Understanding Iterative Revision from Human-Written Text

Pratish Mashankar
G01354094
George Mason University
pmashank@gmu.edu

Sai Likhitha Allanki
G01336091
George Mason University
sallanki@gmu.edu

William J David
G01129185
George Mason University
wdavid2@gmu.edu

1 Introduction

1.1 Task / Research Question Description

The paper aims at addressing the task of developing a comprehensive framework for modeling and understanding the iterative text revision process, with the goal of improving the quality and efficiency of the writing process through the creation of large-scale, multi-domain, edit-intention annotated corpora. The research question that the paper seeks to answer is how to develop a framework that can comprehensively model the iterative text revision process, including various domains of formal writing, edit intentions, revision depths, and granularities, and how to use this framework to create annotated corpora that can support computational modeling of iterative text revisions. Additionally, the paper investigates the relationship between edit intentions and writing quality and explores how incorporating annotated edit intentions can improve automatic evaluations of generative and edit-based text revision models. On these models, we perform Robustness and Multilinguality tests and document our results.

1.2 Motivation and Limitations of existing work

There are other papers within this area in which the paper identifies itself. The authors identified a flaw with how these papers were conducting the training and how these other papers hadtheir data set up. The idea introduced by Du et al. was that writing is an iterative process andother papers which were working on the revision task were doing it in a non-iterative way. Foremost the data used in prior papers were a first draft and the final version of that draft as the inputoutput pairs. Secondly, other papers tend to focus solely on sentence-level revisions rather than whole document revisions. And finally, models used in these other papers tend to focus on one domain like exclusively Wikipedia articles or exclusively academic papers. The authors of this paper present the IteraTeR dataset, which alleviated these issues. The data set spans multiple domains, focusing on Wikipedia, Wikinews, and arXiv; the data has paragraph-level, and sentence-level edits as data points; and the data itself is in multiple states of the iterative revision process with the first part being with the first edits and the final part being the last couple edits made. With this dataset, the authors also created models for the purposes of performing the revision task like the papers they previously identified.

1.3 Proposed Approach

1.3.1 Models for Intent Classification

We compared RoBERTa (Robustly Optimized BERT Pretraining Approach) and XLM Roberta (Robustly Optimized BERT Pretraining Approach) for performing the intent classification on our data. Both these models are based on the transformer architecture and are pre-trained using a large amount of text data, allowing them to effectively capture the nuances of language.

1.3.2 Models for Edit Generation

We used PEGASUS (Pre-training with Encoder-decoder Guides for Automated Summarization) and mT5 (Multilingual Text-to-Text Transfer Transformer.) Pegasus is a transformer-based model that uses an encoder-decoder framework with an autoregressive decoder to generate high-quality summaries and edited text for a wide range of languages and contexts. mT5 is also a transformer-based model that is pre-trained using a large amount of multilingual text data, allowing it to also effectively generate high-quality transla-

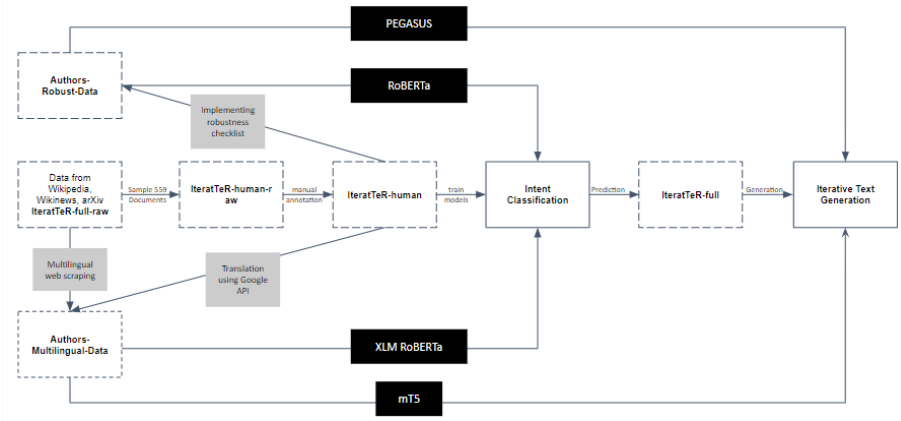

Figure 1: Workflow of our Reproducibility, Robustness, and Multilinguality tests

tions and text edits.

