Bauhaus-Universität Weimar
Faculty of Media
Degree Programme Computer Science for Digital Media

Neural-Based Interactive Paraphrasing Tool

Master's Thesis

Fatema Merchant
Matriculation Number 119431
b. Nov 18, 1993 in Kolkata, India

1. Referee: Prof. Dr. Benno Stein
2. Referee: PD. Dr. Andreas Jakoby

Submission date: October 28, 2020

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, October 28, 2020

. . . . . .

Fatema Merchant

ABSTRACT

Existing work on paraphrasing takes a text span (usually a sentence) as an input and generates a new paraphrased text as an output. This setting, however, confines the ability of users to interact with the paraphrasing process by controlling what text segments should (not) be paraphrased, applying some preferences for the substitute segments, and enriching the text with some additional segments. In this thesis, we propose an interactive tool for text paraphrasing. The tool segments the input text into phrases and allows the users to paraphrase these phrases (using four operators), adding new phrases and replacing segments with new ones based on various suggestions provided using several neural-based language models. We evaluate this tool with a user study that examines the effectiveness of different aspects of the tool. The study shows that users appreciate the interactivity of the tool and manage to use it for generating good quality paraphrases. We also analyze the users' behaviors (e.g., operator usage) and their feedback on the tool. The users point to the importance of providing new functionality that helps rearrange the structure (syntax) of a sentence.

Let no one ever come to you, without leaving happier

— Mother Teresa

ACKNOWLEDGEMENTS

There are numerous people I would like to acknowledge for their contribution in this thesis work.

I would like to express my sincere appreciation to Prof. Dr. Benno Stein, who provided me the opportunity to write my thesis at the chair.

I am deeply indebted to my advisors, Dr. Khalid Al-Khatib and Shahbaz Syed, for their guidance, constant support, and providing ideas and alternatives to problems I encountered along the way, as well as giving me the opportunity to work on a very innovative and new idea of research.

Finally, my profound gratitude goes to my loving parents and my dear husband for their unfailing support and continuous encouragement.

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION AND AIM

A simple definition of a 'paraphrase' is an alternative expression of the same thing. Paraphrasing is providing an alternative surface form in the same language expressing the same semantic content as the original form (Madnani et al. 2010). Automatic paraphrasing is important in many fields such as text reuse in journalism, quality enhancement of customer-written reviews, text anonymization, writing style transformation, and text simplification (Burrows et al. 2013; Zhao and Wang 2010).

Many paraphrasing tools are available in the market that are developed on top of machine learning (e.g., deep learning) models, and often adopt the end-to-end scenario: the input is a sentence, and the output is a new generated paraphrased version of it. Most notably is that the users here do not get any option to control the paraphrasing process and add their preferences (i.e., writing style) in the output. To address this limitation, the goal of this thesis is to develop a paraphrasing tool that gives its users more control over the paraphrasing process.

Approaching this goal, we develop a new prototype of *interactive paraphrasing* tool using neural-based language models. In particular, the tool divides the input sentence into small phrases and then lets the user control the paraphrasing process with the help of different operators that suggests various paraphrasing options, including replacing phrases with synonyms or alternative phrases. We conduct a comprehensive user study of the newly developed tool to collect different insights of how different users would use the tool for different types of sentences. As such, we get feedback from several people for future improvements.

1.2 RESEARCH QUESTIONS

For developing a new interactive paraphrasing tool, several research questions can be raised. In specific, we mainly tackle the following:

- How can paraphrasing be done in an interactive fashion? What options should be provided to the user?
- How can we utilize state-of-art neural models for developing interactive paraphrasing tools? How they differ in performance from other statistical methods?
- How interactive paraphrasing tools should be evaluated? Which aspects should be measured, and how the text genre and user profiles should be considered?

1.3 THESIS CONTRIBUTION

In line with the research questions, the main contributions of this thesis involve the design, development, and evaluation processes for the interactive tool.

The design includes finding ways to divide the sentence into small phrases. The decisions to include different operators that can be used for paraphrasing, distinguishing those which are feasible under the current state of language technologies.

The development covers the selection of the neural models that can be used for developing the designed operator, and how these models can be utilized (e.g., modified) to produce the expected outputs of the operators.

Lastly, the evaluation includes a detailed user study where both native and non-native English speakers measure the usability and effectiveness of the developed operators for paraphrasing sentences from different genres such as scientific text and user-generated content in social media.

1.4 THESIS FINDINGS AND OUTCOME

According to the user study, the users like the interactivity of the tool, as it gives them control over paraphrasing, which is not present in other paraphrasing tools that generate paraphrases fully automatically. The result of the study suggests that the first step of breaking the sentence down into small phrases is

a strong means to achieve interactivity in paraphrasing. The most commonly used paraphrase operator was the one that replaces a word/phrase with an alternative and the operator that suggests a synonym for a word. In addition to the existing operators, users requested a new functionality for rearranging words in a sentence, which in their opinion is essential in paraphrasing. The evaluation results also show that the neural-models outperform the statistical netspeak model. The neural models are pre-trained on sequential data of large size. Hence, they produce high-quality suggestions, without even the need for fine-tuning them for our use.

1.5 STRUCTURE OF THESIS

The following chapters of the thesis are structured as follows.

- Chapter 2 discusses the current state of art in paraphrasing. Besides, it introduces neural language models. Then, it describes briefly interactive interfaces and their design. Lastly, it discusses different strategies for tool evaluation.
- Chapter 3 describes in detail how our proposed interactive paraphrasing tool is designed and developed. Next, it explains how different neural models are used as a building block in the tool. Lastly, it describes an alternative approach for building the tool using Netspeak [1].
- Chapter 4 describes the user study design, its goal, and how it is performed.
- Chapter 5 provides the evaluation result and the review of the tool as well as its features from data and feedback collected in a user study.
- Chapter 6 finally concludes the thesis with a discussion of the results and the proposed future work.

1 https://netspeak.org/

2 BACKGROUND AND RELATED WORK

This chapter gives an overview of the state-of-art in paraphrasing. We begin by introducing some work on paraphrasing and its applications. Then we introduce neural language models for text generation and text prediction. Next, we discuss interactive interfaces and tool evaluation strategies.

2.1 PARAPHRASING

It is known, the solution to paraphrasing is one of the challenging things to achieve in Natural Language Processing (NLP). Research in this field is being done for a long time now, therefore there are different approaches available to do paraphrasing, from rule-based, statistical paraphrasing to using the different neural models.

Zhao, Lan, et al. 2009 used the method of statistical paraphrasing generation for paraphrasing. This approach was designed considering the three paraphrasing applications: sentence simplification, sentence compression, and sentence similarity. The Statistical Paraphrase Generation approach has three processing steps: sentence preprocessing, paraphrase planning, and paraphrase generation (Zhao, Lan, et al. 2009). POS tagging and dependency parsing are done in the first step of preprocessing, they give information about the pattern of the sentence and the data is further used for resource collocation. In planning, target units for paraphrasing are selected, and the candidate paraphrases from multiple resources are selected to replace the target unit. The last step of the paraphrase generation generates paraphrases using a statistical model by selecting a target candidate. As in this approach, in our work, there is a preprocessing step, but instead of POS tagging, *noun phrase chunking* is done. And, after chunking, the users get the control of the paraphrasing process.

Conventional paraphrase generation methods used rules or statistical machine learning principles for paraphrase generation, Prakash et al. 2016 used

deep learning model. They used a stacked residual LSTM based network, in which LSTM layers have a residual connection (Prakash et al. 2016). A 4 layer residual LSTM is used to generate paraphrases, given an input sentence, the residual network outputs a paraphrase. They use PPDB 2.0, WikiAnswers, and MSCOCO paraphrase corpora for training their models and the training time for them is in the range of 14 - 36 hours.

Mallinson et al. 2017 used neural models with machine translation based on bilingual pivoting to generate paraphrases. To train the model they use The Multiple-Translation Chinese (MTC) corpus, The Jules Vernes Twenty Thousand Leagues Under the Sea novel corpus (French to English), and Wikianswers corpus, which has one question and many answers to it. They first translate the sentence from English to another language, which is again translated back to English. To perform this translation they used an encoder-decoder model built using a recurrent neural network. The input/output pattern of this model is the same, given an input sentence, the model generates paraphrases.

Z. Li, Jiang, Shang, and H. Li 2018 used Deep Reinforcement Learning to Generate Paraphrases. In their approach, they propose a framework, consisting of a generator and evaluator. The generator generates paraphrases of an input sentence, which is a sequence-to-sequence model. The evaluator is a deep matching model, which verifies if the given sentence and generated sentence are paraphrases of each other. Quora question pairs and Twitter URL paraphrase corpus are used for training and evaluation of their models.

Z. Li, Jiang, Shang, and Q. Liu 2019 introduced the concept of a decomposable neural network for paraphrase generation. They have used a Transformer based model to train and generate paraphrases at granularities of different levels. This model consists of a separator, encoders, decoders, and an aggregator. The separator divides the sentence into segments of different granularities. Then multiple encoders and decoders of respective granularities process the segments in parallel. Lastly, the aggregator produces the paraphrased sentence by combining the outputs from all the decoders. To do the training WikiAnswers paraphrase, and Quora duplicate question pairs are used.

Most of the paraphrasing approach trains a model, such that given a sentence they generate paraphrases, where the user does not have a choice of what should be changed, added, or deleted. Also, the training needs a large amount of data either parallel paraphrase corpus or bilingual corpus as well as time. We use existing language models for generating suggestions for paraphrasing. With the help of the tool the user can generate different paraphrases for a sentence, instead of just one which the trained models generate. The interactivity in the tool developed by us gives the user more control over the paraphrasing process.

2.2 LANGUAGE MODELS FOR INTERACTIVE PARAPHRASING

The Transformer has become the dominant architecture for natural language understanding and generation by replacing older recurrent neural network models such as the long short-term memory (LSTM). As the Transformer are not sequential, they enable more parallelization during training, which enables training on larger datasets than was feasible prior to its implementation (Wolf et al. 2019).

Language models have traditionally only been able to read text input sequentially — either left-to-right or right-to-left — but they could not do both at the same time. The model BERT which stands for Bidirectional Encoder Representations from Transformers is distinct because it is designed to read at once in both directions (Devlin et al. 2018).

The BERT is pre-trained on two different, but related, NLP tasks using its bidirectional capability: Masked Language Modeling and Next Sentence Prediction. The Masked Language Model (MLM) task hides a word in a sentence and then the program predicts the hidden (masked) word based on the context of the hidden word. In the Next Sentence Prediction approach, given two sentences the program predicts if the sentences have a logical, sequential connection or they just have a random relationship. The BERT was pre-trained using only a plain text corpus (i.e. unlabelled data) from Wikipedia text and Books Corpus.

Along with BERT, we mainly use RoBERTa model to build the tool. The RoBERTa is fine-tuned on BERT and we use it to help generate suggestions

for paraphrasing (Y. Liu et al. 2019). As these models are already trained on a large amount of data, and readily available to use, no time is put into training or fine-tuning them.

2.3 INTERACTIVE INTERFACE

Interactive Interface is meant for human-computer interaction and communication on an app, webpage, or a device. The interactive interfaces allow users to control computers or devices that they are interacting with efficiently and effectively. A good interactive interface is one that is intuitive, efficient, and user-friendly.

User Experience Honeycomb framework by Peter Morville (Morville 2004) is a guide to user interface design. The framework describes the qualities of user experience design:

- *Usable:* The application should be designed in a way that is familiar and easy to understand. The learning curve a user must go through should be as short and painless as possible.
- *Useful:* A application needs to be useful and fill a need.
- *Desirable:* The visual aesthetics of the product, service, or system need to be attractive and easy to translate. The design should be minimal and to the point.
- *Findable:* Information needs to be findable and easy to navigate. The navigational structure should also be set up in a way that makes sense.
- *Accessible:* An application should be accessible to those with disabilities.
- *Credible:* An application should be trustworthy, transparent, and secure.

The above mentioned design qualities help in evaluating an interactive interface.

2.4 TOOL EVALUATION

Different strategies like *usage* and *technical performance* are used for evaluating the tool (Ledo et al. 2018). Evaluating usage helps verify which tasks can a target user group perform and which ones remain challenging. Evaluating the technical performance of the toolkit helps find out *how well* it works.

To evaluate usage a user study is performed on the tool, it helps in identifying the success and shortcomings of the tool. To obtain feedback from the participants, they are asked qualitative questions that are answered on the *Likert scale* (Lewis 1995). The likert scale is a psychometric scale, in survey research, it is the most frequently used method to get scaling responses. Along, with the feedback questions, participants are also asked about their experience and challenges in performing the tasks.

Studying Technical performance helps to benchmark, analyze the tool, and validate the tool's performance. Technical performance can be measured in terms of efficiency and accuracy. These measures show whether the tool meets basic usage standards, or if there is a need for improvement. In this work, both the usage and technical performance are used for evaluating the tool.

3 INTERACTIVE PARAPHRASING

In this chapter, we discuss the necessary conceptual work and design decisions contributed by this thesis. The interactive paraphrasing tool is a web-based tool, its design and working are explained in Section 3.1. The internal (backend) implementation of the tool and its features are discussed in Section 3.2.

3.1 TOOL DESIGN

In this section we describe the design and features from the interactive interface of the tool. We also explain how to use the features to paraphrase a sentence.

