# ReHash: An AI-powered Rewrite Tool

Team: Abigail Kufeldt and Meenal Chavan

Project Mentor Instructor: Delip Rao

Project Mentor TA: Brendan King

March 24, 2023

**Abstract**

For this project, we propose and build ReHash, a rewriting tool with the following features: simplicity style transfer, paraphrasing, and bulleted list conversion – each of which is backed by a finetuned large language model. The final product of this project is a fully functioning website constructed on the Streamlit API, open-source and free to use by the public. We also contribute a small, few-shot finetuning dataset (n=15) for the bulleted list to paragraph conversion task. Finally, we evaluate each of our model's performance against competitive baselines to ensure they are functioning well on their respective tasks and report those results here.

## 1 Introduction

### 1.1 Motivation

In today's fast-paced world, effective written communication is more important than ever. Even so, writing can be a time-consuming and tedious job, especially when it comes to tasks like drafting emails, converting bullet points to coherent paragraphs, or writing the first pass of an essay. Understanding complex writing, like textbooks or legal documents, can also be time-consuming and confusing – a challenge faced by students, professionals, and non-native English speakers alike. Because of this need, we are excited to develop a tool that can alleviate these challenges for a wide range of users.

We were inspired by existing websites like Grammarly, Quillbot, and Wordtune, which offer similar features. However, many of these tools come with a price tag, making it difficult for students and professionals on a budget to access them. With this project, we aim to create an open-source, AI-backed tool that will help individuals at every stage of their writing process, from generating paragraphs from bullet points to brainstorming different ways of writing an email. By providing a free, intuitive, and all-in-one tool, we hope to make writing a more manageable and efficient task for everyone.

Lastly, because bulleted list conversion is an understudied task, we hope that our development of a new dataset will help inspire more research in this area.

### 1.2 Problem Setup

We propose to develop a tool that can rewrite a user's input text in three possible directions: simplicity style transfer, bulleted list to paragraph conversion, and paraphrasing, each of which are accomplished using a neural network finetuned on the given task. We plan to use T5-small as the foundation of each of these three features, primarily because we're interested in gaining experience working with and finetuning this particular model – even though other existing LLMs such as GPT-3 may be more successful at each of these tasks. Note that because bulleted list conversion is an underresearched task, we will also need to generate a dataset to finetune our T5 model on.

### 1.3 Ethical Considerations

Although we do not expect any large societal impacts from this work, we do acknowledge the effect such a convenient and open source tool will have on plagiarism in education. While intended to help professionals draft their writing, naturally it's not difficult to imagine how students may abuse such a tool to avoid practicing their own writing skills.This is an issue we're aware of, but have no clear way to mitigate; we cannot monitor how users of our tool are breaking school guidelines, and thus have no real control over whether or not ReHash contributes to plagiarism.

## 2 Background

To start with our project we reviewed several papers that focused on simpilicity style transfer One of the papers that helped us pick our methodology was a survey paper : Data-Driven Sentence Simplification: Survey and Benchmark. The study focuses on several techniques that can be used to perform simplification of the language of a sentence. One of the tasks that was interesting in this one was how summarization and simplification seem similar but have different objectives. Shardlow (2014) points out, summarisation focuses on reducing length and content by removing unimportant or redundant information. In simplification, some deletion of content can also be performed, also involving replacement of words by more explanatory phrases, making co-references explicit, adding connectors to improve fluency, etc and thus resulting in longer sentences.The paper also talks about various other tasks such as abstractive sentence compression, split-and- rephrase that are sued for simplicity style transfer.

The second task of our project is converting bullet lists to paragraph, which can be viewed as a reverse summarization problem. Though it sounds highly fascinating and something people would use in a day to day life, we found little to no prior research in this area . A similar problem statement which was an NLG task of generating sentences from structed data was the closest kind of problem statement we came across for this task. It uses WebNLG challenge dataset which consists of sets of triplets describing facts (entities andhttps://www.overleaf.com/project/641a46762dff4429f5db5d22 relations between them) and the corresponding facts in form of natural language. Though this was a similar task, it didn't quite work for our project and we focused on prompting-based techniques for our task.