1.3.3 Languages and Multilinguality

We chose these five languages, namely English, French, Mandarin, Hindi, and Tamil, to perform a multilinguality test on our NLP models for several reasons.

We decided to keep a few English sentences as it is a widely spoken language and is the lingua franca of business, science, and technology. It belongs to the Germanic branch of the Indo-European language family and is the most commonly used language for NLP tasks, making it an essential language to test our models' performance.

Secondly, French belongs to the Romance branch of the Indo-European language family and is the official language in many countries, including France, Canada, and several African nations. Testing our models' performance on French data allowed us to evaluate their ability to handle languages from different branches of the same family.

Thirdly, Mandarin is a widely spoken language and the most spoken language in the world. It belongs to the Sinitic branch of the Sino-Tibetan language family, which is structurally different from Indo-European languages. Testing our models' performance on Mandarin data allowed us to evaluate their ability to handle non-Indo-European languages.

Fourthly, Hindi is an Indo-Aryan language spoken by over 500 million people in India and other parts of the world. As a member of the Indo-European language family, Hindi has a different structure than Mandarin and allowed us to test our model's ability to handle languages from different subfamilies of the same family.

Lastly, Tamil is a Dravidian language spoken in South India and Sri Lanka. As a non-Indo-European language, it has a different structure from the other languages we chose, allowing us to evaluate our models' ability to handle typologically distinct languages.

Testing our NLP models on these five languages provided us with a diverse set of linguistic structures and allowed us to evaluate their ability to handle languages from different families and branches, which was crucial in developing robust and accurate models.

1.3.4 Robustness

When working with models that classify intent or generate text, it's important to ensure that they are robust and can handle a variety of input data. In order to test for robustness, we need to introduce noise, errors, and other variations to the data to see how the models react. The NLP Checklist is a valuable tool that helps us generate data with these variations and test the models accordingly. It is also noted that adding things like URLs, switching locations, and adding negation should not cause a model to react negatively.

The NLP Checklist provides a set of guidelines and methods for perturbing existing data to create new examples that differ in subtle ways. For example, we can use the checklist to add spelling mistakes, and grammatical errors, or switch around the order of words in a sentence. These changes are designed to simulate the kind of variations that real-world input data might have, such as typos, missing punctuation, or sentence fragments.

In addition to these common variations, the

NLP Checklist also includes more complex changes that require a deeper understanding of language and context. For instance, we can add negation to a sentence, change the tense or voice, or replace named entities with different values. This would include things like assigning a mostly male-dominated career to a name normally feminine name, in this case, a robust model would output the result expected for that sentence. In the same case, a non-robust model would fail. These changes require the model to understand the meaning and context of the input sentence and generate an appropriate response.

One important aspect of testing for robustness is to ensure that the models can handle unexpected input data without misclassifying or generating bad revisions. To achieve this, the NLP Checklist includes scenarios that introduce more complex variations to the input data. These might include adding URLs or email addresses, switching locations or time periods, or introducing conflicting temporal references. The goal is to simulate more complex real-world scenarios where the model needs to be able to recognize and respond to subtle differences in the input data.

1.4 Likely challenges and mitigations

Our team project had its fair share of challenges, primarily due to the complexity of the research question. We encountered a significant challenge while implementing the user interface based on the research paper we had initially selected. The webpage was taking a long time to load, which made it difficult for us to work with the interface effectively. We hence decided to pivot our approach and focus on the authors' initial paper released a few months before the current one. Our contingency plan involved testing the dataset generation code for intent prediction and evaluating the model's performance using SARI BLEU and RL scores. Although we couldn't delve into the document-level paragraph edits, we made steady progress in analyzing the project's sentence-level aspects

2 Related Work

"Beyond Accuracy: Behavioral Testing of NLP Models with CheckList". (Ribeiro et al., 2020) introduces CheckList, a methodology for testing NLP models that includes a matrix of general linguistic capabilities and test types, as well as a software tool for generating diverse test cases. The authors demonstrate the effectiveness of CheckList by using it to test commercial and state-of-the-art models in three tasks and show that it can identify critical failures. The paper argues that CheckList provides a more comprehensive approach to evaluating NLP models beyond traditional held-out accuracy measures and demonstrates that NLP practitioners can create more tests and find more bugs using CheckList.