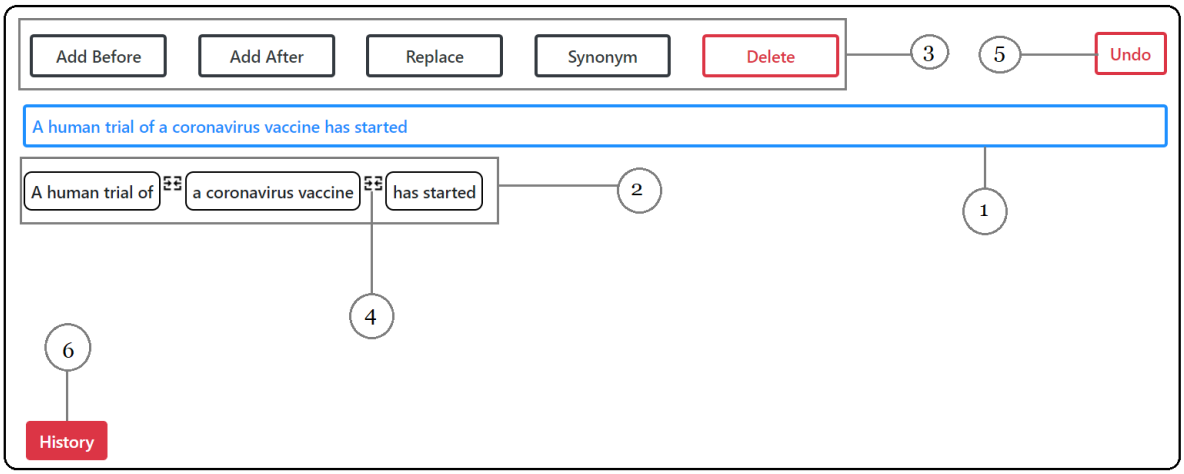


Figure 1: Interactive Paraphrasing tool
1-Input Sentence, 2-Chunks, 3-Operators, 4-Merge, 5-Undo Button, 6-History Button

Figure 1 shows the user interface of the interactive paraphrasing tool. The input sentence (1) is an input field, where the user can write the sentence they want to paraphrase. The first step in paraphrasing after entering an input sentence is dividing the sentence into small phrases called chunks. Once, the sentence is entered into the input field, the pressing of the enter key results in dividing the sentence into chunks (2). The detail, of how chunking is done is discussed in Section 3.2.1.

We have developed different operators (3,4) and features (5,6) that can be applied to chunks for generating paraphrases effectively and efficiently. There are two types of operators (3); first the paraphrasing operators, they are *Add Before/After, Replace* and *Synonym* and the second type of operators provide functionality that will help users in easily paraphrasing a sentence, they are *Merge* (4), *Delete*, and *Split*. The *undo button* (5) on the far right, and the *history button* (6) at the bottom are additional features that may be useful when paraphrasing.

On the top left of the tool are buttons (3), they are the operator that can be applied on the chunks. The buttons (3) are distinguished into two types by color black and red. The black buttons are the paraphrasing operators that help in paraphrasing by retrieving suggestions and the red delete button is for functionality. The other red buttons, undo and history buttons in the interface also provide functionality that would be helpful in the paraphrasing process.

One of the important design decisions was how we represent chunks weather in an input field, which makes it editable or as a non-editable instance, separated in a box. We choose to represent in a box as the phrases appear more clear and intuitive. With this representation, we can add the functionality of the split-merge operator clearly. The merge operator between two chunks with its symbol and position clearly indicates which chunks will be merged. We thought that this design choice would also be easy for the user to understand and use. With chunks separated in boxes, we can simply position the suggestions below them, for a user to easily understand. To distinguish the suggestions from other elements in the interface, we show them in green color.

Split: Tplit operator is used to split words in a chunk into separate chunks, double-clicking on a chunk activates the split operator. The split operator can be used to split a chunk with more than one word, and then paraphrasing operators can be applied on any of the chunks to get a suggestion. Consider the example of input sentence and chunks in Figure 2, here the user would like to add a word before *coronavirus vaccine* and therefore the user would like a

suggestion for the same. In this scenario, the user should first split the chunk *a coronavirus vaccine* into separate chunks by double-clicking on it. After splitting, the chunks are as seen in Figure 3. After applying the split operator, the chunk is split into three separate chunks - *a, coronavirus,* and *vaccine*.

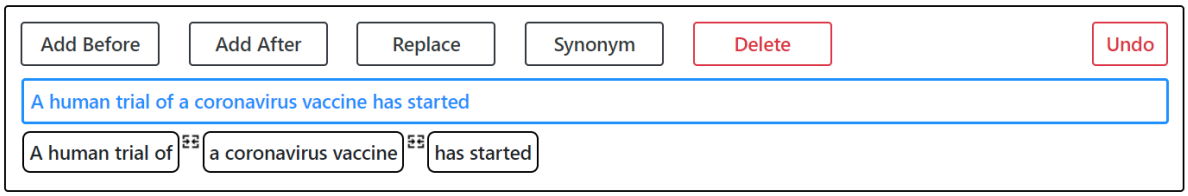


Figure 2: Input sentence and it's chunks

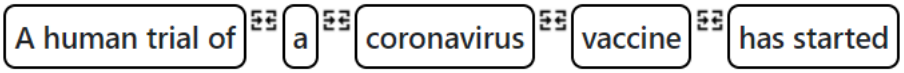


Figure 3: Applying split operator

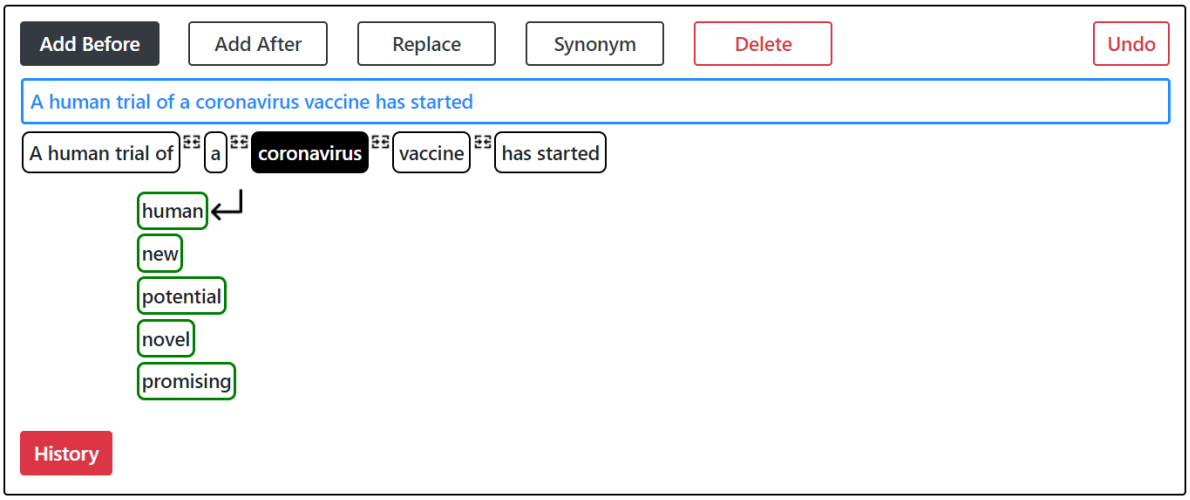


Figure 4: Applying add before operator on a chunk

Add Before: The add before operator fetches suggestions that can be added before the selected chunk. Continuing with the example, here the user wants to add a word before the chunk *coronavirus*. Now as the chunk is split, the user will select the chunk *coronavirus* and then click on Add Before operator. In Figure 4 you can see the suggestion after applying the Add Before operator. There is an arrow pointing to the left, this arrow indicates that the selected suggestion will be added before the word *coronavirus*. Consider, the suggestion

promising is selected. In Figure 5, you can see the updated chunks after the suggestion *promising* is selected.

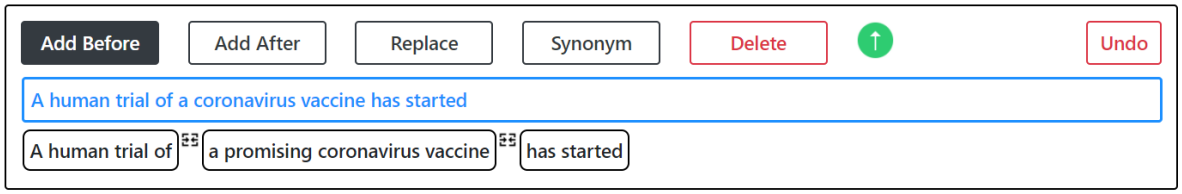


Figure 5: Chunks after selecting a suggestion from suggestions of add before operator

Feedback Metric: The feedback metric indicates the score of the current paraphrased sentence, in comparison to the previous version of the paraphrased sentence. The score is shown as an arrow that indicates if there is an improvement or not in the sentence after a suggestion is selected. In Figure 5 you can see an arrow pointing up, in a green circle after the delete operator. This arrow is the feedback metric. If the score is better, then it will show a green arrow, if not, then it will show an arrow pointing down in a red circle.

Add After: The add after operator fetches suggestions that can be added after the selected chunk. Consider the chunk *has started* is selected, and Add After operator is applied on this chunk. The suggestions after applying Add After are as seen in Figure 6. As seen in Figure 6 a right-pointing arrow is shown

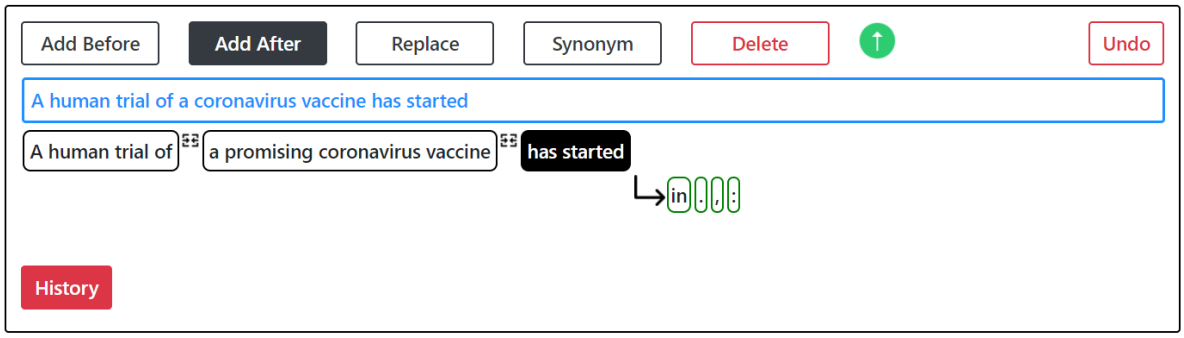


Figure 6: Applying add after operator on a chunk

before the suggestion list, it indicates that, if any suggestion is selected it will be added after the selected chunk. If the suggestion *full-stop (.)* is selected, then the updated chunks will be as in Figure 7. You will also find poor sentence score since the feedback metric arrow is pointing down in a red circle.

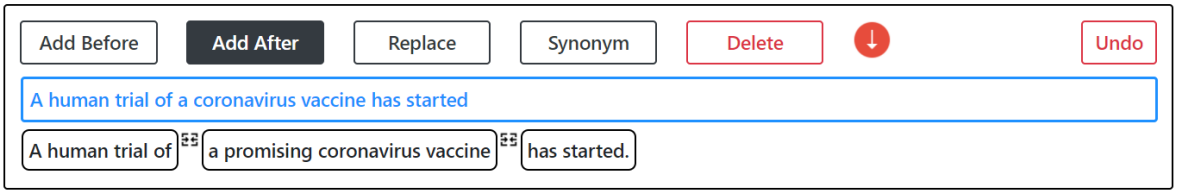


Figure 7: Chunks after selecting a suggestion from suggestions of add after operator

Undo: The undo operator undoes the last change in the paraphrasing process. Considering the example in Figure 7, since the sentence score has become poor, we can undo the last change by clicking on the undo button. After undo the full-stop will be removed as seen in Figure 8. Also, the feedback metric is again improved.

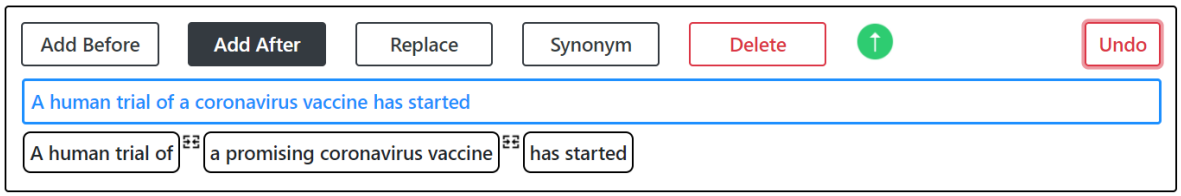


Figure 8: After undo operation

Merge: The merge operator, is used to merge two chunks. It can combine more than one neighboring chunks into one chunk and then paraphrasing operators can be applied to it to get a suggestion. The merge operator can be useful to replace two chunks with one word. As an example, consider the user would like to replace *trial of* with something different and therefore would like a suggestion for the same. So, in this scenario the user can first *split* the chunk *A human trail of* into separate chunks by double-clicking on it. Once the split operation is performed, the chunks will be as seen in Figure 9. In the next

Figure 9: Applying split operator

step, the user merges the chunks *trial* and *of*, by clicking on the merge icon between both the chunks. In Figure 10 you can see the new chunk structure after merging the chunks *trial* and *of*.

Figure 10: Applying merge operator

Replace: The replace operator fetches suggestions that can replace the selected chunk. It can be any suggestion that can be an alternative to the chunk and fits in the context of the remaining sentence. Proceeding with the example in Figure 10, the next step is to get a suggestion to replace the chunk *trial of*. As seen in Figure 11, replace operator is applied on *trial of* chunk, along with its suggestions. If the user selects the suggestion *test*, then *trail of* is replaced with

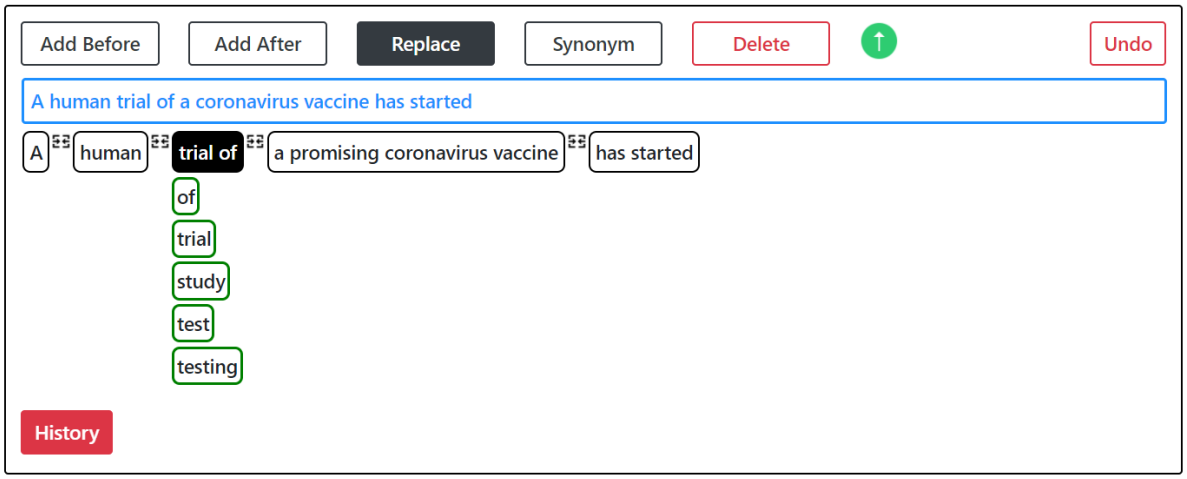


Figure 11: Applying replace operator on a chunk

test and chunks are updated as seen in Figure 12. The sentence looks incom-

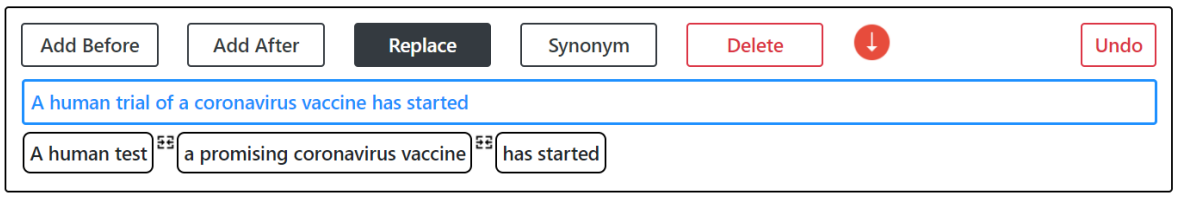


Figure 12: Chunks after selecting a suggestion from suggestions of replace operator

plete, therefore the chunk *A human test* is selected and add after the operator is applied on it, and a suggestion *for* is selected. The updated sentence looks as seen in Figure 13.