Our last task, paraphrasing, is a widely researched area in the field of NLP. We mainly referred to Paraphrase Generation: A Survey of the State of the Art, for an in-depth study on past research in the topic. It talked about various deep learning tasks which utilized several datasets such as WikiAnswer, MSCOCO, PPDB, Quora, TwitterURL to name a few. Jianing Zhou and Suma Bhat also covered how traditional rule-based, therasus-based, and SMT-based approaches were different from neural methods in paraphrasing tasks. We realised how neural network based approaches gave state-of-art results on quora and MSCOCO dataset which motivated us to use this dataset while fine-tuing T5.

## 3 Summary of Contributions

1. Contribution(s) in Application/Dataset: We build a fully functioning, free, and open-source website capable of each of the three features proposed in this assignment (simplicity, list conversion, and paraphrasing). We also produce a small (n=15) few-shot finetuning dataset for list conversion.

2. Contribution(s) in Algorithm: N/A

3. Contribution(s) in Analysis: We few-shot finetune T5-small on the understudied list conversion task and compare its performance against a model finetuned for paraphrasing. In doing so, we demonstrate that further work is necessary in this domain to better understand the best approach for this task.

## 4 Detailed Description of Contributions

### 4.1 Methods

#### 4.1.1 Simplicity Style Transfer

In this task, we finetune T5-small from HuggingFace on the ver. 2.0 sentence-aligned Simple English Wikipedia dataset for text simplification, as recommended by Professor Delip Rao[7]. PWKP is a common dataset for this task, as it consists of sentences from English Wikipedia paired with their Simple English Wikipedia counterparts. This set in particular consists of 167,000 aligned sentence pairs.

Some sentence pairs consisted of the same sentence as the input and the target, which would be an issue for our model; we don't want our model to learn to output the same sentence. So, in terms of preprocessing, we removed all such pairs.

Some sentence pairs were marginally different, and its important to note that none of the input sentences in this dataset were exceedingly complex. In other words, the dataset is not optimum for our use case as we want users to be able to simplify very complex pieces of text into something simple that an English second language learner could understand, for example. This dataset is not designed for that, and future work on ReHash should involve finding a way around this issue – perhaps scraping a better dataset, or using a more sophisticated model than T5 such as GPT-3.

#### 4.1.2 List Conversion

This task is unique out of the three we work on in this project in that there is no adequate preexisting dataset for finetuning T5-small. Thus, we intended to produce a dataset for this task ourselves. To do this, we originally planned to use the LangChain API[11] to zero-shot prompt GPT-3 (specifically, the davinci base model) to generate simple bulleted lists from academic paragraphs. We used the ARXIV dataset from Quest 2 of this course as our set of academic tone paragraphs. Note that training our list conversion model on such a dataset may result in slight style transfer, specifically that of a professional or academic tone, but we find this to be useful for our tool given that ReHash is intended for the academic domain regardless. We planned to first experiment with different prompting approaches, then generate a dataset of around 100 samples; however, we ran out of OpenAI credits at the end of our prompt tuning experimentation and were unable to create the dataset. We will describe the prompts tested and the results of our experimentation here regardless, as it was an enlightening process.

We wanted to produce a dataset representative of that which our users may input into our model and expect in return. Thus the bulleted lists need to be simple, unprofessional, and consisting of sentence fragments. They also need to retain enough information from the paragraph they're paired with that it can be possible for a neural net to produce an output prediction relatively close to that paragraph; thus we need simple lists that are not too simple – a challenge for zero-shot prompting even a huge LLM like GPT-3. We also theorized that it may be useful to prompt GPT-3 to produce a bulleted list from the inputted abstract, and then prompt GPT-3 to further simplify that produced bullet point list. This second approach can be implemented with the LangChain API by way of a Sequential Chain. Thus we experimented with that too.