"Multilinguality Checklist and Generation". (K et al., 2022) presents a solution for creating multilingual CheckLists that are useful for evaluating NLP models. It is challenging to create these CheckLists due to language diversity. Therefore, the authors propose the Template Extraction Algorithm (TEA) that extracts target language CheckList templates from machine-translated source language templates. The authors experiment with multiple languages and found that TEA followed by human verification provides a good estimate of model performance. They release the code of TEA and the CheckLists created in multiple languages to facilitate future work in this area. This paper provides a cost-effective and scalable method for creating multilingual CheckLists for NLP model evaluation.

"wikiHowToImprove: A Resource and Analyses on Edits in Instructional Texts" (Du et al., 2022) Understanding Iterative Revision from Human-Written Text, analyzes the quality of instructional texts and proposes a method to train models to do sentence-level edits. "Text Editing by Command" proposes an interactive text editing system based on user commands, while the third paper introduces the PEGASUS model for abstractive summarization. Wordcraft: a Human-AI Collaborative Editor for Story Writing" (Coenen et al., 2021), the Wordcraft text editor utilizes a few-shot learning algorithm and a conversational network to assist with creative writing. Overall, these papers showcase various approaches and techniques for improving the quality and efficiency of text editing and generation.

3 Experiments

3.1 Comparing Baseline models

In this study, we compared our results with a baseline method employed by Wanyu D et al. For Intent Classification, both we and Wanyu D et al used a pre-trained language model, but we also

used DistillBERT in our experiments. The results of our experiments were similar to those reported by Wanyu D et al, with a minor difference of ±0.1. For text generation, Wanyu D et al used BART and Pegasus, with Pegasus producing better results, which was also reflected in our case. As a baseline for our robustness tests, we used both Roberta and Pegasus. Additionally, for multilingual testing, we performed a zero-shot execution on both Roberta and Pegasus, and used these models as our baseline. To further compare our results, we also used XLM Roberta and mT5 for intent classification and text generation, respectively, using both zero-shot and fully supervised methods, and compared them with our multilingual baseline.

3.2 Evaluation Metrics

3.2.1 Metrics for Classification

We implemented several metrics to evaluate the performance of our intent classification model, including accuracy, precision, recall, and F1 score.

Accuracy: was used to measure the overall performance of our model, calculated as the ratio of the number of correct predictions to the total number of predictions made. While accuracy provided us with a high-level understanding of how well our model was performing, it was not sufficient when the classes were imbalanced.

Precision: was used to measure the proportion of correct positive predictions out of all the positive predictions made by the model. This helped ensure that the model did not incorrectly classify a sentence as belonging to a certain intent when it actually did not.

Recall: We also used precision to measure the proportion of correct positive predictions out of all the positive predictions made by the model. This was useful when we wanted to minimize false positives and ensure that the model did not incorrectly classify a sentence as belonging to a certain intent when it actually did not.

F1 score: which was the harmonic mean of precision and recall, provided a single metric that balanced between precision and recall. We found the F1 score to be particularly useful when dealing with imbalanced classes. It allowed us to assess the trade-off between precision and recall and identify areas where our model could be improved.

Using these metrics, we identified the strengths and weaknesses of our model and made informed decisions on how to improve it. For example, if we observed low precision for one of the intents, we analyzed the misclassified examples and modified our model accordingly to improve precision. Similarly, if we observed low recall for a certain intent, we focused on improving recall by adding more training examples or modifying our model architecture.