Delete: The delete button is used to delete a chunk. For example, the first chunk *A human test for* is split by double-clicking on it, and after splitting the chunks

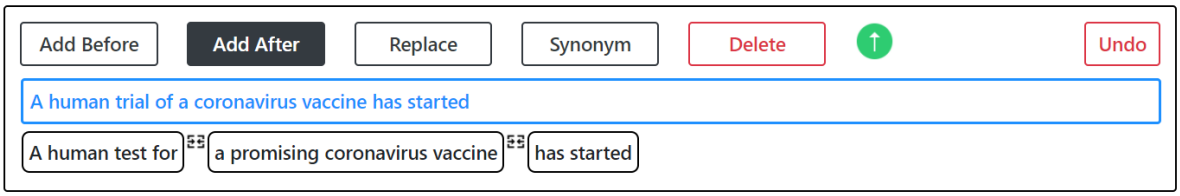


Figure 13

are *A, human, test, for*. Then the chunk *"A"* is deleted. In the Figure 14, the chunk *"A"* is deleted, however, after deleting the chunk *"A"*, the score of the sentence becomes lower, user can either keep the updated sentence as it is, if they find it good else they can also **undo** the last delete operation. The Figure 15

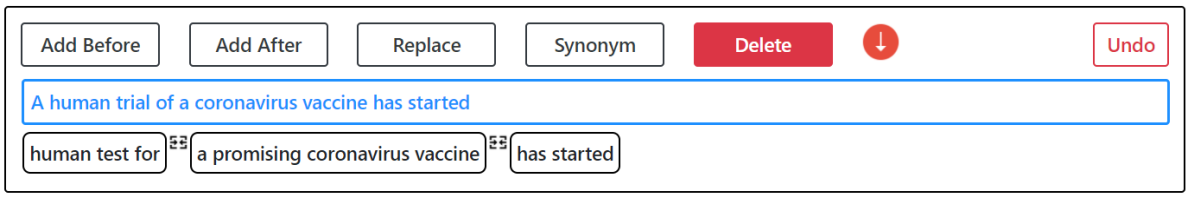


Figure 14: Applying delete operator on a chunk

shows the sentence after the undo operation. Also, after the undo operation the update score of the sentence can be seen, which is better than the previous version.

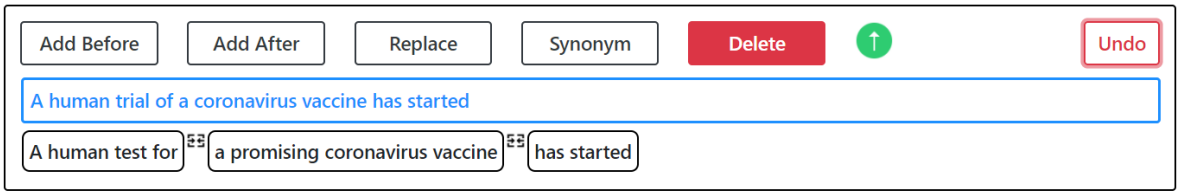


Figure 15: Performing undo operation

History button: It shows all the changes done while doing paraphrasing along with the time at which the changes were performed, as seen in Figure 16. The highlighted word indicates, what was updated/added in the sentence. The history can be useful to check the different options the user tried and then the user can select the most suitable one.

Synonym: The synonym operator fetches synonyms for the selected word. The synonym operator can only be applied to chunks having only one word because synonyms are normally for a word and not for two or more words. Therefore,

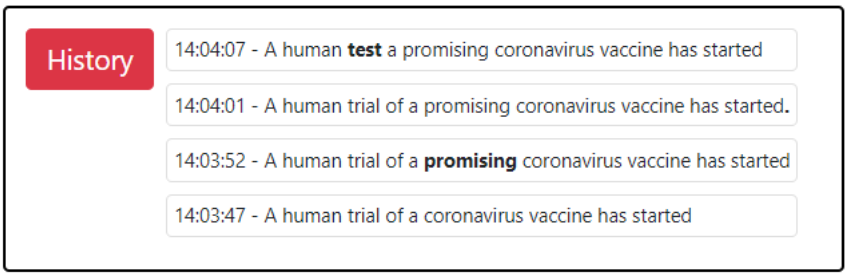


Figure 16: History tracker

if a chunk has more than one word, the user will have to apply a split operator on that chunk and split it. Consider another example sentence for Synonym

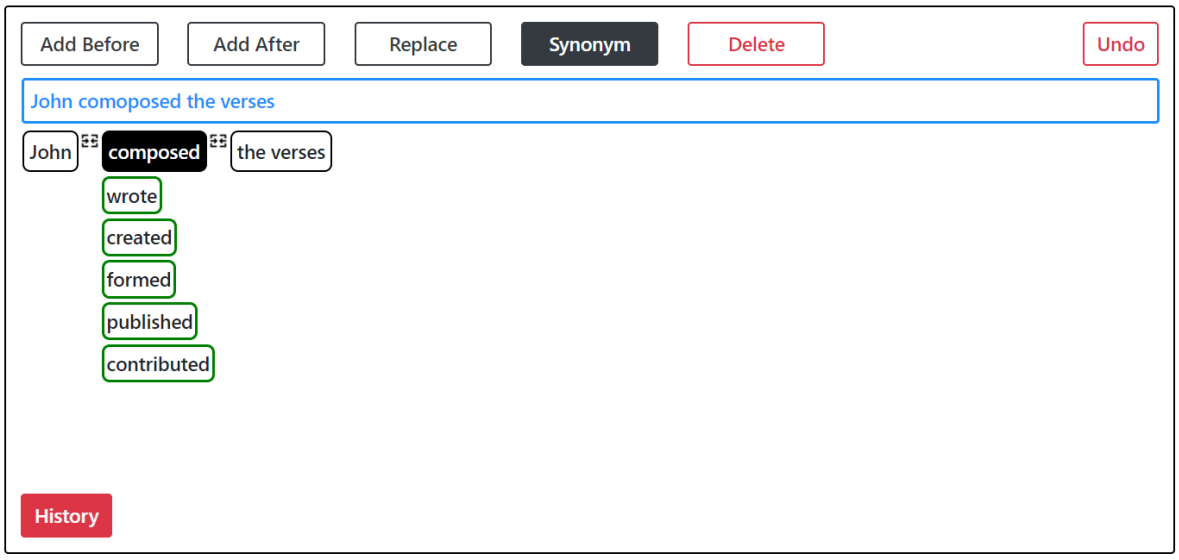


Figure 17: Applying synonym operator

operator, *John composed the verses*. In this example, the synonym operator is applied on the chunk *composed*. In Figure 17, you can see the suggestions from the synonym operator for the word *composed*. As seen, these suggestions can be a substitute for the word *composed*.

Synonym vs Replace: The difference between the synonym and the replace operator is that the synonym operator finds the suggestion that can substitute the selected word, while the replace operator fetches any suggestions that are frequently used in the sentence in the position of this chunk and not necessarily relevant to the meaning of the selected chunk/word. The suggestions for the replace operator can even be a punctuation mark. In Figure 18, it can be seen how different are the suggestions from the replace operator for the same sen-

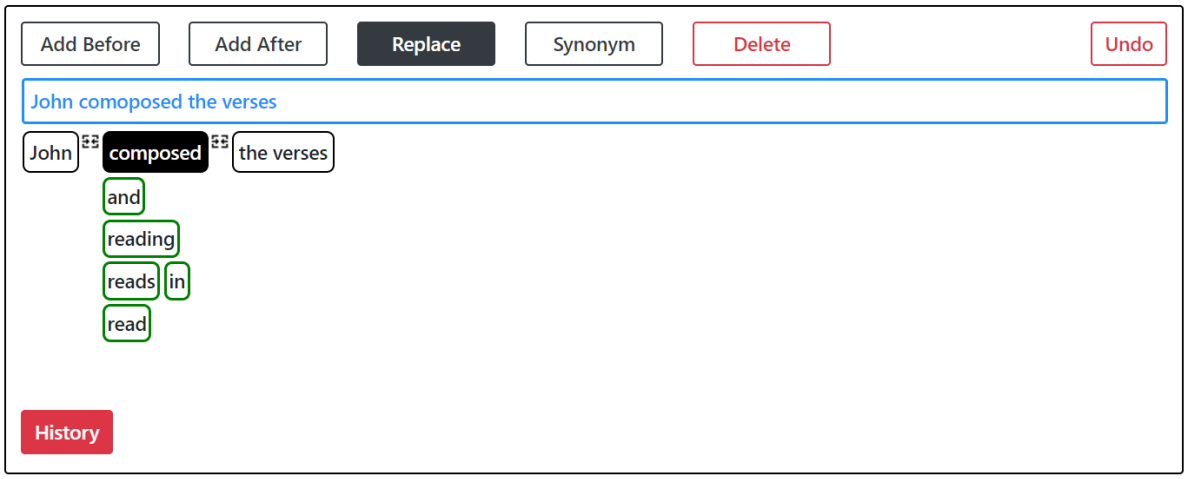


Figure 18: Replace vs Synonym Operator

tence and the same chunk (*composed*) than a synonym operator. The suggestion from the synonym operator are *wrote, created, formed, published, contributed* whereas from the replace operator are *and, reading, reads, in, read*. These suggestions show that the synonym operator suggests words that have the same meaning as the word whose synonym is requested, whereas the replace operator suggests words that can replace the selected chunk and fit in the context of the sentence.

In this section, we discussed the features and how the tool can be used. In the next section, we discuss how chunks are created and how the suggestions are fetched and generated for the paraphrasing operators.

3.2 TOOL IMPLEMENTATION

In the following section, the paraphrasing process is described. The Figure 19 shows the basic flow for paraphrasing a sentence using interactive paraphrasing tool. The paraphrasing process starts by chunking, i.e dividing the sentences into small phrases, it is explained in Section 3.2.1. Then different operators are applied to these chunks to get the suggestions. The operators are built using available neural models, Section 3.2.2 describes the operators in detail. After a suggestion is selected the feedback metric is updated with a new score of the sentence, Section 3.2.3 discusses it. Repeating the process of applying different operators on chunks of user's choice and selecting a relevant suggestion helps to generate a paraphrased sentence.

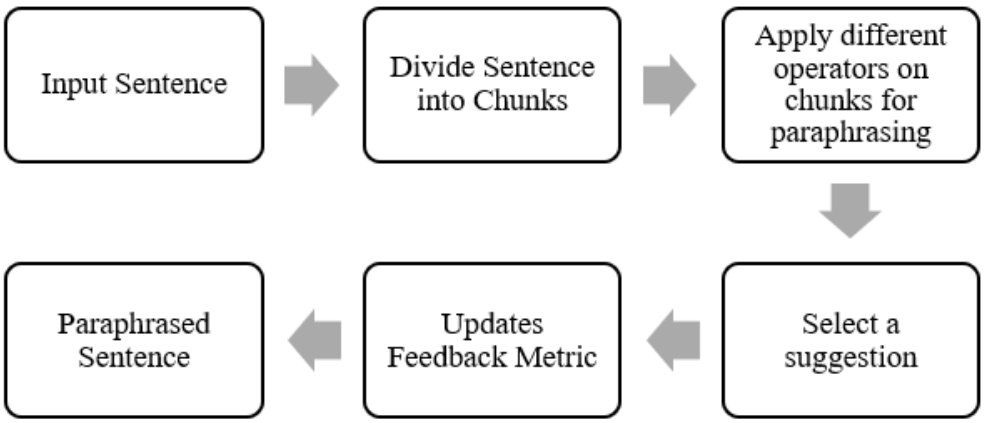


Figure 19: Overview of flow for paraphrasing a sentence

3.2.1 *Chunking*

As mentioned earlier, the main motivation of this thesis is to give the user more access and options when paraphrasing a sentence. To achieve it, in this work the first step is to divide the sentence into small phrases called chunks. The benefit of this division is, it becomes easy for the user to work with individual chunks, then the whole sentence at a time. With the help of suggestions from operators, mentioned in Section 3.2.2, users can enrich a chunk by adding more words or replacing word(s). The chunk is small independent units, on which the user can work at a time. Also, the grouping of words into chunks (basically a phrase) makes it simpler to do paraphrasing. Even if effective changes are made to one chunk, the user can get meaningful paraphrases.

There is also an option to word tokenize the sentence i.e divide the sentence by words (in this scenario each chunk will only contain one word), as seen in Figure 20. The drawback of this approach is, firstly there will be many chunks. Also, the user will have to work with one word at a time, which means they work on the complete sentence as they will have to consider the word in the context of the sentence. This makes the task of paraphrasing a sentence interactively challenging.