We tested quite a few prompts to find the optimum language for GPT-3 to produce lists that satisfied the conditions described above. A very simple prompt, (1) "Convert the following paragraph into a bulleted list" resulted in bullet points that were essentially complete sentences – too complex for what we expect our users to input into ReHash. We then experimented with giving GPT-3 a persona; (2) "You want to help your friend understand the following paragraph. Make a list of bullet points summarizing it. Be brief and use simple language." This prompt was wordy, but resulted in nearly acceptable bullet point lists – still just a bit too complex for what we were looking for. We then settled on the following prompt as the best for our use case: (3) "You want to help your friend understand the following paragraph. Make a list of bullet points summarizing it. Use simple language." We also tried using this prompt in a Sequential Chain, where the second prompt read: (4) "Rewrite the following bullet point list to be simpler. Use many short bullet points." However, we found that the Sequential Chain approach reduces the input abstract far too much and removes vital information from the list. Thus, as the result of our experimentation we settled on list (3). It was at this point that we realized we were out of OpenAI credits and could not generate the dataset as originally planned.

Now, it is relevant to note that we unfortunately lost the code recording our prompt tuning process as described above. We had experimented with more prompts than those mentioned in the previous paragraph, but all record of those tested prompts were lost with our code. Luckily, due to the simple and intuitive nature of the LangChain API, it wasn't much more than a few lines of code that were lost; simply a few Prompt(), SingleChain(), and SequentialChain() object initializations.

After that approach proved infeasible, we decided to generate a small, few-shot finetuning dataset instead. We generate 15 samples, 10 reserved for finetuning and 5 for evaluation. Some of these datapoints were retained from our prompt tuning process (and thus were generated by GPT-3). Others were generated by prompting CHAT-GPT to produce bulleted lists from ARXIV abstracts. Finally, because we want our dataset to simulate what users of our tool might input, we decided to diversity our dataset by prompting CHAT-GPT to give us lists about various topics that are academic-leaning but not solely based on ARXIV abstracts (e.g. movies, omega-3s, language learning, etc.). We then prompt CHAT-GPT to convert those lists into coherent paragraphs. Note that when we prompted CHAT-GPT for lists, we used the following prompt: 'Can you give me a list of simple bullet points about X?" where X was the topic of interest. When we prompted CHAT-GPT to generate paragraphs from lists, we used the following: "Can you rewrite this list as a coherent paragraph?". This final generated dataset can be found in the `/Training/ReHash_BulletPointList.ipynb` file of the GitHub repository for this work[9].

#### 4.1.3 Paraphrasing

In this task, we finetune T5-small once again, although for this task we use the Quora question pair dataset for paraphrasing[8]. We originally wanted to use the TwitterURL dataset, as according to our research it tends to be a popular dataset for this task, but were unable to get access in time. The Quora dataset consists of question pairs, in which each question pair is marked as a duplicate or not duplicate. For the purpose of paraphrasing this distinction is not useful to us, thus we ignore this label and simply extract all dataset pairs that are not exact copies of one another (the same preprocessing we applied to the PWKP dataset for the simplicity style transfer task).

Many of these pairs are still very marginally distinct from one another – in some cases only one or two words have been changed. Each datapoint is also very simplistic, which is not necessarily our intent for the Paraphrase feature of ReHash. This makes the Quora dataset not ideal for our use case, in much the same way as we are unsatisfied with using PWKP for our simplicity style transfer model. Thus, future work on ReHash should explore other methods of developing a successful paraphrase model – ie. finetuning on a different dataset such as TwitterURL, or perhaps using GPT-3 or another LLM instead of T5.

## 4.2 Experiments

#### 4.2.1 Simplicity Style Transfer

For this task, we seek to finetune T5 on simplicity style transfer to produce reasonable output text. We use the PWKP dataset as described in the Methods section above. As is the case for the entirety of this project, we are not interested in state of the art results here; we simply want to produce a functioning simplicity feature for ReHash.

In terms of technical details, we finetune T5 for conditional generation from HuggingFace (the small checkpoint) for 5 epochs using the Adam optimizer with a learning rate of 0.0005 – after experimenting with a variety of learning rates. Note that we are using the Trainer API from HuggingFace to finetune our model, as is the case for all the models described in this report. We use an 80/10/10 dataset split, using only a subset of the PWKP dataset as we're finetuning and don't need inordinate amounts of data. Thus we use only 5,000 samples for training, with 625 for validation and 625 for test. Our batch size for training and evaluation is 8. We prepend the prefix "simplify: " to each of the inputs in our dataset. Also, because our dataset has the special tokens "-LRB-" and "-RRB-" for denoting left and right parentheses, we manually add those tokens to the tokenizer and the model. We pad all inputs, targets to a length of 128, truncating any samples longer than that.