3.2.2 Generation metrics

SARI : (System output Against Reference and the references Improved) is a metric that measures the similarity between the system-generated output and human-generated reference sentences. It takes into account not only the content overlap but also the fluency and grammaticality of the output. The SARI score ranges from 0 to 4, with higher scores indicating better performance.

BLEU: (Bilingual Evaluation Understudy) is a metric that measures the similarity between the system-generated output and the human-generated reference sentences. It is based on the n-gram overlap between the two sets of sentences. The BLEU score ranges from 0 to 1, with higher scores indicating better performance.

RougeL: (Recall-Oriented Understudy for Gisting Evaluation) is a metric that measures the similarity between the system-generated output and the human-generated reference sentences. It is based on the longest common subsequence between the two sets of sentences. The RougeL score ranges from 0 to 1, with higher scores indicating better performance.

By using these metrics, we were able to comprehensively evaluate the performance of our iterative text generation model. For example, if we observed that the SARI score was low for a particular sentence, we could analyze the system-generated edit and identify ways to improve its fluency and grammaticality. Similarly, if we noticed that the BLEU or RougeL score was low for a specific sentence, we could investigate further to determine whether the issue was with the content overlap or the sentence structure.

3.3 Datasets

The data we are using for robustness is the Human Sentence Level data proposed by the authors and the data generated as described in the approach section regarding robustness data. The original Human Sentence Level data is still in use for the purposes of the train and dev sets. The test sets are exclusively the robustness data we generated ourselves. The reason we are not generating robustness data for the train and dev set is that we want to test the generated models' robustness without the models being overfitted for the robustness test cases.

We have been working with the Human Sentence Level datasets proposed by the authors. Our goal is to evaluate the multilingual capabilities of the models on this task. To do this, we have decided to create multilingual versions of the datasets by translating them into several languages: French(fr), Chinese (zh-cn), Hindi(hi), and Tamil(ta). To achieve this, we have used machine translation to convert the original train, dev, and test datasets into each of these languages. This has allowed us to create multilingual datasets that are required for the task. By combining these multilingual datasets, we have created comprehensive train, dev, and test multilingual sets that can be used to test the ability of models to handle different languages for the proposed task.

3.4 Implementation

Google Colab Link: https://colab.research.google.com/drive/1oUtjXqfODzFj1EJOWFKjvNkA0fJ9wyzf?usp=sharing

Git Hub Link: https://github.com/PratishMashankar/cs678-cp1-cp2

3.4.1 Performing Robustness Tests

To ensure the robustness of our models, we generated two sets of data to test their performance. The first dataset was created for the intent classification task, using the NLP Checklist import to perturb the original data. We employed various methods such as introducing typos, negation, expanding and contracting contractions, removing punctuation, and changing names to create noisy data. The perturbed data was then labeled and saved in a new JSON file format identical to the original data. This dataset was used to test the original RoBERTa model as described in the paper.

For the second dataset, we aimed to evaluate how well the PEGASUS model could generate revised text. We again used the NLP Checklist import to perturb data, but this time we generated new sentences instead of modifying the existing data. The base sentences included elements such as URLs, grammatical errors, and conflicting temporal references. We then used the NLP Checklist's editor function to fill in various areas with different names, pronouns, verbs, and nouns to create a large dataset. Since the amount of data generated was too large to label manually, we randomly sampled from it and assigned labels in a way that allowed us to assess how the PEGASUS model would revise a data point with a selected label.

It's important to note that this second dataset was not suitable for the RoBERTa model's intent classification task, as the labels were arbitrary and not a good reflection of the model's robustness. However, it was useful in evaluating the PEGASUS model's ability to generate revised text and its performance on perturbed data.

3.4.2 Performing Multilinguality Tests

Translation module: We created multilingual datasets by translating text in JSON files into four other languages using the Google Translate API. We looped over each target language and JSON file combination, extracted and translated the 'before_sent' and 'after_sent' fields, and updated the dictionaries with the translated text and intent labels. We then wrote the updated dictionaries to new JSON files, resulting in four new JSON files for each original JSON file, containing the same data but with text translated into four other languages..