The Figure 21 shows an example of a sentence and its chunks, these chunks are generated using FLAIR[1] : An Easy-to-Use Framework for State-of-the-Art

1 https://github.com/flairNLP/flair

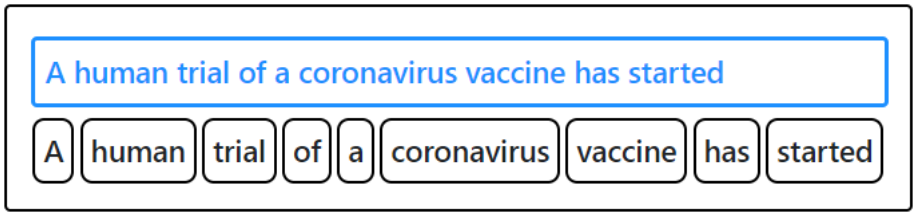


Figure 20: Word tokenized sentence

NLP (Akbik, Bergmann, et al. 2019). With the help of chunking functionality from this framework, the sentence can be divided into the following phrases:

- Noun Phrase
- Verb Phrase
- Prepositional Phrase
- Adverb Phrase
- Adjective Phrase
- Subordinate Clause

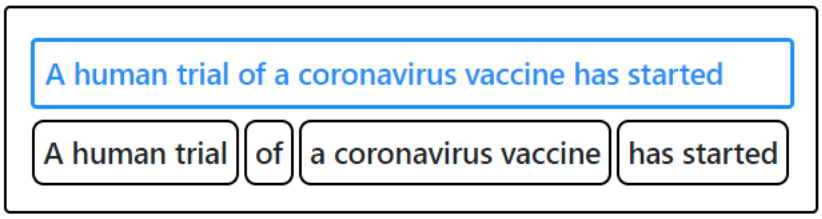


Figure 21: Example of chunking

Flair is built on state of the art sequence labeling task, which is used for performing chunking. This is a syntactic task, in which they tag the noun phrase in the sentence. They have proposed the sequence tagging architecture, where each sentence is passed as a sequence of characters to a bidirectional character-level neural language model, from which they retrieve for each word the internal character states to create a contextual string embedding. This embedding is then utilized in the bidirectional recurrent neural networks - conditional random field (BiLSTM-CRF) sequence tagging module to address a downstream NLP task of phrase chunking (Akbik, Blythe, et al. 2018).

This is a good start to get small phrases, however, as seen in Figure 21 there is a chunk *'of'* that consist of only one word. This chunk is a prepositional phrase, it does not have any meaning by oneself and also increases the number

of chunks. To avoid this, we tried and tested different examples and then we decided to add prepositional phrases with only one word to the previous chunk. This is only done for a prepositional phrase, other phrases with only one word are kept as they are. The examples in Table 1 shows the chunk structure after merging prepositional phrases with one word, to its previous chunk.

No	Sentence	Original Chunk	Restructured Chunk
1	...my friend are flying to...	my friend, are flying, to	my friend, are flying to
2	Racing toward the finish...	racing, toward, the finish	racing toward, the finish
3	...lies in the name of...	lies, in, the name, of	lies in, the name of
4	...distinct coding for all classes of...	distinct coding, for, all classes, of	distinct coding for, all classes of

Table 1: Chunk Restructuring

The original chunks are the chunks as received from Flair's sequence labeling model. The restructured chunks column shows the chunks after merging prepositional phrase chunks with only one word to the previous chunk. This is how we generate chunks.

3.2.2 *Operators*

Once the chunks are available, the next step is using different operators on these chunks for paraphrasing a sentence. The operators are an intuitive way of modifying the chunks or adding words before/after a chunk, which will help users in paraphrasing the sentence step by step. The operators are a way to enrich the sentence by adding adjectives, adverbs to it, or by removing what is not necessary. We have developed four operators using neural models that help in paraphrasing: *Add Before, Add After, Replace, Synonym*.

The operators from the UI make an API request to fetch the suggestions. These APIs are built using *Masked Language Models* from RoBERTa and BERT, which are neural-based language models. The masked language modeling is a fill-in-the-blank task, where a model uses the context words surrounding a <mask> token to try to predict what the <mask> word should be. When making an API request, in the query a keyword <*mask*> is added that indicates for which token the user wants the suggestions i.e at which position the user

wants to add/replace the word. Below are the sample query formats for making an API request to the neural-based operators, and the response from them:

1. **Add Before**

   Consider the following sentence:

   Sentence: *A human trial of a **coronavirus** vaccine has started*

   The user is using *add before* operator and the user wants to add a word before **coronavirus**. The query to get the suggestion to add word before corona virus is:

   Query: *A human trial of a **<mask>** coronavirus vaccine has started*

   The result of this query are suggestions that can be put in place of ***<mask>***. The results from this query are:

   Results: possible, novel, potential, promising, new

   If any of the suggestion is suitable the user can select the suggestion, which will be added to the sentence before coronavirus. For e.g.:

   Example: *A human trial of a **promising** coronavirus vaccine has started*

2. **Add After**

   Consider the following sentence:

   Sentence: *I and my friend are flying to **Malta** tomorrow morning*

   The user is using *add after* operator and the user wants to add a word after **Malta**. The query to get the suggestion to add word after Malta is:

   Query: *I and my friend are flying to Malta **<mask>** tomorrow morning*

   The result from this query are suggestions that can be put in place of ***<mask>***. The results are:

   Results: again, together, late, early, "," (suggestion can also be a punctuation mark)

   If any of the suggestion is suitable the user can select the suggestion, which will be added to the sentence after Malta. For e.g.:

   Example: *I and my friend are flying to Malta **early** tomorrow morning*

3. **Replace**

   Consider the following sentence:

Sentence: *love this idea and **most definitely** want to help*

The user is using *replace* operator and the user wants to replace **most definitely**. The query to get the suggestion to replace most definitely is:

Query: *love this idea and **<mask>** want to help*

The result from this query are suggestions that can be put in place of ***<mask>***. The results are:

Results: we, I, really, just

If any of the suggestion is suitable the user can select the suggestion, which will replace *most definitely*. For e.g.:

Example: *love this idea and **really** want to help*

4. **Synonym**

   Consider the following sentence:

   Sentence: *That one nightmare single handedly **scared** me more than anything else in my entire life*

   The user is using *synonym* operator and the user wants synonym for **scared**. The query to get the suggestion for synonym of scared is:

   Query: *That one nightmare single handedly **<mask>** me more than anything else in my entire life*

   Along with the above query, the word for which synonym is being requested is also send when making api request. The reason is, synonym operator needs to get suggestions of a similar words to the selected word *scared*. The results from this query are:

   Results: frightened, terrified, worried, startled, panicked

   If any of the suggestion is suitable the user can select the suggestion, which will replace *scared*. For e.g.:

   Example: *That one nightmare single handedly **terrified** me more than anything else in my entire life*

We use the available neural model for creating an operator and do not create a new neural model for the paraphrasing tool. The four operators - *Add Before, Add After, Replace* and *Synonym* are built using available neural-based

language models. We use RoBERTa's[2] masked language model (Y. Liu et al. 2019) for the development of *Add After, Add Before* and *Replace* operator. The *Synonym* operator is developed using LSBERT: A Simple Framework for Lexical Simplification[3] (Qiang et al. 2020), which is built using BERT.

ROBERTA MODEL: RoBERTa stands for Robustly optimized BERT. It is a strongly optimized approach for pretraining natural language processing systems. RoBERTa is built on BERT's masked language model, in which the system is trained to predict the masked part of a sentence, which is changed dynamically when training data. The next-sentence prediction objective of BERT is removed from RoBERTa. RoBERTa changes key hyper-parameters in BERT, is also trained with larger batches on more data, for a longer amount of time and longer sequences than BERT. Along with existing unannotated NLP data sets, the CC-News dataset which is generated from public news articles is used for training.

We have used *roberta.large* from RoBERTa's different available model for building operators, it's size is *625 MB*. The *fill_mask* function is used to predict the <*mask*> token in a sentence. To the *fill_mask* function two parameters are passed, first is a sentence with <*mask*> token and second parameter is *topk*, it is the number of predictions for the <*mask*> token to be retrieved.

fill_mask ('Berlin is the <mask> of Germany.', topk=5)

The *fill_mask* function works as follows:

1. The input sequence is encoded using byte pair encoding into id. Also, the index of the masked token is located in the list of ids.
2. The predictions at the index of masked tokens are retrieved in the form of tensor, which is of the same size as vocabulary.
3. Using the *topk* method, *top n* tokens are retrieved.
4. Predicted tokens are decoded and returned.

2 https://github.com/pytorch/fairseq/blob/master/examples/roberta/README.md
3 https://github.com/qiang2100/BERT-LS

The *fill_mask* function is used to get the suggestion for *Add Before, Add After* and *Replace* operators. Once we receive the suggestions from the MLM, the suggestions are put in place of the <*mask*> token one by one, and the score (Section 3.2.3) of the sentence is calculated. The suggestions are sorted based on the score and returned as the response to the API request.

LSBERT- A SIMPLE FRAMEWORK FOR LEXICAL SIMPLIFICATION: To help generate synonyms for a word approach of lexical simplification (LS) is used. The LS framework has three steps: complex word identification, substitute generation of complex words, and filtering and substitute ranking (Qiang et al. 2020). To implement the synonym operator, we have modified this pipeline as per our requirement. The first step of complex word identification is dropped, as the suggestion for every requested word's synonym in the tool is to be given. The next two steps i.e. substitute generation and substitute ranking are used to generate synonyms.

Masked Language Model (MLM) from BERT is used for substitute generation. Once the suggestions are available from MLM, the top 10 candidates are used for filtering and substitution ranking. LSBert computes various rankings according to their scores for each of the features. The different features used for calculating the ranking are; best prediction order, language model feature, semantic similarity, frequency feature, and PPDB (A Paraphrase Database). After obtaining all rankings for each feature, LSBert scores each candidate by averaging all its rankings. Finally, the candidates are sorted based on the highest-ranking and are returned for suggestions as synonyms.

3.2.3 *Feedback Metric*

Perplexity is the most common metric for evaluating language models. It indicates the *naturalness* of a sentence, i.e how high is the likelihood of occurrence of a sequence of words in the given sentence. Perplexity is the property, we show to the users through feedback metric to tell them if their changes improve the overall quality of their sentence.

The feedback metric is updated every time word in the sentence is modified in the paraphrasing process i.e a new suggestion is selected. To calculate the feedback metric of the sentence, Language Model based sentence scoring library[4] is used, in which we use GPT-2 language model. We use *mean* strategy to compute the score of the sentence. With this strategy, the score is computed as the mean of the token's probabilities. This feedback metric can be useful to indicate whether the new change while paraphrasing is making the sentence better or not. It can also help the user choose one suggestion from the available choices.

3.3 ALTERNATIVE IMPLEMENTATION USING NETSPEAK

Netspeak is a tool that is helpful when writing a text to make context-sensitive word choice (Riehmann et al. 2011). It helps with retrieving alternatives for a word or phrase using wildcard queries from the indexed corpus, which has more than 3 billion words in the form of *n-grams*, where $n \leq 5$. The occurrence frequencies of these *n-grams* are also stored with them (Riehmann et al. 2011).

Netspeak is built on a query processor tailored for the following task: consider a query with a wildcard q, and a set S of *n-grams*. The task is to retrieve the *n-grams* $S_q \subseteq S$ that matches the pattern defined in query q (Potthast et al. 2014). The corpus used to retrieve the query is stored as an inverted index based on hashing. The hash function is used to map the vocabulary V of the corpus to the storage positions. To be space optimal the hash functions are constructed using the CHD algorithm. The indexing provides a top-k retrieval strategy which finds n-grams that match the query.

Section 3.2.2 discussed the implementation of operators using neural models. We also develop the same operators using netspeak, to compare how the suggestions vary from both the approaches. To implement the operators using netspeak, the netspeak API is used.

The *Add Before, Add After* and *Replace* operator use the *question mark (?)*, which matches exactly one word, means the substitute for a question mark will

4 https://github.com/simonepri/lm-scorer

be only one word. The *Synonym* operator uses *hash sign (#)*, the hash sign before a word (#<*word*>), finds matches for the word's synonyms.

As known netspeak has a corpus in the form of *n-grams*, where $n \leq 5$, therefore, the query can only have a maximum of 5 words. When making an API call to the netspeak API, two words before and after the selected chunk are only sent as part of the input query. The query forms for the operators when using netspeak API are:

1. **Add Before**
   Sentence: *A human trial of a **coronavirus** vaccine has started*
   The user is using *add before* operator and the user wants to add a word before **coronavirus**. The query to get the suggestion to add word before coronavirus is:
   Query: *of a **?** coronavirus vaccine*

2. **Add After**
   Consider the following sentence:
   Sentence: *I and my friend are flying to **Malta** tomorrow morning*
   The user is using *add after* operator and the user wants to add a word after **Malta**. The query to get the suggestion to add word after Malta is:
   Query: *to Malta **?** tomorrow morning*

3. **Replace**
   Consider the following sentence:
   Sentence: *love this idea and **most definitely** want to help*
   The user is using *replace* operator and the user wants to replace **most definitely**. The query to get the suggestion to replace most definitely is:
   Query: *idea and **?** want to*

4. **Synonym** Consider the following sentence:
   Sentence: *That one nightmare single handedly **scared** me more than anything else in my entire life*
   The user is using *synonym* operator and the user wants synonym for **scared**.

The query to get the suggestion for synonym of scared is:
Query: *single handedly* ***#scared*** *me more*

The results from these operators are updated in the same way as explained in Section 3.2.2.

If the query does not return any result or the frequency of a suggestion is less than 1.5% or the count of the number of times a suggestion's presence in the corpus is less than 100 then we perform a *backoff* operation on the query. In the backoff operation, one word from either end of the query is removed and then a request to the netspeak API is made, this process is continued till there are results which are above the mentioned criteria. If there are the same number of words on either side of the wild card character, then first the word on the left end is removed and an API request is made, if the results do not match the criteria or there are no results then in the next iteration word on the right end is removed. If anyone of the side of wild card character has more number of words than other, then a word from that end is removed. Consider the following example of a query where backoff is performed. In this query, a substitute for the word *hard* is being requested.

Art is ***hard*** *to categorize*

#	**Query**
1	Art is **?** to categorize
2	is **?** to categorize
3	is **?** to

Table 2: Backoff query sequence

The first query does not give any result, therefore the word *Art* is removed from the query and a request is made again. The #2 query as mentioned in Table 2 gives a result which are above the mentioned criteria. So at this step, the backoff stops. If there are no results, or results do not match the criteria, then backoff would be done again and another request would be made to the netspeak API, and the word *categorize* would be removed and a request using query #3 would be made to the netspeak API.

4 USER STUDY OF INTERACTIVE PARAPHRASING TOOL

In this chapter, we discuss the specifics of the user study design and how we evaluate the obtained paraphrases. The user study aims to obtain feedback on the features and usability of the tool. The paraphrases as a result of the study are evaluated for their quality using ranking. In addition to the findings from the user study, the collection of paraphrases helps in building a paraphrasing corpus.