In order to gauge the functionality of this feature we evaluate using SARI, BLEU, ROUGE-1, and a readability score (specifically, Flesch-Kincaid grade level score). SARI is a metric specific to the simplicity style transfer task, and is relevant in determining our model's ability to simplify the input. BLEU and ROUGE will give us a sense of the grammaticality of the model output and its accuracy to the reference text. Finally, we choose to evaluate on readability to ensure that our model is producing coherent, easily-interpretable text.

For our baselines, we pick two competitive architectures on this task, informed by the survey paper for this task discussed in the Background section above[1]. The first, our weak baseline, is an unsupervised encoder-decoder model reported in Surya et al. 2019, scoring 38.41 SARI and 38.28 BLEU on the PWKP dataset[2]. We call this baseline UNSUP. Our strong baseline is DRESS, an encoder-decoder model with reinforcement learning introduced in Zhang and Lapata 2017[3]. This model scores a 40.04 SARI and a 34.53 BLEU on PWKP.

We expect our finetuned T5 to perform better than both these baselines but, once again, are not looking for significant gains.

#### 4.2.2 List Conversion

In this task, we finetune T5-small on list conversion using the small few-shot finetuning dataset we generated using GPT-3 and CHAT-GPT, as described in the previous section. On both our main model and our baseline, we experiment with using dashes to denote bullet points in the inputs as compared to using a special token <BPT>.

As for technical details, we finetune T5 for conditional generation from HuggingFace for 20 epochs using the Adam optimizer with a learning rate of 0.0005 – after experimenting with a variety of learning rates. Because of the limited amount of data available to us for this task, we train on 10 samples with a batch size of 1 and no validation set. We then evaluate our model's performance on 5 samples, again with a batch size of 1. We prepend the prefix "rewrite as a paragraph: " to each of the inputs in our dataset; we do not experiment with other prefixes in this work, although future work may warrant looking into the ideal prefix for finetuning T5 on this task. Note that when we train/test with the special <BPT> token, it is necessary to manually add that token to the tokenizer and the model. We pad all inputs, targets to a length of 400, truncating any samples longer than that.

We evaluate using BLEU, ROUGE-1, and FKGL readability score. Note that SARI is not applicable for this task.

As for baseline, once again the lack of research on this task makes it difficult to choose an adequate baseline. We decide to use our finetuned paraphrasing T5 model as a baseline; intuitively, a good paraphrasing model should be able to paraphrase each bullet point in the list into a sentence and produce a paragraph in that way. However, because of limitations of the dataset we finetune our paraphrasing model on (these limitations are described in section 4.1.3 of this report), we expect this to be a very weak baseline. Note that this finetuned paraphrase model we few-shot finetune on our generated dataset, using once again 20 epochs, the Adam optimizer with a learning rate of 0.0005, a batch size of 1, and input, target lengths of 400. In the case of using the <BPT> token, we also manually add this token to the baseline tokenizer and model.

We expect our model to perform much better than the baseline.

#### 4.2.3 Paraphrasing

We are interested in finetuning T5-small for paraphrasing on the Quora dataset as described in the previous section.

In terms of technical details, we finetune T5 for conditional generation from HuggingFace (the small checkpoint) for 10 epochs using the Adam optimizer with a learning rate of 0.001 – after experimenting with a variety of learning rates. We use an 80/10/10 dataset split as was the case for simplicity style transfer task, using only 5,000 samples for training and 625 each for validation and test. Our batch size for training and evaluation is 8. We prepend the prefix "paraphrase: " to each of the inputs in our dataset. We pad all inputs, targets to a length of 250, truncating any samples longer than that.

Like the bullet point conversion task, we evaluate using BLEU, ROUGE-1, and FKGL readability score.

