, Ujwal Gadiraju, Jie Yang, and Alessandro Bozzon. What should you know? a human-in-the-loop approach to unknown unknowns characterization in image recognition. In *Proceedings of the ACM Web Conference 2022*, pages 882–892, 2022.

[38] Kim Cheng Sheang and Horacio Saggion. Controllable sentence simplification with a unified text-to-text transfer transformer. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 341–352, 2021.

[39] Attend Show. Tell: Neural image caption generation with visual attention kelvin xu. *Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio arXiv (2015-02-10) https://arxiv. org/abs/1502.03044 v3*.

[40] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[41] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[42] Hendrik Strobelt, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander M Rush. S eq 2s eq-v is: A visual debugging tool for sequence-to-sequence models. *IEEE transactions on visualization and computer graphics*, 25(1):353–363, 2018.

[43] Elior Sulem, Omri Abend, and Ari Rappoport. Bleu is not suitable for the evaluation of text simplification. *arXiv preprint arXiv:1810.05995*, 2018.

[44] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[45] António Teixeira, Vera Lúcia Strube de Lima, Luís Caldas de Oliveira, and Paulo Quaresma. *Computational Processing of the Portuguese Language*. Springer, 2008.

[46] Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020.

[47] Jesse Vig. A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*, 2019.

[48] Eric Wallace, Jens Tuyls, Junlin Wang, Sanjay Subramanian, Matt Gardner, and Sameer Singh. Allennlp interpret: A framework for explaining predictions of nlp models. *arXiv preprint arXiv:1909.09251*, 2019.

[49] Maonan Wang, Kangfeng Zheng, Yanqing Yang, and Xiujuan Wang. An explainable machine learning framework for intrusion detection systems. *IEEE Access*, 8:73127–73141, 2020.

[50] Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1015–1024, 2012.

[51] Wei Xu, Chris Callison-Burch, and Courtney Napoles. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297, 2015.

[52] Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415, 2016.

[53] Xingxing Zhang and Mirella Lapata. Sentence simplification with deep reinforcement learning. *arXiv preprint arXiv:1703.10931*, 2017.

[54] Sanqiang Zhao, Rui Meng, Daqing He, Saptono Andi, and Parmanto Bambang. Integrating transformer and paraphrase rules for sentence simplification. *arXiv preprint arXiv:1810.11193*, 2018.

[55] Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1353–1361, 2010.

[56] Chengqing Zong, Jian-Yun Nie, Dongyan Zhao, and Yansong Feng. natural language processing and chinese computing. *Communications in Computer & Information Science, vo*, 333(3):262–273, 2012.