4.1 USER STUDY DESIGN

In this section, we describe the design of the user study. We start by describing the target users participating in the study, how the study is conducted, and how the different sentences are collected for the paraphrasing task. Then, we describe how the feedback is obtained, along with the feedback questionnaire. Lastly, we discuss the tool activity logging; these are activities from the tool when participants are performing the paraphrasing task.

4.1.1 *Target Users for Study*

The users who participate in the study are expert writers, fluent in English, and are categorized into two groups:

- Native speakers
- Non-native speakers

Native speakers are users having English as their first language. Non-native speakers are those whose first language is not English, but they are fluent in it. A total of 8 users participate in this study, 4 from each category. The expert writers who participate in the study are crowdsourced from Upwork[1]. They either have experience in English Writing for 10-20 years or have a certification or degree in Language Education. There are 5 female participants and 3

1 https://www.upwork.com/

male participants, and their age is in the range of 22 years to 65 years. The native English speakers are from Canada, United Kingdom, and United States of America. And non-native English speakers are from Bangladesh, Egypt, Japan, and Philippines. We pay them a decent fee of 100$ per participant for participating and completing the user study.

4.1.2 *Study Task*

The user study is divided into two models: **Autumn** and **Spring**. The user interface in both the models is the same, but the operators are built using two different approaches:

- **Autumn:** The operators of this model are developed using Netspeak (Section 3.3)
- **Spring:** The operators of this model are developed using Neural Models (Section 3.2.2)

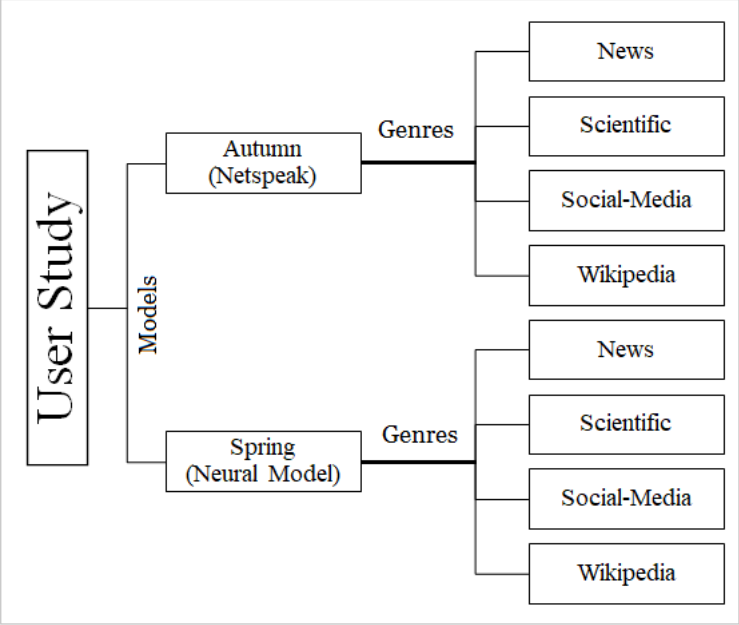


Figure 22: User Study Task

In each model, there are 48 sentences from four different genres, therefore together each user will paraphrase 96 sentences (48 x 2 models). To avoid bias in the feedback, 4 participants perform the paraphrasing for the autumn model first and then for the spring model, and vice-versa for the other 4 participants. In each model, the sentences are categorized into 4 genres (Figure 22), and

the users perform the paraphrasing of the sentences genre by genre. The Section 4.1.3 describes the genre and from where the sentences are chosen for paraphrasing. After completion of paraphrasing for each genre, the user gives feedback about the quality of suggestions for that genre. To avoid bias in the feedback after each genre, genre order for all the users are modified. The Table 3 gives the order of model and genre for each user.

#	User Type	Model 1	Model 2	Genre 1	Genre 2	Genre 3	Genre 4
1	Native	Autumn	Spring	News	Scientific	Social-Media	Wikipedia
2	Native	Autumn	Spring	Wikipedia	News	Scientific	Social-Media
3	Non-Native	Autumn	Spring	Social-Media	Wikipedia	News	Scientific
4	Non-Native	Autumn	Spring	Scientific	Social-Media	Wikipedia	News
5	Native	Spring	Autumn	Scientific	News	Social-Media	Wikipedia
6	Native	Spring	Autumn	Social-Media	Wikipedia	Scientific	News
7	Non-Native	Spring	Autumn	Wikipedia	Scientific	News	Social-Media
8	Non-Native	Spring	Autumn	News	Social-Media	Wikipedia	Scientific

Table 3: Order of model and genre for each user

Finally, after paraphrasing all the sentences from both the model's, user provide final feedback for the tool. Once, the feedback is submitted, the user study is completed. The details about the feedback questions are discussed in Section 4.1.4. Also, there is a fifth category called *Try*, it has 4 sentences from each genre. These sentences are for the user to get familiar with the tool and no data from these are saved. In the user study, the user do not have the option to try any sentence of their choice for paraphrasing or change the structure of sentence or add words manually, to accomplish this we disabled the input field. The input sentence are loaded from the database on page load and their chunks are also retrieved simultaneously.

In addition to the user study task, all the participants are asked their definition of *paraphrasing*. The definition helps to interpret and infer data and feedback from the user study.

4.1.3 *Sentence Collection for Paraphrasing*

The first step of the user study is collecting the sentences that the users will paraphrase using the interactive paraphrasing tool. These 48 sentences to be paraphrased are from the following four genres:

- News
- Scientific texts
- Social media
- Wikipedia

We have divided the sentences into different genres to distinguish for which type of sentence the tool provides better suggestions. The categorization helps us to understand how the suggestion varies according to the writing style of the sentence, also helps to find the quality and quantity of suggestions for the different genres.

In each model, there are 12 sentences from each genre. And in each genre, 6 sentences are common to both models (Autumn and Spring) and 6 are unique to each model. The reason to have 6 common sentences in both the models is to compare the output paraphrases from both the models and also get preference from the user about the suggestions from which model are better. The sentences for all the genres are from dataset as mentioned in Table 4.

Genre	**Dataset**
News	Multi-News (Fabbri et al. 2019)
Scientific	Scientific Papers (Cohan et al. 2018)
Social-Media	Reddit (Völske et al. 2017)
Wikipedia	Wiki-Split (Botha et al. 2018)

Table 4: Sentences of genre's are from the dataset/corpus

The sentences for *News* genre are randomly taken from *Multi-News* dataset (Fabbri et al. 2019). The Multi-News dataset consists of news articles and human-written summaries of these articles from the site *newser.com*. The sentences for *Scientific texts* genre are selected randomly from the *Scientific Papers* dataset obtained from open access *PubMed* repository (Cohan et al. 2018).

Sentences for *Social media* genre are chosen from the *Reddit* dataset consisting of posts from the Reddit, the largest discussion forum on the web (Völske et al. 2017). The sentences for *Wikipedia* genre are chosen randomly from *Wiki-Split* dataset (Botha et al. 2018). The *Wiki-Split* dataset is extracted from Wikipedia, it is constructed from publically available wikipedia revision history.

4.1.4 *Feedback Questionnaire*

The feedback questionnaires help measure user's satisfaction with the features and usability of the tool. These questions are formed using questions from subjective usability measurement at IBM as a reference (Lewis 1995). The feedback for the question is taken on the *likert scale*.

The feedback questions are divided into two groups, the first group of questions are answered by the user after each genre, and the second, after paraphrasing of all the sentences in both the models is completed.

Genre Feedback:

These feedback questions are answered separately for each genre in both the models after the genre's sentences are paraphrased. It has the following questions:

1. Rank the quality of suggestions for each operator:
    a) Add Previous/ Add Next
    b) Replace
    c) Synonym

    (Very Poor / Poor / Acceptable / Good / Very Good)
2. How often did you consider the feedback indicator for picking a suggestion for an operator? *(Never / Rarely / Sometimes / Very Often / Always)*

Tool Feedback:

These feedback questions are answered after paraphrasing sentences in both the models is completed, which is at the end of the user study. Its questions are:

1. How useful were the split and merge operators? *(Extremely / Very / Moderately / Slightly / Not at all)*
2. How useful was the edit history? *(Extremely / Very / Moderately / Slightly / Not at all)*
3. How was the learning curve for you to use the tool ? *(Very High / Above Average / Average / Below Average / Very Low)*
4. Any other feedback or comments:
    - What features would you add to / remove from the tool?
    - List a few positive/ negative points about the tool and its user experience
5. How much would you pay per month to use this tool?

In addition to the feedback, the interactions of the user with the tool are logged, which is discussed in the Section 4.1.5.

4.1.5 *Tool Activity Logging*

Activity logging helps us track the user's interactions with the interactive paraphrasing tool. Complete activity for each paraphrased sentence is logged. This logging is to get data about how the tool is used, which operators they used the most, what all features they used, how the usage of operators varied between genres and models and how much time users needed to paraphrase a sentence.

There are different activities logged to know usefulness of the tool features and to get evidence of the feedback that user gave. Following activities are logged:

- time to paraphrase the sentence
- operator usage count for add before/after, replace, synonym, merge, and split operator
- sequence of activities (e.g clicking on a operator/undo button). For each individual activity we log the following:
    - selected chunk
    - selected suggestion, if any

- time at which a chunk is selected
- time at which a suggestion is selected
- sentence score
- sentence before and after an operation is performed
- chunk list
- suggestion list from the operation performed
- is history shown or not shown

The activity data is useful for us in analyzing user interaction of the tool and it gives insights, if all the provided features are useful or not.

4.2 USER STUDY EVALUATION SETTINGS

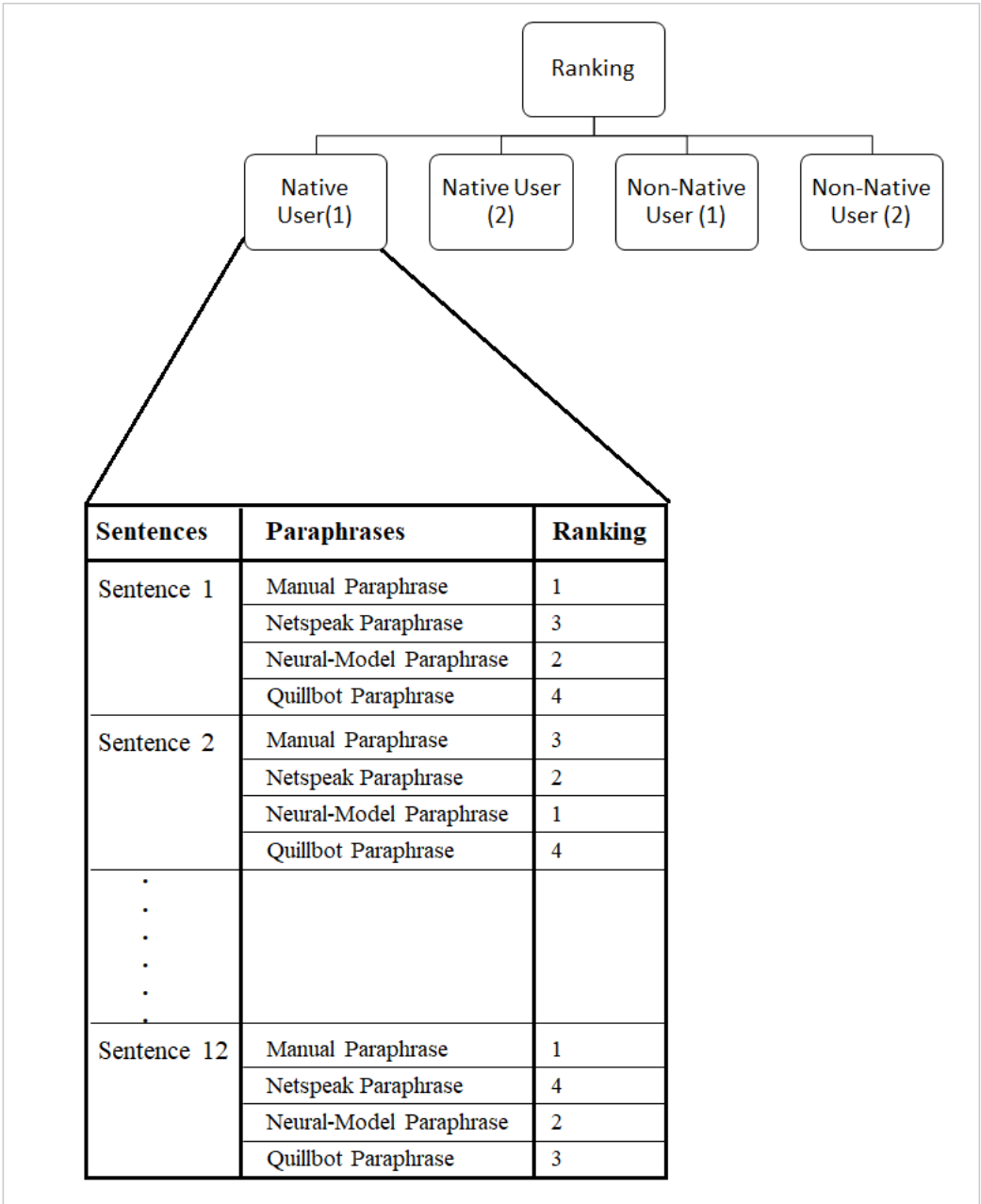
This section describes the setting in which the acquired paraphrases and feedback are analyzed and evaluated. The settings are mainly divided into the following categories:

A: Manual vs tool paraphrases comparison:

This analysis is performed to observe how differently humans paraphrase manually and using the tool. In the user study, the user performs paraphrasing in total on 96 sentences using tools. In order to perform manual and tool paraphrase comparison, 12 sentences are picked out of 96 sentences on which manual paraphrasing is performed by participants of user study separately before starting the task of paraphrasing using the tool. As previously stated each genre has 6 common sentences in both the models and the 12 sentences for manual paraphrasing contains 3 common sentences out of 6 common sentences in each genre which makes up 12 (3 sentences x 4 genres) sentences for manual paraphrasing. Once both the paraphrasing is completed, the paraphrases are observed for differences. This difference puts light on how people can think differently when paraphrasing manually and when using a tool.