Our baselines were once again chosen based on a survey paper of this task[4]. For a weak baseline, we use Prakash et al. 2016's deep LSTM with residual [5]. This model implementation scores a 24.97 BLEU and 57.27 ROUGE-1 on the Quora dataset. For a strong baseline, we use a vanilla transformer as reported in the same paper, which achieves a 30.38 BLEU and 61.25 ROUGE-1 on Quora[5].

We expect our model to perform significantly better than the weak baseline, and perform comparatively – or, ideally, better – than the strong baseline.

## 4.3 Results

### 4.3.1 Simplicity Style Transfer

As mentioned in the previous section, we experimented with a variety of learning rates and number of epochs in order to achieve the best model performance in finetuning our T5 model on this task.

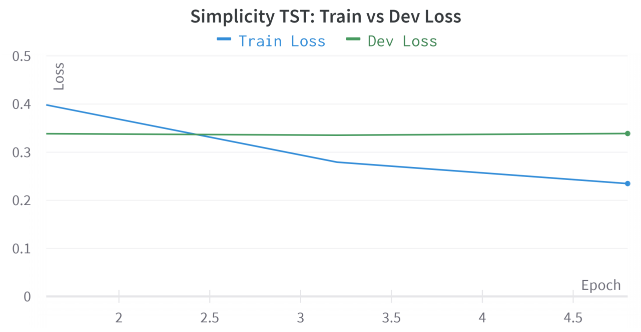


Figure 1: Simplicity TST: Loss during Finetuning

In Figure 1, we report training and validation (dev) loss during finetuning. Note that the validation loss doesn't change significantly, indicating overfitting to the train data.

Even so, our model performs very competitively on the test set as compared to both our strong (DRESS) and weak (UNSUP) baselines. These results are illustrated in Table 1 below.

Table 1: Simplicity TST: Evaluation Results on Test Set

| **Model** | **SARI** | **BLEU** | **ROUGE-1** | **FKGL** |
|---|---|---|---|---|
| Reference | 100 | 100 | 100 | 8.07 |
| Our Model | 62.74 | 54.46 | 74.14 | 7.76 |
| DRESS | 40.04 | 34.53 | — | 8.40 |
| UNSUP | 38.41 | 38.28 | — | 7.75 |

Note that the evaluation scores for DRESS and UNSUP were found in the survey paper mentioned previously in this report, and thus we do not have ROUGE-1 values for these baselines[1]. Regardless, the SARI score indicates that our model is clearly producing much simpler outputs than the baselines, showing that it learned to perform the task. The readability score for our model is as expected, at around 8. Finally, our model is able to achieve a BLEU score at least 16 points larger than the baselines.

Thus, despite the subpar validation loss convergence, our model performs well on all evaluation metrics. Because of this, we are satisfied with moving forward in implementing this model to support the simplicity feature in ReHash.

### 4.3.2 List Conversion

Because we did not have a validation dataset for this task, we are unable to examine validation loss during finetuning to assess the ability of our model to converge. Instead, we rely on evaluation metrics to determine the success of our model.

In the table below, we report our model and baseline's performance on the test set, with and without the use of a special <BPT> token to indicate bullet points.

Table 2: List Conversion: Evaluation Results on Test Set

| **Model** | **BLEU** | **ROUGE-1** | **FKGL** |
|---|---|---|---|
| Reference | 100 | 100 | 12.04 |
| Our Model (w/o <BPT>) | 44.97 | 72.91 | 11.62 |
| Our Model (w/ <BPT>) | 43.89 | 70.91 | 11.24 |
| Baseline (w/o <BPT>) | 35.58 | 66.04 | 12.66 |
| Baseline (w/ <BPT>) | 32.74 | 65.20 | 12.02 |

Surprisingly, both our model and the baseline paraphrasing model perform significantly better with dashes instead of <BPT> to denote bullet points. The original motivation behind the <BPT> token was that dashes can appear in text as hyphens between words or as punctuation, such as em dashes. Because of this potential source of confusion for the model, we expected the use of a special token to help make bullet point boundaries clear to the model. One possible explanation why the use of dashes results in better performance is because the training set is limited, and these sources of confusion are not prevalent in the train data. To explore this further, future work should develop a more extensive dataset for sufficient finetuning.