Multilinguality models: XLMRoBERTa and mT5 are two pre-trained transformer-based language models that can handle multilingual text processing tasks. They are useful for intent classification and generation in a multilingual context as they can handle multiple languages and can learn representations of language shared across different languages. XLMRoBERTa is based on RoBERTa but has been trained on a large corpus of multilingual text, while mT5 is designed specifically for multilingual natural language processing tasks, capable of handling text in over 100 different languages and achieving state-of-the-art results on several multilingual

NLP tasks.

Using XLMRoBERTa and mT5 for multilinguality in intent classification and generation is essential as it allows us to handle text in multiple languages and generate accurate and coherent responses in different languages. This is particularly important in a global context where multilingual communication is increasingly common. XLMRoBERTa and mT5's ability to learn representations of language shared across different languages enables us to classify and generate intents in a language-independent way, making them valuable tools for multilingual natural language processing.

For the multilinguality checklist, we evaluated four different models - Roberta, XLMRoBERTa, Pegasus, and mT5 - in two ways: zeroshot and fully supervised learning. In zeroshot, we used the original English train and dev data and multilingual test data. In fully supervised learning, we used multilingual train, dev, and test data. We adjusted the hyperparameters for both methods and found that our models performed well after 10 epochs.For the intent classification task, we used the RoBERTa and XLMRoBERTa models. For the text generation task, we used the Pegasus and mT5 models. We evaluated the multilinguality checklist using these models.To use these models appropriately, we had to make some changes to the shell scripts. We added the huggingface path to our shell scripts so that we could run the models effectively.

3.5 Results and Discussions

3.5.1 Robustness Results

For the Intent classification task we observed that the RoBERTa model tested on robustness data, preformed about as well as the base RoBERTa model when looking at the various label's precision, recall and F1 scores we can see that the RoBERTa model has similar accuracies across the board. Notably, however we see the prediction of Coherance and Meaning Change labels increase. We believe the reason for this is due to the increase of spelling, grammatical and overall errors in the text increases when we add the corresponding changes to the test data. One explanation is that these labels are the ones that see the most amount of in text error changes. So adding significant typos to a text may cause the RoBERTa model to predict a coherence revision more frequently and a