B: Ranking Paraphrases:

Along with the manual vs tool paraphrase comparison, the paraphrases are also evaluated by ranking. Manual and tool paraphrases of 2 native and 2 non-native



Sentences	Paraphrases	Ranking
Sentence 1	Manual Paraphrase	1
	Netspeak Paraphrase	3
	Neural-Model Paraphrase	2
	Quillbot Paraphrase	4
Sentence 2	Manual Paraphrase	3
	Netspeak Paraphrase	2
	Neural-Model Paraphrase	1
	Quillbot Paraphrase	4
. . . . .		
Sentence 12	Manual Paraphrase	1
	Netspeak Paraphrase	4
	Neural-Model Paraphrase	2
	Quillbot Paraphrase	3

Figure 23: Ranking Paraphrases

users are given for ranking to an expert writer crowd sourced via *Upwork*. The paraphrases are ranked by the expert on the following criteria: (1) faithfulness of the paraphrase to the original sentence, (2) use of words is different from the original sentence, (3) difference in structure than the original sentence. In the ranking process 4 paraphrases of a sentences are ranked. Of the 4 paraphrases three are done by user, first manually, second using the autumn model (netspeak), third using the spring model (neural-model). The last paraphrase is

done using the tool *Quillbot*[2] by us, which is to compare the paraphrases from interactive paraphrasing tool (neural-model based) we developed with paraphrases from quillbot. Quillbot's free version is used for paraphrasing, which automatically paraphrases a given sentence. It also has additional suggestion which can be manually selected, but those are not used in this study. In all 48 (12 x 4 (2 native and 2 non-native user)) groups of sentences are ranked, and each group of a sentence has 4 paraphrases. Theses sentence were combined in a table and given to the expert for ranking, as in Figure 23.

C: Model wise comparison:

In this comparison, we compare the operator usage and it's feedback for the different genres and models. This helps us in analyzing the operator's performance for which genre(i.e the type of sentence) and model is better. It also gives insight about which operator is preferred most when paraphrasing.

D: Tool Review:

In the tool review, we present the user's feedback on the tool. And we also analyze the feedback from the user about the history button, split-merge operator, the learning curve of the tool, usefulness of feedback metric, and how much participants would like to pay monthly for the tool?

The results from the analysis and evaluation of user study are presented in the next chapter.

2 https://quillbot.com/

5 USER STUDY ANALYSIS

In this chapter we discuss the analysis of results from user study and results from evaluation of different paraphrases. First, we discuss the results from comparison of manual and tool paraphrases, then the results of ranking of different paraphrases are discussed. Then we analyse the activity data for count of operator usage and feedback for quality of suggestion for operators. At the end we discuss user's overall feedback of the tool.

5.1 PARAPHRASE DEFINITION AS PER USER

In this section we discuss the paraphrase definition according to the user. Before the user study all the users were asked, their definition of paraphrasing. Their meaning of paraphrasing were:

- Rewriting or changing structure of a sentence to maintain the same meaning but with different words or synonyms
- Rewording a sentence with an eye on shortening the length
- Removing unnecessary words and details, simplifying the text, making it shorter and snappier, reducing the word count so only the essential details remain

User's definition indicate that restructuring the sentence, with same or different words is their approach to paraphrasing mainly, the next thing that could be done is shortening the sentence. The tool, do not have a option to restructure the sentence, also none of the users mentioned about adding additional word to a sentence to enrich it. So, it is good to analyse how they used the tool without the option to rearrange the words and how they used the add operator, weather they use it or not?

5.2 MANUAL VS TOOL PARAPHRASING

In this section we describe the observation's from comparison of manual paraphrases and paraphrases from tool. The definition, along with the differ-

ences in manual and tool paraphrases help in analyzing how differently user can paraphrase using a tool with help of suggestions it provides.

There are different observations from the sentence paraphrased manually and using tool. Some example sentences paraphrased manually and using tool are listed below:

1. *Sentence:* Do Re Mi is a song by Kurt Cobain of the band Nirvana, and is *believed* to be one of the *final songs he wrote* before his *apparent* suicide in April 1994.
   *Manual Paraphrase:*Kurt Cobain of band Nirvana wrote the song Do Re Mi, and it was *thought* as one of his *last works* before he *committed* suicide in April 1994.
   Tool Paraphrase: Do Re Mi is a song by Kurt Cobain of the band Nirvana, and is *thought* to be one of the *final songs he composed* before his *reported* suicide in April 1994
2. *Sentence:* Her *mother worked as a seamstress* and her father was a German immigrant, and Autumn *says* that she did not share a close relationship with him.
   Manual Paraphrase: Autumn *claims* that she does not have close *ties* with her father, a German immigrant and *married* to a seamstress.
   Tool Paraphrase: Her *mom was a tailor* and her dad was from Germany. Autumn *states* that she did not have a close *connection* with him.
3. *Sentence: Further* randomized studies are *needed* to *confirm* these findings.
   Manual Paraphrase: We need to carry on further studies at random in order to *confirm* our findings.
   Tool Paraphrase: Additional randomized studies are *necessary* to *validate* these findings.

These paraphrases indicate following differences in a sentence paraphrased manually and using tool:

- In manual paraphrase always the structure of the sentence is changed i.e words are rearranged, but with the tool the structure is never changed, as the tool does not provide this functionality.
- In certain cases, the same terms as modified in manual paraphrases are updated with the **same** alternative/synonym.
- In certain cases, the same terms as modified in manual paraphrases are updated, but with the **different** alternative/synonym.
- In certain cases, words which are not changed in manual paraphrasing are updated with distinct alternatives.
- A longer sentence is split into two shorter sentences when using the tool. However, this is a rare occurrence observed in any of the manual paraphrases.
- Sometimes, additional adjectives/adverbs are added to a sentence when using the tool.

These observations indicate, given additional options when paraphrasing a sentence, the output paraphrases will always be improved. It also shows the different way of thinking when there are different suggestions available when paraphrasing a sentence.

5.3 RANKING PARAPHRASES

Ranking of paraphrases helps to know the quality of the paraphrases generated using our tool. The comparison of paraphrase also, helps to know where the netspeak[1]model and neural-paraphrasing[1] model from the interactive paraphrasing tool stands in comparison to Quillbot.

The Figure 24, shows the results from ranking. It clearly indicates that the manual paraphrases done by the expert users are best as 37/48 are ranked as first. From the three tools, the one developed by us, using neural-model ranks second, quillbot is ranked third and netspeak fourth. The ranking, confirms that the suggestions from the neural-model developed by us are good. As quillbot automatically paraphrases a sentence and also restructures a sentence in con-

1 In this chapter to avoid confusion, instead of autumn and spring model, we directly mention netspeak model and neural-paraphrasing model.

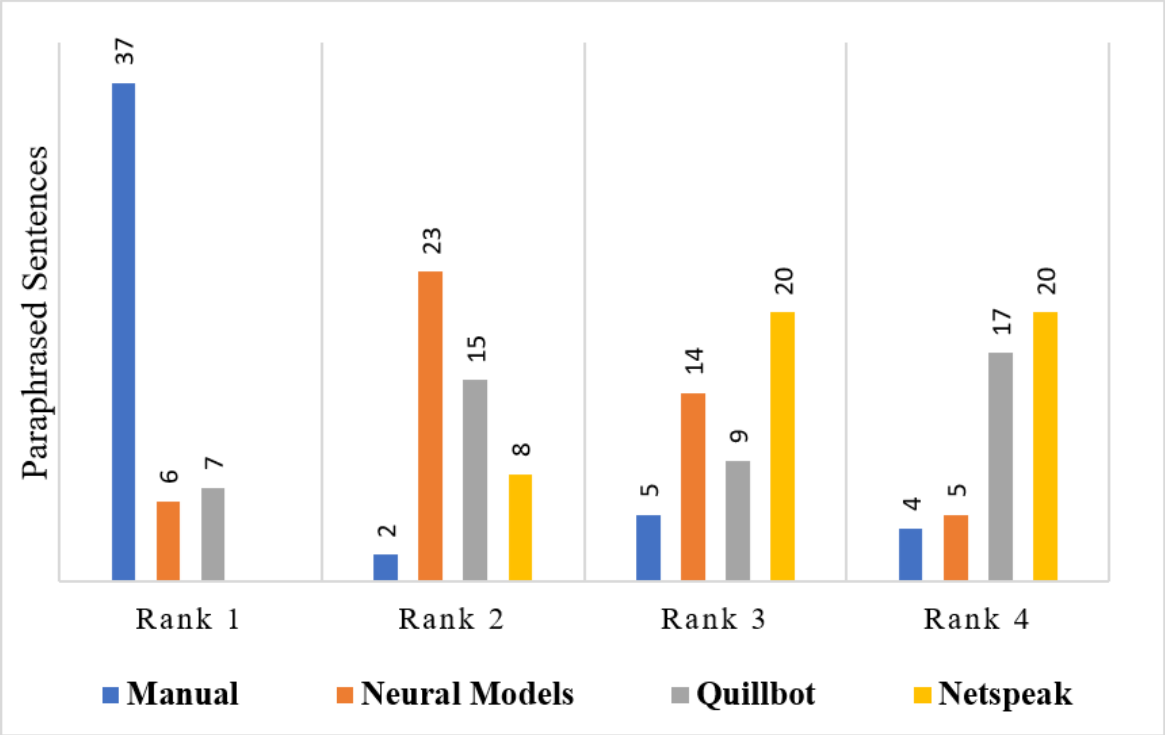


Figure 24: Paraphrases Ranking

trast to the neural-model, it's performance is not as good as the neural-model. The ranking also shows that paraphrases from neural-model and quillbot are much better than those from netspeak.

5.4 OPERATOR USAGE

This section discusses about the operator usage count and compares them between the different models and genres. The data in Table 5 shows the maximum, minimum and average usage count of each operator and the time spent on paraphrasing each sentence.

Table 5, indicates the average add operator usage for all the genres and both model is almost the same and is less than replace and synonym operator usage. This shows, that users did not try much to add new words to a sentence, they preferred to replace a word or use synonyms.

The average usage of add, replace and synonym operator in netspeak model is more compared to neural-paraphrasing model, the reason for this could be the users did not get the desired suggestion, therefore they tried the operator many times for different words or phrases. This can also be verified from the feedback

Genre	Operator	Netspeak			Neural-Paraphrasing		
		Max.	Avg.	Min.	Max.	Avg.	Min.
News	Add	21	4	0	13	3	0
	Replace	33	8	1	17	4	0
	Synonym	18	7	0	15	5	0
	Merge	10	2	0	25	2	0
	Split	28	10	2	21	8	1
	Time (sec)	3022	282	31	2043	250	54
Scientific	Add	27	5	0	13	3	0
	Replace	19	8	0	11	4	0
	Synonym	22	7	0	15	6	0
	Merge	6	1	0	5	1	0
	Split	22	9	0	19	10	2
	Time (sec)	794	233	23	2951	312	44
Social-Media	Add	30	4	0	13	3	0
	Replace	32	7	1	14	4	0
	Synonym	14	5	0	13	4	0
	Merge	22	2	0	7	1	0
	Split	27	8	1	21	8	1
	Time (sec)	879	169	20	864	194	20
Wikipedia	Add	21	3	0	26	4	0
	Replace	23	6	0	16	4	0
	Synonym	21	6	0	12	4	0
	Merge	6	1	0	12	1	0
	Split	24	8	1	25	9	2
	Time (sec)	908	174	33	1059	257	54

Table 5: Operator Usage

the users gave about the suggestions from each model as seen in Figure 25. There is only one exception about the add operator usage for the Wikipedia genre, its usage count is greater for the neural-paraphrasing model as compared to the netspeak model and as the difference is small it is difficult to conclude the quality of suggestions for it.

In the netspeak model, the most used operator is replace, followed by synonym and add. In neural-paraphrasing model, synonym and replace are used almost the same number of time, and add is used comparatively less.

We assumed that users would try combining different words/chunks and apply the add/replace/synonym operators on them, but average merge operator count which is 1 or 2 shows that the users did not try different combinations of words/chunks to get suggestions.

The most used operator is split as it is needed to get synonym for words, as for synonym operator to work a chunk can have only one word. This also, supports that synonym operator was used the most or the replace operator was mostly used on a chunk which had only one word.

The time range for paraphrasing each sentence was 2-5 minutes. This shows the efficiency of tool with respect to time is good. Except for the News genre, the average time spent spent in paraphrasing a sentence in neural-paraphrasing model is more compared to netspeak model, it could be because neural-paraphrasing model gave better suggestions as compared to netspeak model, and users took some time in deciding which suggestion to select.

5.5 OPERATOR EVALUATION

The feedback about the operator was collected to know the quality of suggestions for each genre in both the models. The feedback for each model (netspeak/ neural-paraphrasing) is plotted separately for comparison.

The reason to collect feedback for each genre separately was to know how the quality of suggestions varies for different types of sentences. The Figure 25 compares the feedbacks for Add Before/After, Replace, and Synonym Operator from Netspeak and Neural-Paraphrasing Models for each genre. The feedback makes it clear that the suggestions from the neural models are better than those from netspeak. When comparing each operators performance in each genre for both the models, there is not much difference, the feedback is almost the same.