That said, Table 2 shows that our model strongly outperforms the baseline, producing reasonable BLEU and ROUGE-1 scores on the test set. The readability score, at around 11.5, is what we'd expect as we want the model to produce academic-sounding, coherent paragraphs from the input lists. Because of these scores, we feel comfortable using this model to perform list conversion in our website, although we do feel that future work is necessary to improve performance.

Also worthy of note is that our test set only consists of 5 samples; the scores reported in Table 2 are best interpreted as a rough estimate of model performance.

#### 4.3.3 Paraphrasing

For this task, we were able to record loss during training, and a comparison between train and validation (dev) loss for the best version of our paraphrasing model are displayed in Figure 2.

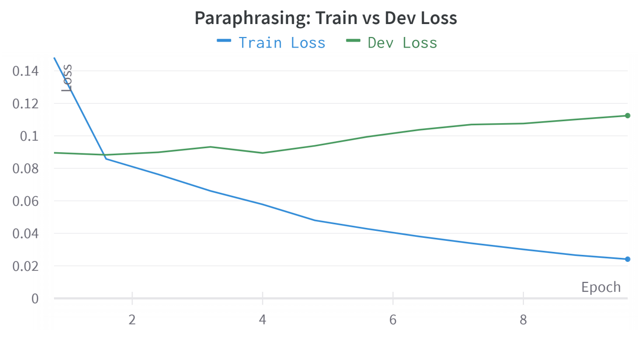


Figure 2: Paraphrasing: Loss during Finetuning

Although validation loss increases throughout training, note that the y-axis scale of this figure is very finegrained; though it's not ideal, the increase is not particularly large. The training loss decreases how we would expect during training. Similar to the simplicity model, though, the validation and train loss do not converge as we would hope – an indication of overfitting.

Still, our model performs better than our strong (Transformer) and weak (Deep Residual LSTM) baselines. Test results for all three models on the evaluation metrics are displayed in Table 3 below.

Similar to the baselines for the simplicity style transfer task, the evaluation scores for our baselines were found in the aforementioned paraphrasing survey paper, which is why we do not have readability values for these models[4]. This

Table 3: Paraphrasing: Evaluation Results on Test Set

| **Model** | **BLEU** | **ROUGE-1** | **FKGL** |
|---|---|---|---|
| Reference | 100 | 100 | 4.32 |
| Our Model | 34.41 | 62.72 | 4.16 |
| Transformer | 30.38 | 61.25 | — |
| Deep Residual LSTM | 24.97 | 57.27 | — |

isn't an issue however, because the FKGL metric was chosen to serve as an indicator of the interpretability of our model outputs anyway. Thus, with a slightly lower readability score than the reference texts, we can conclude that our model produces simplistic but interpretable outputs.

Also worthy of note in this table is that our model barely outperforms the vanilla transformer baseline in BLEU and ROUGE-1 scores. This is perhaps in line with the less than ideal training and validation loss convergence illustrated in Figure 2. Still, the improvement in performance warrants us moving forward with using this model for the paraphrasing feature of our tool.

We would like, still, to further explore different datasets or models for this task, due to our concerns about the Quora dataset as described in section 4.1.3.

## 5 Website Details

We used streamlit, a python-based open-source framework, to build our web app. The app consists of a single web page with the following features:

- To choose what task a user wants to perform, a drop-down menu is given. Figure 4 shows that the user can choose one option from Simplify, List to paragraph, and Paraphrase.
- There is a text box right below this for entering the input text.
- As the user clicks "Go" the selected task's predict function is called for generating the output. Figure 5 shows the spinner, a temporary animation with a message to indicate the execution of the task.
- The output is then displayed in the "Magic Area" section of the web app, on which if the user hovers, can use the "Copy to Clipboard" button (Figure 6). To include this functionality on the output box, we used a code box instead of a simple textbox in streamlit.
- To erase the generated output, the user can click on clear button below the output box.

- Lastly, we have included a footer consisting of the details of the developers.