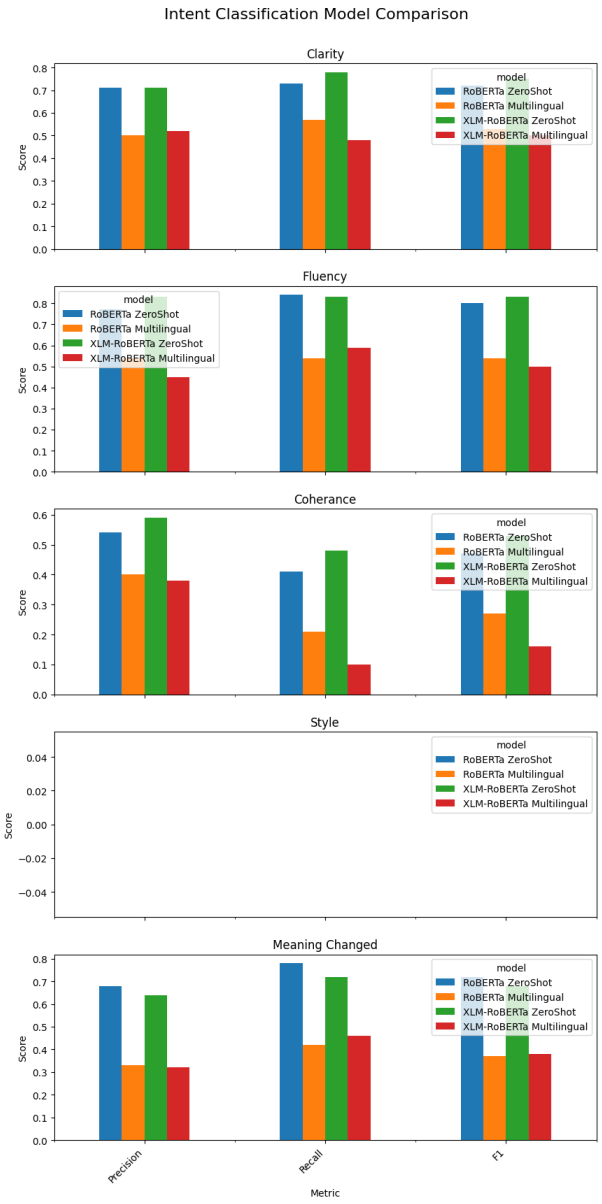

Figure 2: Intent Classification Models Comparisions for Multilingual Data

similar reaction may occur when data is perturbed for more negation instances making the meaning change revision to be predicted. As for the revision generation task with the PEGASUS model we observed almost similar results when comparing the BLEU, ROUGE and SARI for the robust and standard test data. Both the ROUGE and the SARI scores achieved with the robustness testing where similar but slightly higher compared to the standard test data, however the BLEU score that was achieved though had a twenty percent drop. When looking at the numerical scores overall one could believe that the PEGASUS model is a robust model. However if you look at the generated text

Robustness Test				
Intent Classification				
Model	Label	Precision	Recall	F1
RoBERTa clean	Clarity	0.72	0.72	0.72
	Fluency	0.73	0.82	0.77
	Coherance	0.45	0.34	0.38
	Style	0	0	0
	Meaning Changed	0.58	0.62	0.58
RoBERTa robust	Clarity	0.72	0.68	0.70
	Fluency	0.73	0.87	0.79
	Coherance	0.47	0.41	0.44
	Style	0	0	0
	Meaning Changed	0.64	0.72	0.68

Edit Generation			
Model	RougeL	BLEU	SARI
Pegasus clean	0.84	0.70	30.1
Pegasus robust	0.87	0.53	31.83

Table 1: Robustness Test Results

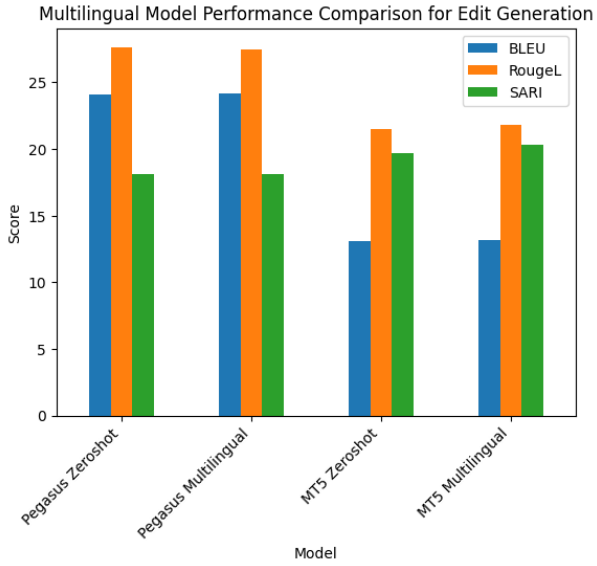

Figure 3: Edit Generation Models Comparisions for Multilingual Data

from the PEGASUS model you are able to see that some of the revised text is off. If a piece of text has significant errors like a large portion is just typos the model will not be able to reconize what sort of revision it should make, an example of this was a test case where the model just repeated the intial sentence over and over again without making any changes to the initial sentence. It is good to note though the generated model does reconize thing like URLs and it can make smaller grammatical revisions like changing him to he, in the right cases. We believe the generated revisions are like this because of the significant number of errors within the generated data, and that the model is able to preform well if the imputed test text has minimal errors. So it would be useful for a student writing a paper but not a child writing a bunch of random gibberish.

3.5.2 Multilinguality Results

Our study found that the zero-shot models outperformed the fully supervised models, even though the latter had access to labeled data for all five languages, including English, French, Hindi, Tamil, and Chinese. This could be due to the transfer learning approach used by the zero-shot models, which allowed them to generalize to new languages and tasks with limited data by leveraging their pre-trained knowledge.