5.6 TOOL REVIEW

In addition to feedback about the quality of suggestions from operators, users were also asked about usefulness of feedback metric, history button, split/merge

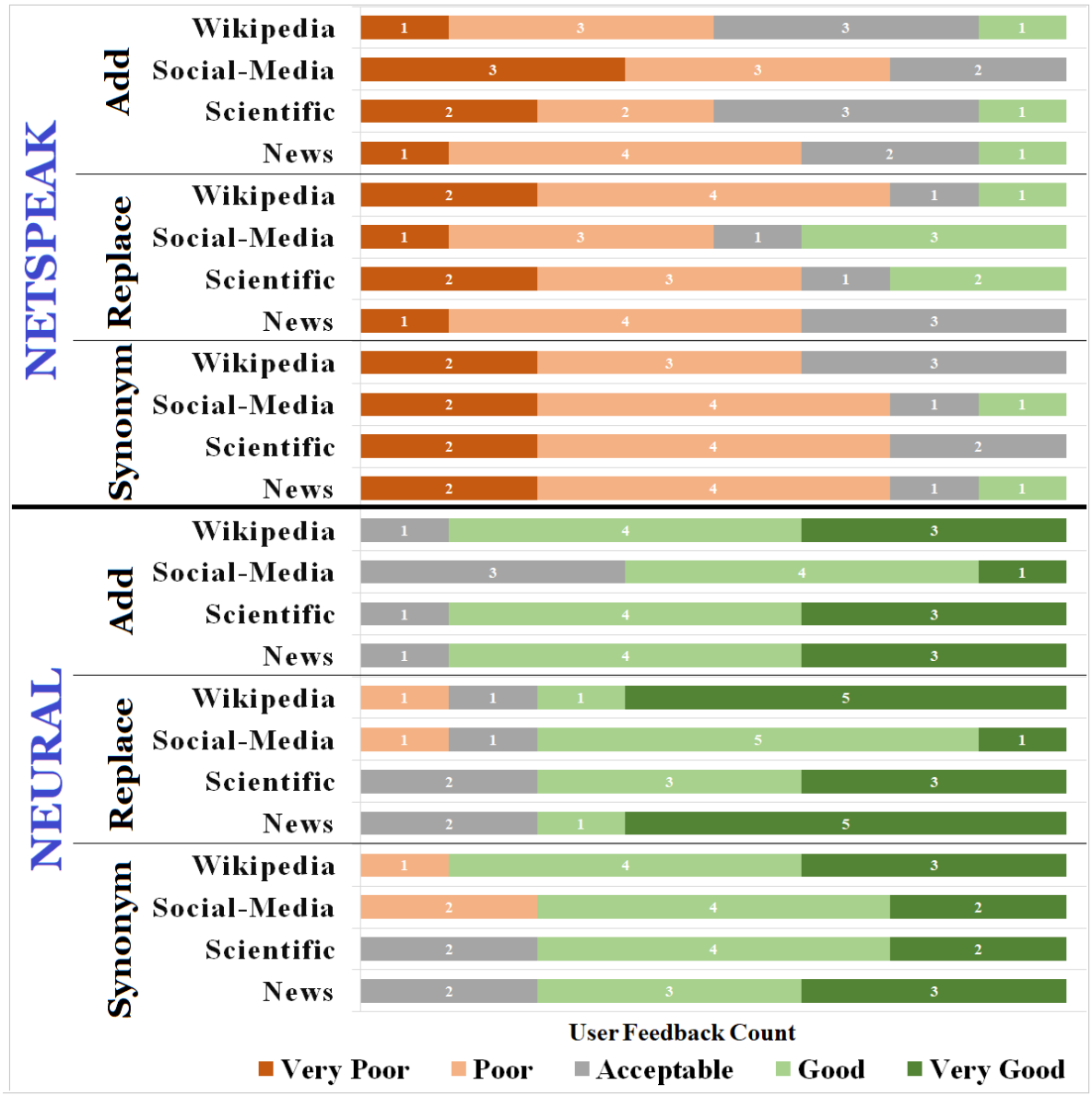


Figure 25: Feedback on Operator categorised by genre

buttons, the learning curve of the tool and a review. In this section, the overall review of the tool is discussed. The Figure 26 shows the usefulness for history button and split-merge operator. The chart indicates that history button was used moderately, and the split-merge was very useful and can be considered a useful feature to be used in the tool. The Figure 27 shows the learning curve of the tool, and it is in the range of above average and average, which means the tool functionality was not that difficult to understand and learn. The Figure 28, shows that the feedback metric which indicates the score of the sentence as the paraphrasing progressed, was not useful. It means, another approach needs to

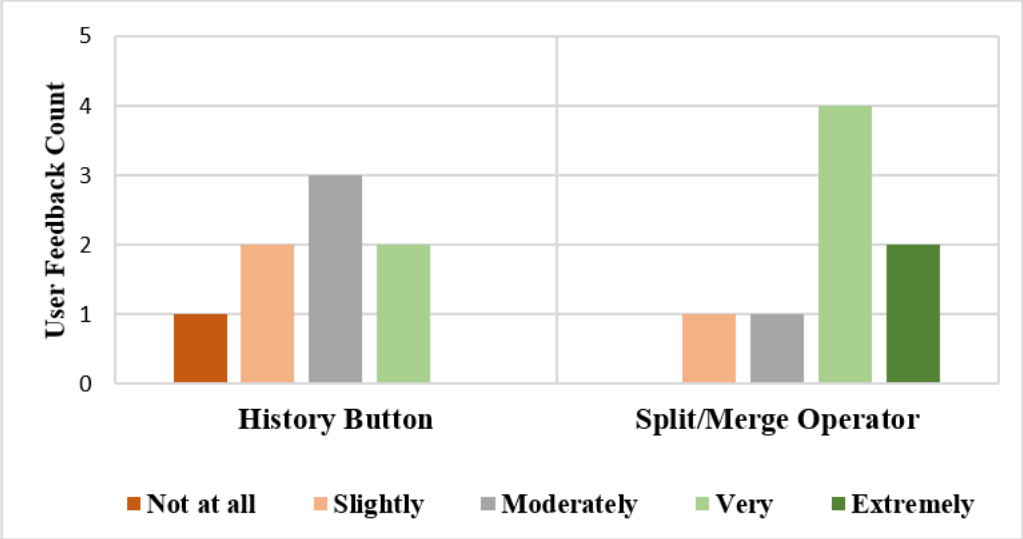


Figure 26: Usefulness of history button, split-merge functions

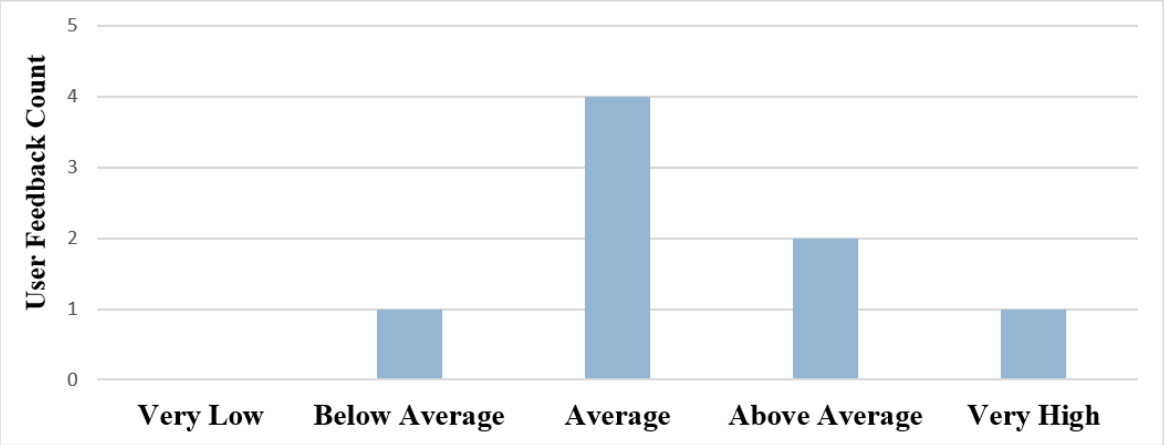


Figure 27: Learning Curve of the tool

be used to score the sentence, which would be more intuitive and useful for the users. The usefulness of the feedback metric was asked to each user after paraphrasing sentences in each genre, in total there are 64 (8 users x 8 (2 models x 4 genres)) feedback for it.

On completing the user study, along with the usefulness of different features, the users were asked to give a review of the tool, there were both positive and negative reviews. Following are the reviews from participants of user study:

- The neural-model based tool is better than netspeak one.
- A functionality to restructure the sentence is essential.

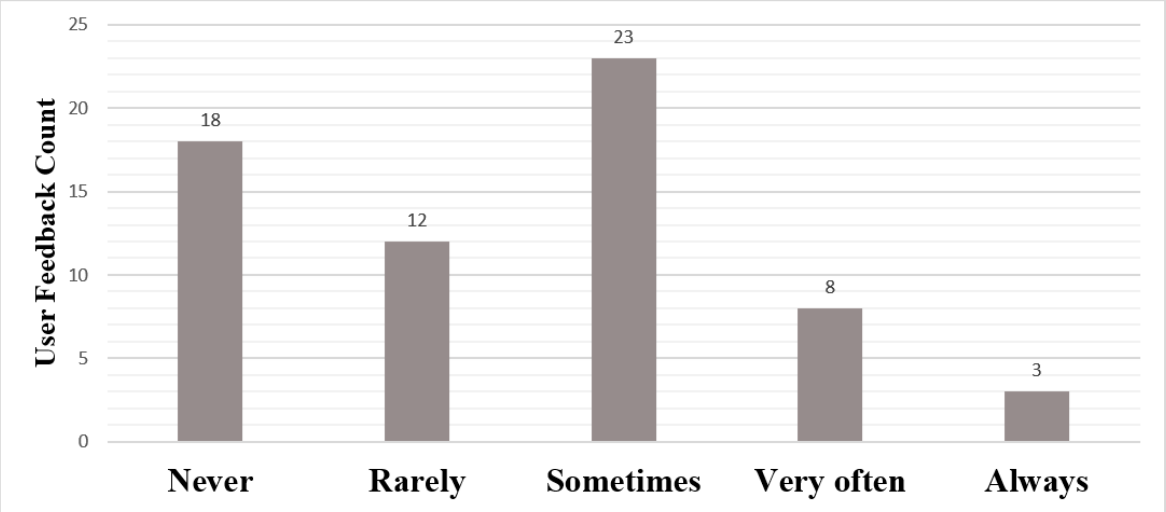


Figure 28: Use of feedback metric

- The synonym operator in the neural-model needs improvement, as they are sometimes not accurate.
- There should be grammar check functionality as the paraphrasing is progressing.
- A functionality to get synonym for a phrase would be beneficial, along with the current one.

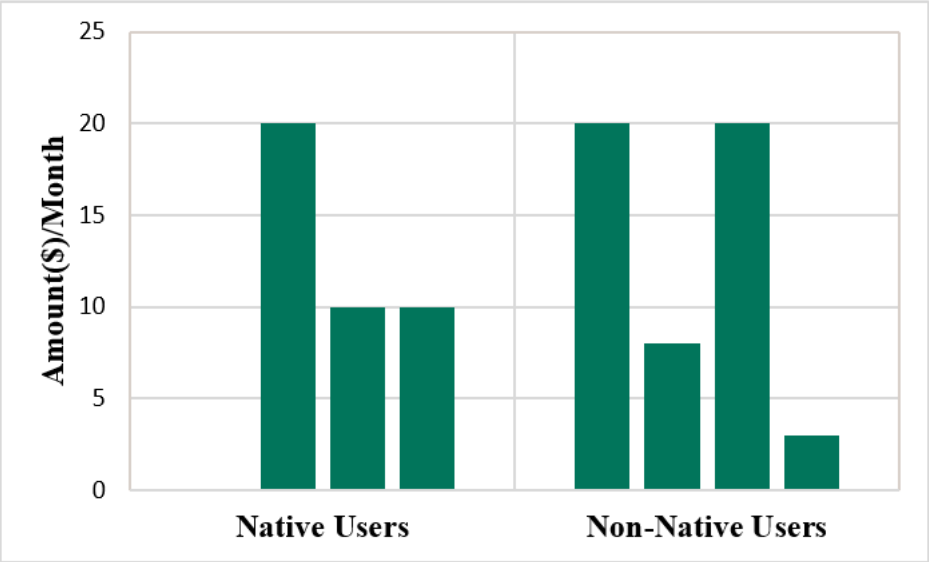


Figure 29: Monthly payment ($) for tool

The Figure 29 show the distribution of the amount the user would pay monthly for using the tool. Only, one native user said that she would pay 0$ to use the

tool, as she felt there is some more functionality and improvement needed in the tool. Except for one native user, everyone else quoted an amount. Some of the participants have quoted a amount less than 10$, the reason for this could be the country they live in. We feel 10$ is a good amount for using the tool and this shows that if participants are ready to pay, the tool gives good suggestions.

EFFICIENCY: There are different parameters, which validate the efficiency of the tool. The first and the most important for us was the time taken to get suggestions. As seen from the activity logs, the time to get suggestions is in the range of 3-5 seconds for the Add Before/After and Replace operator and 8-10 seconds for synonym operator. We feel that it is quick, but can be made better. In addition to time, the neural-paraphrasing tool helps in fulfilling the task of paraphrasing by providing good suggestions, which is proven by the feedback. The average learning curve also shows that it is not that difficult to use the tool, although there is a scope for improvement. Using the results from the evaluation of paraphrases and analysis of feedback and reviews, we come to know the tool fulfills its purpose and therefore we can say that it is efficient and effective.

6 CONCLUSION

The aim of this research was to develop a *prototype interactive paraphrasing tool* that gives users more access and control over the paraphrasing process and which is developed using neural models. Once the tool is developed, a user study is performed to get a feedback from user about the tool and it's features. Our work is divided into two parts: tool design and implementation and user study design and evaluation

TOOL DESIGN AND IMPLEMENTATION: In designing the tool we tried to keep the tool simple, easy to use, and understand. This was accomplished by appropriately positioning elements (i.e. chunks, operators, functionality buttons) and using color to distinguish between them.

In implementing the tool, the first step was to divide a sentence into small phrases called chunks, we use the FLAIR framework (Akbik, Bergmann, et al. 2019) to generate them. Once the chunks are generated, the users control the further paraphrasing process with the help of neural based paraphrasing operators. *Add Before/After* and *Replace* operator are built using the *fill-mask* function from the *RoBERTa* model (Y. Liu et al. 2019). The *Synonym* operator is built using Masked Language Model (MLM) from BERT and lexical simplification technique (Qiang et al. 2020). Besides, the neural implementation, we also implement an alternative implementation using *Netspeak*. This implementation is used in the user study to compare the performance of the neural-paraphrasing model and netspeak model.

The chunks are a good way to divide a sentence into small phrases to work with, and they are a useful starting point in the interactive paraphrasing process. Based on the user study we can infer that the *Add Before/After* and *Replace* operators suggestions are good, relevant and are generated quickly. The

suggestions from the *Synonym* operator are not always as expected and it also takes a bit longer to generate them.

USER STUDY DESIGN AND EVALUATION: User study is performed for the evaluation of the efficiency and effectiveness of the tool. We evaluate the data, feedback, and paraphrases from the user study in different forms. The results from the comparison of manual paraphrases and those from the tool show that when tool provides quality suggestions users use them which helps to generate better paraphrases. It also demonstrates the different approaches of human thinking when there are different suggestions available for paraphrasing a sentence. Next, we evaluated the ranking of sentences by an expert writer. The results from ranking indicate that we can consider the tool developed by us on par with the one already in the market - Quillbot. It also confirms the quality of suggestions the neural model generates are effective. Then we evaluated the operator usage and feedback from the user. The evaluation shows the most used operator is *Replace*, followed by *Synonym* and the least used is *Add*. This compliments users definition of paraphrasing; *paraphrasing is re-arranging words in a sentence or replacing words with alternatives or synonyms.* The operator feedback indicates the suggestions from the neural-model are better than those from the netspeak model. The overall feedback from the user study reflects the performance of the neural model is substantially better than the netspeak model. Along with the operators, users point to the significance of providing a feature to rearrange words in a sentence.

