The purpose of this project was to generate summaries of stories from the Brothers Grimm Fairy Tales. To accomplish this, we utilized transformer models. Transformer models are a type of deep learning model that handles sequences of data. Because it handles sequential data well, transformers are particularly adept at handling text data. They perform tasks such as summarizing text, translating languages, and analyzing sentiment. For text summarization specifically, encoder-decoder architectures are typically used. The encoder processes the input text and compresses the information, and the decoder generates the summary based on the encoder output.

For this project, we used the pre-trained BART (Bidirectional and Auto-Regressive Transformers) model from Hugging Face. We chose this model specifically due to its ability to handle large-scale data. The BART model allows up to 1024 tokens compared to the previous BERT (Bidirectional Encoder Representations from Transformers) model that we used which only allowed for 512 tokens to be analyzed at a time by the model. This feature was important for our task as the stories used from the Grimm Fairy Tales consisted of long passages with on average 2500 words. Thus, we wanted to ensure that the model used would effectively consume this data.

Even while using the model better suited for large-scale text analysis, we still had to cut the stories into smaller segments. Originally, we used 500 characters for each segment to ensure that the model would be able to handle the input, as often characters correspond to more tokens. After confirming that 500 characters was within the model's bandwidth, we increased the limit of each segment to 700 characters.

Hyper parameter tuning:

In round 1 we used the default configuration for the hyperparameters: max_length=50, min_length=15, top_k=100, top_p=0.85, temperature=0.6, repetition_penalty=1.5, and num_beams=6. This resulted in a summary that has a couple incomplete sentences making it a little hard to interpret or understand. The summary overall is also incomplete as it's missing a chunk of the end of the Golden Bird story. Here are the rouge scores for story 1 and story 2 respectively: 'rouge1': 0.4240740740740741, 'rouge2': 0.3903644224830142, 'rougeL': 0.4, 'rougeLsum': 0.42098765432098756 'rouge1': 0.40206399121493464, 'rouge2': 0.3560048680183109, 'rougeL': 0.3667789757412399, 'rougeLsum': 0.39748843632491426. These score are all mediocre and we will find later that changing the hyperparameters will improve the scores.

In round 2 we used changed the hyperparameters to the following: max_length=60, min_length=15, top_k=150, top_p=0.8, temperature=0.6, repetition_penalty=1.5, and num_beams=8. This resulted in a similar summary to the round 1. However the rouge score have improved going up to the following: 'rouge1': 0.4669867947178871, 'rouge2': 0.4342342342342343, 'rougeL': 0.43817527010804314, 'rougeLsum': 0.4639855942376951 and 'rouge1': 0.45495902641212127, 'rouge2': 0.4121203617634507, 'rougeL': 0.41753121482055855, 'rougeLsum': 0.4502158840189772. These scores are pretty good and are a much better improvement to round 1.

Finally in round 3 we changed the hyper parameters to: max_length=60, min_length=15, top_k=200, top_p=0.7, temperature=0.6, repetition_penalty=1.5, and num_beams=10. This resulted in the following rouge scores: 'rouge1': 0.45131938125568705, 'rouge2': 0.4157814871016692, 'rougeL': 0.41370943281771305, 'rougeLsum': 0.44949954504094636 and 'rouge1': 0.44632917389076193, 'rouge2': 0.40301269477034674, 'rougeL':

0.40379979278194716, 'rougeLsum': 0.4402193857801417. The summary for this one is more or less the same as the other two and the scores are slightly lower than round 2's. Due to this, we decided to settle on Model 2 (round 2) as the model with the most optimal hyper parameters.

Overall our model does an ok job at summarizing the stories. There are still parts of the summary that do not really make any sense. And the rouge scores are not perfect. Perhaps in the future a different model can be used, or more tuning can be done. Also summarizing all of it instead of a chunk would help a lot as well.