Besides these features, to regenerate the outputs, the user can click on "Go". This simply calls the predict function and gives a different output. Additionally, one of the major issues faced by us while building the app was the long runtime of the application, which we solved by making the code modular and loading all the models in the beginning before any action can be made by a user.

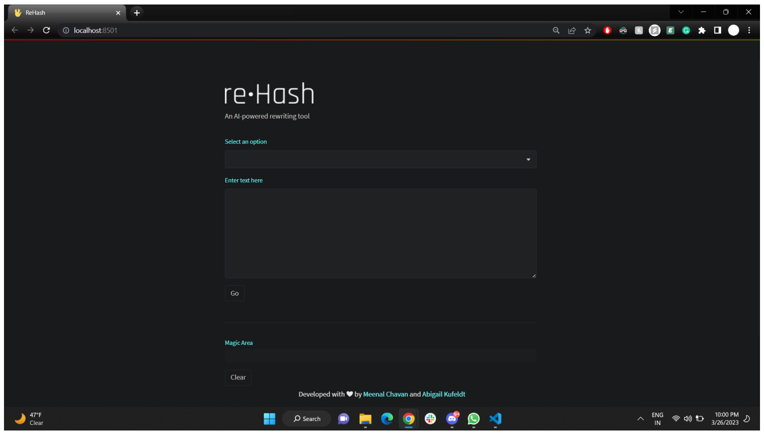

ReHash
localhost:8501
re·Hash
An AI-powered rewriting tool
Select an option
Enter text here
Go
Magic Area
Clear
Developed with ♥ by Meenal Chavan and Abigail Kufeldt

Figure 3: ReHash: Landing page

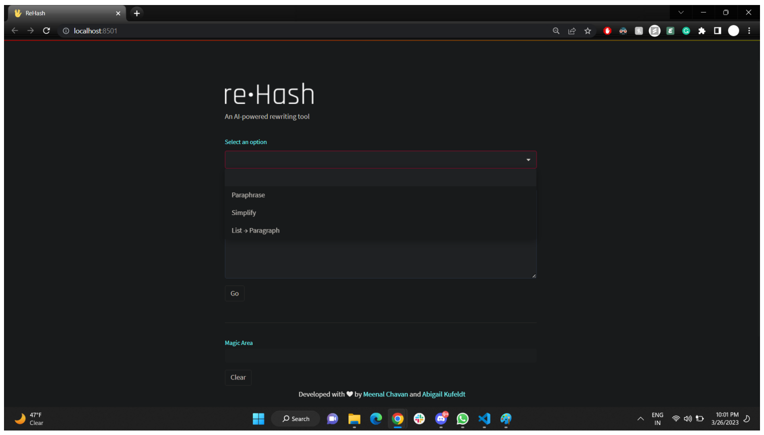
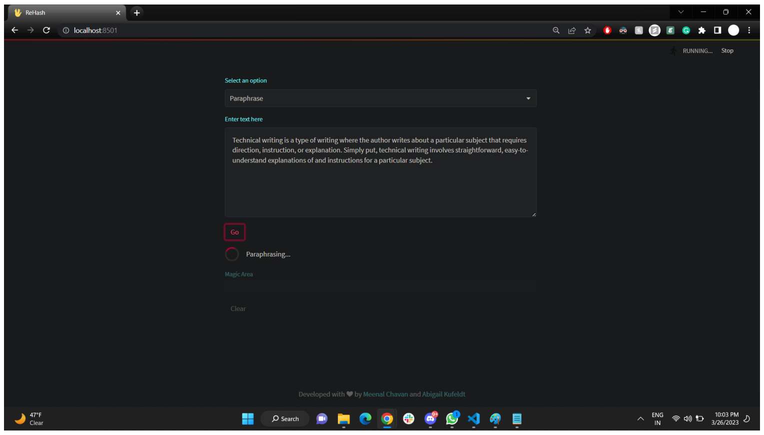


Figure 4: ReHash: Select a task to perform

## 5.1 Link

We haven't published the website for public use yet as we plan to incorporate more features and changes to make it even better. On running app.py from