Moreover, the performance of the models could have been influenced by the quality and quantity of the training data, as well as their architecture and hyperparameters. These findings highlight the benefits of transfer learning and pre-training in machine learning, particularly in multi-lingual datasets, and emphasize the importance of carefully selecting the appropriate model and training data for each specific task.

MultilinguaityTest					
Intent Classification					
Model	Accuracy	Label	Precision	Recall	F1
RoBERTa ZeroShot	0.71	Clarity	0.71	0.73	0.72
		Fluency	0.77	0.84	0.8
		Coherance	0.54	0.41	0.47
		Style	0	0	0
		Meaning Changed	0.68	0.78	0.72
RoBERTa Multilingual	0.48	Clarity	0.5	0.57	0.53
		Fluency	0.54	0.54	0.54
		Coherance	0.4	0.21	0.27
		Style	0	0	0
		Meaning Changed	0.33	0.42	0.37
XLM-RoBERTa ZeroShot	0.72	Clarity	0.71	0.78	0.75
		Fluency	0.83	0.83	0.83
		Coherance	0.59	0.48	0.53
		Style	0	0	0
		Meaning Changed	0.64	0.72	0.68
XLM-RoBERTa Multilingual	0.44	Clarity	0.52	0.48	0.5
		Fluency	0.45	0.59	0.5
		Coherance	0.38	0.1	0.16
		Style	0	0	0
		Meaning Changed	0.32	0.46	0.38

Edit Generation			
Model	BLEU	RougeL	SARI
Pegasus Zeroshot	0.24	27.67	18.08
Pegasus Multilingual	0.24	27.48	18.08
MT5 Zeroshot	0.13	21.51	19.73
MT5 Multilingual	0.13	21.83	20.33

Table 2: Multilinguality Test Results

Intent Classification Analysis: Furthermore, we observed that XLMRoberta outperformed Roberta in terms of classifying the multilingual data. One possible explanation for this performance difference is the design of the two models. XLMRoberta is a cross-lingual model that has been specifically pre-trained on a large and diverse set of languages, whereas Roberta is a monolingual model that has been trained only on English data. The pre-training of XLMRoberta on multiple languages enabled it to capture cross-lingual patterns and improved its performance on multilingual data, whereas Roberta could have struggled with languages it hadn't been trained on. Another factor that could have influenced the performance of the models is the size and quality of the pre-training data. XLMRoberta has been pre-trained on a large and diverse set of languages, which could help it to capture more generalizable features across languages. In contrast, Roberta has only been pre-trained on English data, which may limit its ability to generalize to other languages.

Additionally, XLMRoberta has a more complex architecture than Roberta, with additional layers and attention mechanisms designed to capture cross-lingual patterns. The hyperparameters used during training could also have been optimized differently for XLMRoberta compared to Roberta, which could have affected their performance on multilingual data.

Edit Generation Result Analysis: We com-

pared Pegasus and MT5 in generating edited text and found that Pegasus performed better in BLEU and Rouge-L, while MT5 outperformed Pegasus in SARI. One possible explanation for this discrepancy is that SARI is a more comprehensive metric that takes into account the fluency and meaning preservation of the generated text, which MT5 may have performed better in.

The difference in performance between Pegasus and MT5 could also be attributed to their architectures. Pegasus uses an encoder-decoder framework with an autoregressive decoder, while MT5 uses a sequence-to-sequence model with a non-autoregressive decoder. The autoregressive nature of Pegasus' decoder may allow it to better capture dependencies between input and output text.

Finally, it's worth noting that the two models have different pre-training objectives, with Pegasus using masked language modeling and next-sentence prediction, and MT5 using denoising auto-encoding. This difference in pre-training objectives may have also influenced their ability to capture the linguistic diversity of the dataset. Overall, our observations suggest that the autoregressive decoder architecture and pre-training objectives of Pegasus may contribute to its superior performance in generating edited text for a variety of languages and contexts.