6.1 IMPROVEMENTS AND FUTURE WORK

The interactive paraphrasing tool was developed as a prototype, and the suggestion it generates are effective. We can incorporate additional features to improve the productivity of the tool.

First, the operator we need to improve the most is the synonym operator. Instead of using neural models, we can use thesaurus or wordnet to generate synonyms or a combination of wordnet/thesaurus and neural-models. Along,

with synonyms for the word if we can generate synonyms/alternatives for a phrase that would be an additional feature for the tool.

Second, we need to add functionality for rearranging a sentence. It would also be beneficial if we can automatically generate re-ordered examples of sentences and suggest them to the user. This is a bit tricky, but if accomplished it would be best for our tool. If automatic re-ordering is not possible, we can give drag-and-drop functionality in the interface, using which users can move the chunk and re-order the sentence.

Third, we need a feature that corrects the sentence grammatically. If we can fetch grammatically correct suggestions that would be even better. To achieve this once we get the suggestions from MLM, we correct them grammatically so they fit correctly in place of the masked token and then send them to the interactive interface.

Fourth, using the activity logs, we can train a model, that suggests which operator the user should use on which chunk to generate good paraphrases according to the input sentence. Currently, the user has to try different operators on different chunks, by giving them a clue of which operator to use, they can paraphrase more quickly.

These are a few suggestions to improve the tool, but we are not restricted to this. We can think creatively and incorporate functionality that can differentiate the tool from others in the market.

A APPENDIX I

The input sentence that were used in the user study for performing paraphrasing task are listed in this appendix.

A.1 GENRE: NEWS

Sentences common in both models:

1. Crews had the fire under control within an hour and were searching for anyone who may have been in the homes.
2. Employers pulled back sharply on hiring last month, a reminder that the American economy may not be growing fast enough to sustain robust job growth
3. Investigators could be on the scene for three to seven days for what they call the fact finding phase.
4. On Monday night, while the rest of the world was watching Charlie Sheen flame out live on CNN, Tucker Carlson took to Twitter to make some impolitic statements of his own.
5. Television news footage of the scene showed one home nearly destroyed, with a car in the driveway.
6. The investigations into a plane crash that left six people dead in Gaithersburg on Monday evening are just beginning.

Sentences from Autumn(Netspeak) Model:

1. A monster tornado hit Moore Okla Monday afternoon, leaving scores dead as the threat for more storms continues.
2. A simple dispute as to the luggage cannot possible be grounds for recusal.
3. Calls to suicide hotlines have spiked dramatically since the deaths of Kate Spade and Anthony Bourdain not an unusual phenomenon in the wake of celebrity suicides.

4. Severe threat continues farther to the east Wednesday, although the overall severity appears to be lower.
5. Strong wind gusts have been reported in the area but so far no tornadoes.
6. The Dallas zoo closed Tuesday afternoon due to the forecast

Sentences from Spring(Neural-Paraphrasing) Model:

1. A small, private jet has crashed into a house in Marylands Montgomery County on Monday, killing at least three people on board, authorities said.
2. But the Qataris were said to be upset when a number of decisions went against them.
3. He deleted the original tweet obviously aware that what he had posted was wrong.
4. The black box, which has recordings from the crash, has been recovered.
5. They'll be conducting interviews and documenting the wreckage.
6. Weather Channel meteorologist Kevin Roth said early Tuesday that the threat area appeared to be east and south of Oklahoma City.

A.2 GENRE: SCIENTIFIC

Sentences common in both models:

1. Further randomized studies are needed to confirm these findings.
2. Intravenous iron treatment alone is safe and may reduce blood transfusion requirements and improve hemoglobin level in patients with cancer who are undergoing anticancer therapy.
3. The effect of blood transfusion is often temporary and may be associated with serious adverse events.
4. The present study was carried out to assess the effects of community nutrition intervention based on advocacy approach on malnutrition status among school aged children in shiraz, iran.
5. These insects have negative ecological and economic impacts since they lower crop yield, and pesticides are expensive and can have off target effects on beneficial arthropods.

6. This was a cross sectional study that investigated pesticide exposure and its risk factors targeting vegetable farmers selected through cluster sampling.

Sentences from Autumn(Netspeak) Model:

1. Before starting ovulation induction and after oocyte harvesting, the general health questionnaire was filled by women who were under treatment.
2. Encouraging adolescents to control their weight in healthy ways is imperative.
3. Mental fatigue is for many a distressing and long term problem after stroke.
4. Suicide is a leading cause of death among adolescents globally , and body weight is also a recognized reason for adolescent suicide.
5. The process of assisted reproductive treatment is a stressful situation in the treatment of infertile couples and it would harm the mental health of women.
6. Therefore , we investigated the association between weight control behaviors and suicide ideation and attempt , focusing on inappropriate weight control measures.

Sentences from Spring(Neural-Paraphrasing) Model:

1. Anemia in patients with cancer who are undergoing active therapy is commonly encountered and may worsen quality of life in these patients.
2. Community nutrition intervention based on the advocacy process model is effective on reducing the prevalence of underweight specifically among female school aged children.
3. The data can be used for the formulation of an integrated program on safety and health in the vegetable industry.
4. The intention with this study is to investigate mental fatigue in relation to depression and cognitive functions.
5. The project provided nutritious snacks in public schools over a 2 year period along with advocacy oriented actions in order to implement and promote nutritional intervention.

6. We collected venous blood samples under fasting conditions, calculated bmi by height and weight, and assessed relevant biochemical factors.

A.3 GENRE: SOCIAL MEDIAL

Sentences common in both models:

1. Art is about the hardest thing to categorize in terms of good and bad.
2. Ask me what I think about the Wall Street Journal and I will tell you about it is bland, monumental, walls of text.
3. Hitting the moves on the beat is the basics of not only breakdancing, but all dancing.
4. It is fine to work on one or two volunteer projects for fun and to build interest and community but somewhere down the line it must be made sustainable.
5. Now the valley is dried up but has some of the best soil for crops in the world.
6. This is a fact.

Sentences from Autumn(Netspeak) Model:

1. I am simply asking you to present evidence or a reasonable argument.
2. I appreciate our level of conversation but i think you can understand if i tell you this is getting annoying.
3. I personally enjoy the work of street artists but I do see where you are coming from.
4. Like I said, I do not know and do not have the expertise or education.
5. Look, if you dont have a reasonable argument to back it up, we can move on.
6. You are perfectly entitled to a point of view without evidence or reasonable argument.

Sentences from Spring(Neural-Paraphrasing) Model:

1. That one nightmare single handedly scared me more than anything else in my entire life.

2. The lake should be pretty empty right now.
3. To consider one work or artist as dominate over another comes down to personal opinion.
4. To say otherwise would be an insult to the blood and sweat poured into this performance by both crews.
5. To simply say that The Massive Monkees were better dancers is a bit of an overstatement.
6. Yeah, but most folks think avoiding gluten will cause them to become thin, thats why I call it a fad.

A.4 GENRE: WIKIPEDIA

Sentences common in both models:

1. An essay written in 1848, included a cosmological theory that presaged the Big Bang theory by 80 years, as well as the first plausible solution to Olbers paradox.
2. Do Re Mi is a song by Kurt Cobain of the band Nirvana, and is believed to be one of the final songs he wrote before his apparent suicide in April 1994
3. Her mother worked as a seamstress and her father was a German immigrant, and Autumn says that she did not share a close relationship with him.
4. He was the fourth of the nine children of Jeremiah Scotland Allison and his wife Mariah Ruth Brown, and his father was a Presbyterian minister, who raised cattle and sheep to support his family.
5. Its Armenian population specialized in the production of handicraft and sericulture and in the sixteenth century, numerous sources spoke of it as a thriving town that maintained strong commercial links with India, Russia, Safavid Persia and Western Europe.
6. Wilson commenced recording a new solo album with Thomas, who purposely took it upon himself to ensure that the new work would sound as close to adult contemporary radio as possible.

Sentences from Autumn(Netspeak) Model:

1. My Creation, is a song by the thrash metal band Megadeth, was written by Dave Mustaine and Nick Menza.
2. One who prostrates himself on the mountain are Japanese mountain ascetic hermits who according to a traditional Japanese mysticism are believed to be endowed with supernatural powers.
3. Over the mountains Mary goes is a sacred motet by the Renaissance composer and musician Johannes Eccard, who wrote it on a German text by Ludwig Helmbold in two stanzas.
4. Sound you songs is a festive concerto, with words and music possibly based on an earlier lost secular.
5. Spanish for The Vanguard is a Spanish daily newspaper printed in Spanish and since 3 May 2011 also in Catalan.
6. The Puritans, is an opera in three acts by Vincenzo Bellini, which he wrote for the Theatre Italian in Paris and which was first presented on 24 January 1835.

Sentences from Spring(Neural-Paraphrasing) Model:

1. Prior to the release of the song, Omarion released snippets, the song was later premiered on November 11, 2014 by the LA Leakers.
2. She has an older brother, and from her father's second marriage to McKenzie, she has a younger half sister, Courtney Taylor and half brother, Jake.
3. The 26 season, has completed airing, and the series has been renewed for the 2013 television season
4. The English equivalent of azure is not considered to be a shade of blue but rather the opposite that is blue is a darker shade of azzurro.
5. The Lover, is a suite by Jean Sibelius, composed in 1911 for string orchestra, percussion and triangle.
6. The Traditional Values Coalition used the article to urge the Centers for Disease Control to cut down on its AIDS funding

BIBLIOGRAPHY

Akbik, Alan, Tanja Bergmann, et al. (June 2019). "FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 54–59. URL: https://www.aclweb.org/anthology/N19-4010.

Akbik, Alan, Duncan Blythe, et al. (Aug. 2018). "Contextual String Embeddings for Sequence Labeling." In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, pp. 1638–1649. URL: https://www.aclweb.org/anthology/C18-1139.

Botha, Jan A et al. (2018). "Learning To Split and Rephrase From Wikipedia Edit History." In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. arXiv preprint arXiv:1808.09468, to appear.

Burrows, Steven et al. (July 2013). "Paraphrase Acquisition via Crowdsourcing and Machine Learning." In: *ACM Trans. Intell. Syst. Technol.* 4.3. ISSN: 2157-6904. URL: https://doi.org/10.1145/2483669.2483676.

Cohan, Arman et al. (2018). "A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents." In: *CoRR* abs/1804.05685. arXiv: 1804.05685. URL: http://arxiv.org/abs/1804.05685.

Devlin, Jacob et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *CoRR* abs/1810.04805. arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

Fabbri, Alexander R. et al. (2019). "Multi-News: a Large-Scale Multi-Document Summarization Dataset and Abstractive Hierarchical Model." In: *CoRR* abs/1906.01749. arXiv: 1906.01749. URL: http://arxiv.org/abs/1906.01749.

Ledo, David et al. (2018). "Evaluation Strategies for HCI Toolkit Research." In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: Association for Computing Machinery, pp. 1–17. ISBN: 9781450356206. URL: https://doi.org/10.1145/3173574.3173610.

Lewis, James R. (1995). "IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use." In: *International Journal of Human-Computer Interaction*, pp. 57–78.

Li, Zichao, Xin Jiang, Lifeng Shang, and Hang Li (Oct. 2018). "Paraphrase Generation with Deep Reinforcement Learning." In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 3865–3878. URL: https://www.aclweb.org/anthology/D18-1421.

Li, Zichao, Xin Jiang, Lifeng Shang, and Qun Liu (July 2019). "Decomposable Neural Paraphrase Generation." In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 3403–3414. URL: https://www.aclweb.org/anthology/P19-1332.

Liu, Yinhan et al. (2019). "RoBERTa: A Robustly Optimized BERT Pretraining Approach." In: *CoRR* abs/1907.11692. arXiv: 1907.11692. URL: http://arxiv.org/abs/1907.11692.

Madnani, Nitin et al. (2010). "Generating Phrasal and Sentential Paraphrases: A Survey of Data-Driven Methods." In: *Computational Linguistics* 36.3, pp. 341–387. URL: https://www.aclweb.org/anthology/J10-3003.

Mallinson, Jonathan et al. (Apr. 2017). "Paraphrasing Revisited with Neural Machine Translation." In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 881–893. URL: https://www.aclweb.org/anthology/E17-1083.

Morville, Peter (June 2004). "User Experience Honeycomb." In: URL: http://semanticstudios.com/user_experience_design/.

Potthast, Martin et al. (Aug. 2014). "Improving Cloze Test Performance of Language Learners Using Web N-Grams." In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, pp. 962–973. URL: https://www.aclweb.org/anthology/C14-1091.

Prakash, Aaditya et al. (Dec. 2016). "Neural Paraphrase Generation with Stacked Residual LSTM Networks." In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, pp. 2923–2934. URL: https://www.aclweb.org/anthology/C16-1275.

Qiang, Jipeng et al. (2020). *LSBert: A Simple Framework for Lexical Simplification*. arXiv: 2006.14939 [cs.CL].

Riehmann, P. et al. (2011). "The NETSPEAK WORDGRAPH: Visualizing keywords in context." In: *2011 IEEE Pacific Visualization Symposium*, pp. 123–130.

Völske, Michael et al. (Sept. 2017). "TL;DR: Mining Reddit to Learn Automatic Summarization." In: *Proceedings of the Workshop on New Frontiers in Summarization*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 59–63. URL: https://www.aclweb.org/anthology/W17-4508.

Wolf, Thomas et al. (2019). "HuggingFace's Transformers: State-of-the-art Natural Language Processing." In: *ArXiv* abs/1910.03771.

Zhao, Shiqi, Xiang Lan, et al. (Aug. 2009). "Application-driven Statistical Paraphrase Generation." In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Association for Computational Linguistics, pp. 834–842. URL: https://www.aclweb.org/anthology/P09-1094.

Zhao, Shiqi and Haifeng Wang (Aug. 2010). “Paraphrases and Applications.” In: *Coling 2010: Paraphrases and Applications–Tutorial notes*. Beijing, China: Coling 2010 Organizing Committee, pp. 1–87. URL: https://www.aclweb.org/anthology/C10-4001.