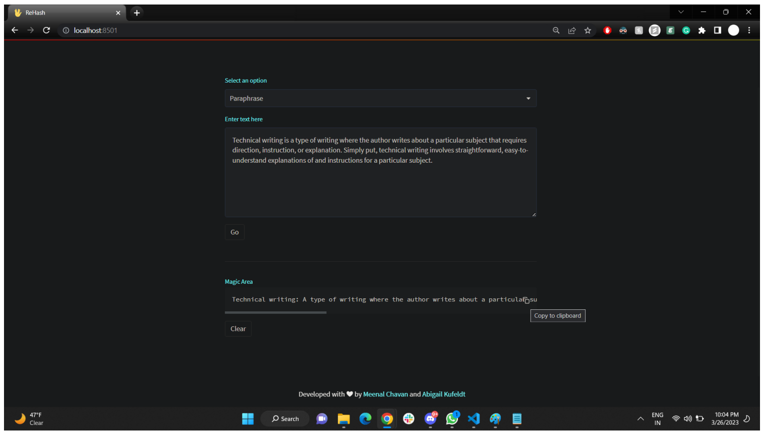
Select an option: Paraphrase

Enter text here:

Technical writing is a type of writing where the author writes about a particular subject that requires direction, instruction, or explanation. Simply put, technical writing involves straightforward, easy-to-understand explanations of and instructions for a particular subject.

Developed with ❤ by Meenal Chavan and Abigail Kufeldt

Figure 5: ReHash: Executing the input



Select an option: Paraphrase

Enter text here:

Technical writing is a type of writing where the author writes about a particular subject that requires direction, instruction, or explanation. Simply put, technical writing involves straightforward, easy-to-understand explanations of and instructions for a particular subject.

Magic Area

```
Technical writing: A type of writing where the author writes about a particula su
```

Developed with ❤ by Meenal Chavan and Abigail Kufeldt

Figure 6: ReHash: output with option to copy it

https://github.com/meenal-chavan/ReHash using streamlit run app.py command, you would get a localhost URL and a network URL which would enable the user to access the website and play around.

## 6 Acknowledgements

We would like to acknowledge Brendan King, our TA, and Professor Delip for the advice and support shown to us throughout this quarter on our project.

## 7 References

[1] Data-driven sentence simplification: Survey and benchmark (no date). Available at: https://www.researchgate.net/publication/338350367$_{D}ata-Driven_{S}e ntence_{S}implification_{S}urvey_{a}nd_{B}enchmark(Accessed: March27, 2023)$.

[2] Unsupervised neural text simplification - arxiv.org (no date). Available at: https://arxiv.org/pdf/1810.07931v6.pdf (Accessed: March 27, 2023).

[3] Sentence simplification with Deep Reinforcement Learning - ACL Anthology (no date). Available at: https://aclanthology.org/D17-1062.pdf (Accessed: March 27, 2023).

[4] Paraphrase generation: A survey of the state of the art - ACL anthology (no date). Available at: https://aclanthology.org/2021.emnlp-main.414.pdf (Accessed: March 27, 2023).

[5] Prakash, A. et al. (2016) Neural paraphrase generation with stacked residual LSTM Networks, arXiv.org. Available at: https://arxiv.org/abs/1610.03098 (Accessed: March 26, 2023).

[6]Anastasia Shimorina / webnlg-dataset · GITLAB (no date) GitLab. Available at: https://gitlab.com/shimorina/webnlg-dataset (Accessed: March 26, 2023).

[7]Dataset for Style Transfer (no date) Text simplification data sets. Available at: https://cs.pomona.edu/ dkauchak/simplification/ (Accessed: March 26, 2023).

[8] First Quora dataset release: Question pairs (no date) Quora. Available at: https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs (Accessed: March 26, 2023).

[9] Meenal-Chavan (no date) Meenal-Chavan/rehash, GitHub. Available at: https://github.com/meenal-chavan/ReHash (Accessed: March 26, 2023).

[10] Streamlit docs (no date) Streamlit documentation. Available at: https ://docs.streamlit.io/ (Accessed: March 26, 2023).

[11] Welcome to Langchain (2023) Welcome to LangChain - LangChain 0.0.123. Available at: https://python.langchain.com/en/latest/index.html (Accessed: March 26, 2023).