3.6 Resources

The team members for this project, Pratish, William and Likhitha played different roles and made significant contributions to the project. Pratish worked on creating the Git repository, aggregated the code in the final Google Colab, and defined the process flow for completion of the project. He also assisted in training the models that are being used in project. William worked on the execution of Robustness task needed for the project and evaluated the robustness metrics. Likhitha worked on Multilingual task by gathering the multilingual data and running and training the models accordingly. Also evaluating the performance of multilinguality of the models. Togetherly we have worked on writing the report, error analysis and results.
The cost of reproducing the project in terms of resources was significant, requiring considerable computation, time, people, and development effort. The team used Google Colab GPU and created a Git repository to manage the multiple shell scripts required for the project. Training times varied for different models. The entire project took 4 weeks to complete, with with regular communication through calls and in-person meetings. However, there were challenges related to the GPU power offered by Colab, which prevented the team from running Pegasus and BART models for document-level changes data. Despite the challenges, the team was able to produce the desired results through the collective efforts of all members.

3.7 Error Analysis

Robustness: For intent classification we found that the RoBERTa model preformed just as similar on the robust data as compared to the non robust data. From the results we found on the intent classifier case the RoBERTa model was making the same mistakes on the robust data as compared to the normal test data. The same is not true for the PEGASUS model when we where testing with the robust dataset. An extreme example of a poor revision that was made by the PEGASUS model was when the an imputed sentence had major grammar issues and the model outputed the same sentence over and over. In some cases the model does not make any revisions at all this is a positive in the cases where the imputed sentence had URLs within it and there was not mistakes how ever when there was major typos the model was unable to make any sort of revisions likely due to not being able to understand what was happening in typo covered text.

Multilinguality: We performed an error analysis on the workflow that involved implementing Roberta and XLMRoberta for intent classification followed by suggesting edits using Pegasus and mT5 models. One major finding was that sentences in Hindi and Tamil, which belong to the Indo-European and Dravidian language families respectively, were not classified accurately by the models. This could be attributed to the fact that these languages have complex syntax and morphology, which make it challenging for the models to capture the nuances of the language. Additionally, the training data used to train the models could have been insufficient or not diverse enough to handle the intricacies of these languages. Another observation we made was that the style metric received a precision score of

0. This was because the frequency of the style metric was low, which means that there were not enough instances of this metric in the training data for the model to learn from. This highlighted the importance of having a diverse and balanced training dataset that captures all the possible variations and niches of the language.

Furthermore, we noticed that the models stopped suggesting edits after a depth of 2, while the human-annotated data had more plausible edits. This was particularly evident in languages like Chinese, where the sentence structure has less effect on the edit. A key takeaway was the idea of exploring other models and techniques, such as neural machine translation and transfer learning, which could help improve the performance of the workflow.

4 Conclusion

In conclusion, our reproducibility, robustness, and multilinguality tests on Wanyu D's paper make an important contribution to the development of effective and efficient multilingual and robust models. The paper's successful zero-shot evaluation demonstrates the robustness of their models and provides promising results for future work in multilingual modeling. We observed that Pegasus models were robust to some extent of noise which was not the case with RoBERTa for our datasets. And while RoBERTa and Pegasus had decent multilingualism scores their multilingual counterparts, XLM-RoBERTa and mT5 outperformed by a small margin. However, to further improve the robustness and applicability of their models, it is important to continue experimenting with more diverse and larger datasets, including more languages and domains. In addition, future work should investigate the models' robustness with respect to multilingualism and evaluate them on more complex multilingual tasks, such as cross-lingual transfer learning and language-specific tasks. This would help to identify potential limitations and biases in their models and provide insights into how to improve their robustness in more complex multilingual scenarios.

References

Andy Coenen, Luke Davis, Daphne Ippolito, Emily Reif, and Ann Yuan. 2021. Wordcraft: a human-ai collaborative editor for story writing.

Wanyu Du, Vipul Raheja, Dhruv Kumar, Zae Myung Kim, Melissa Lopez, and Dongyeop Kang. 2022. Understanding iterative revision from human-written text. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Karthikeyan K, Shaily Bhatt, Pankaj Singh, Somak Aditya, Sandipan Dandapat, Sunayana Sitaram, and Monojit Choudhury. 2022. Multilingual checklist: Generation and evaluation.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist.