NLP Using Transformers

Question Answering on Alice’s Adventures in Wonderland

Project and Dataset Selection

This project uses Alice's Adventures in Wonderland by Lewis Carroll, obtained from the Project Gutenberg website. The book is available in plain text format, making it suitable for NLP tasks that require direct access to the raw text.

The novel is also a good choice for transformer-based question answering because it contains a mixture of descriptive narration, dialogue, and character interactions. This variety provides a meaningful test of how well a QA model can extract information from different writing styles within the same document.

NLP Tasks Using Transformers

The chosen NLP task is an extractive question answering using a pretrained transformer from HuggingFace. This approach allows the model to highlight the exact text span that answers a given question. Since novels contain information spread across narration and dialogue, this task provides a clear way to evaluate how well transformers handle long, continuous text. The book is split into smaller chunks so the model can process it, and retrieval methods help select the most relevant sections before running QA.

Data Pre-Processing

The project was started by uploading the data from the Project Glutenberg Website and proceeded to remove the gutenberg license.

Proceeded by cleaning the formatting a little to ensure:

- Not full of empty lines
- Evenly spaced lines
- Ready for splitting into even chunks

Tokenizing the Text for Question Answering

Since transformers cannot read the whole book at once (too long), we split the book into small pieces that the QA model can search. This step is required for a QA system.

Transformer Architecture & QA Pipeline

Loading the Pretrained QA Model

The *deepset/roberta-base-squad2* was chosen because it is specifically trained for extractive question answering, it performs extremely well on long text when chunked, it is lightweight and

fast (other models like BERT-large and RoBERTa-large are very slow on colab), and because it works well directly with HugginFace's pipeline API.

Creating the QA Pipeline

The tokenizer and model using HuggingFace's *AutoTokenizer* and *AutoModelForQuestionAnswering* classes were loaded. Then, they were wrapped up in a *pipeline("question-answering")*, which handles tokenization, running the model, and extracting answer spans. This allows me to ask a question and pass in a text passage, and the pipeline returns the model's answer in one step.

Connecting the Pipeline to Chunks

Since transformers cannot process the entire book at once, the novel was split into smaller text chunks. The book was split into 128 chunks, each around 350 tokens. For each user question, the QA pipeline is run on every chunk. The system compares the confidence scores across all chunks and returns the answer with the highest score. This lets the model answer questions about long documents even though it can only process limited text at a time.

We started by testing it with a real question from the book and the model answered with a low confidence of 0.053 (very low). This tells us the model did scan the entire book and it did find something but the chunk retrieval wasn't optimal. Since the confidence is low, the answer is unreliable.

Because the question is answered in chunk 0, but the model must search through all 128 chunks, Sometimes the question wording isn't close to the book working, so the model doesn't know which chunk is best. So we change the questions to something more relevant.

Initial question asked:

- "Where was Alice sitting at the beginning of the story?"

In the book, the relevant line is:

- "Alice was beginning to get very tired of sitting by her sister on the bank..."

So, to fix this, before we run QA, we first choose which chunk is most relevant based on keyword matching. This will improve accuracy by 2x-10x.

After adding retrieval, the system returned the correct answer ("her sister on the bank"), although with a lower confidence score (0.015). This is normal for long-document QA, even when the answer is correct, confidence scores decrease because the model is evaluating longer, noisier narrative text rather than short passages like those it was trained on. The important outcome is that retrieval improved accuracy by directing the QA model to the correct chunk.

Model Hyper-parameters tuning

Chunk Size

Now, chunk size, retrieval top_k, and chunk overlap were used to run actual tests and record the results we will be tuning.

The following chunk sizes were used for the first run:

- 200 tokens
- 300 tokens
- 400 tokens

The goal of testing different chunk sizes is to find a balance. Smaller chunks can be more specific and have a higher confidence, but sometimes missing context. And larger chunks can have more context and a lower confidence, but sometimes more correct.

Chunk Size	**Number of Chunks**	**Answer**	**Score**
200	223	her sister on the bank	0.0366
300	149	her sister on the bank	0.0284
400	112	her sister on the bank	0.0142

All three chunk sizes returned the correct answer, but the scores dropped as chunks got bigger. So, confidence score decreases as chunk size increases and a chunk size of 200 tokens performed best for this book.

top_k

Now, top_k controls how many of the most relevant chunks (based on keyword overlap) we pass to the QA model.

- If top_k is too small, the model might miss the right chunk
- If top_k is too big, the model becomes slower with more noise and a lower confidence

So we test the following values to see which gives the best performance using the best chunk size found (200):

- 3
- 5
- 10
- 15

top_k	answer	score
3	her sister on the bank	0.037
5	her sister on the bank	0.037
10	her sister on the bank	0.037
15	her sister on the bank	0.037

Now, since the answer to the question is in chunk 0, chunk 0 has an extremely high keyword overlap with the question. So no matter what top_k value we choose (3, 5, 10, 15), the retrieval function always ranks chunk 0 as # and the QA model always finds the answer in chunk 0.

This shows the retrieval system is stable, consistent, non sensitive to irrelevant chunks, and correctly identifies the location of the answer.

Chunk Overlap

Moving on, chunk overlap means that when creating chunks, the next chunk starts before the previous chunk ends (creates an overlap). This helps because sometimes an answer crosses a boundary between chunks.

overlap	num_chunks	answer	score
0	133	a—I'm a—"	0.031
25	152	her sister on the bank	0.018
50	177	a—I'm a—"	0.031
75	213	her sister on the bank	0.018

With overlap 0 and 50:

- The boundaries shift in a way where the text around Alice's introduction lands inside as earlier chunk
- That chunk includes irrelevant dialogue
- The QA model gets confused and picks that sentence
- Confidence is higher (0.03) because the chunk is more compact (fewer tokens)

With overlap 25 and 75:

- The boundaries shift differently
- The true answer ends up inside the top retrieved chunk
- Those chunks are larger, so the QA model sees more noise
- Confidence is lower (0.018) because extra text hurts precision

A moderate overlap of 25 tokens provided the best overall balance, ensuring the correct chunk is retrieved without excessively increasing the number of chunks.

Results Obtained

After tuning the hyperparameters (chunk size = 200, top_k = 5, overlap = 25), the final QA system was evaluated using several questions from Alice's Adventures in Wonderland. These questions were selected to test a mix of simple factual details and more ambiguous narrative moments.

Question	**Answer**	**Score**
Who did Alice follow into the rabbit hole?	her sister	0.076
What was the White Rabbit looking at when Alice first saw him?	peeping anxiously into her face	0.029
Who are you?	Bill	0.236
Who stole the tarts?	The Knave of Hearts	0.869
Who is the Queen in the Queen of Hearts scene?	The Knave of Hearts	0.068

The model was able to answer some questions correctly, especially when the answer appeared in a clear and distinct sentence in the book. For example, for the question "Who stole the tarts?", the system returned "The Knave of Hearts" with a confidence score of 0.868, which is the highest score out of all questions, showing that the system performs well when the retrieval step identifies the correct passage.

However, some questions were answered incorrectly. This mainly happened because the retrieval step selected chunks that contained repeated character names or nearby context, instead of the exact sentence answering the question. As a result, the QA model produced answers from those mismatched chunks. This behavior is expected for long document QA systems since transformers do not read the entire book at once, and rely entirely on the retrieved chunks.

Analysis of Results

The final results show that the QA system works well when the question directly matches a clear line in the book. For example, the model correctly answered "Who stole the tarts?" with high confidence.

However, most incorrect answers happened because the retrieval step selected chunks that were related to the question but did not contain the exact sentence needed. This caused the model to extract answers from the wrong context, especially in scenes where several characters are mentioned together. The system also struggled when the question wording did not match the book's phrasing.

Overall, the model performs best on simple factual questions but becomes less reliable when retrieval selects overlapping or partially relevant chunks.

Conclusion

This project demonstrated how a transformer-based QA system can be built using a pretrained model and applied to a full novel from Project Gutenberg. Through preprocessing, chunking, retrieval, and hyperparameter tuning, the system was able to answer several questions from Alice's Adventures in Wonderland.

The results show that the model performs well when the retrieval step identifies the correct passage. Clear, direct questions with unique answers produce the most accurate results. At the same time, the project also highlights the limitations of long-document QA. When retrieval selects partially related text or when the question wording differs from the book, accuracy drops.

Overall, this project shows that transformers can handle question-answering tasks on long texts when combined with chunking and retrieval, but their performance depends heavily on how the input text is divided and how well the retrieval method identifies the correct context. Despite these limitations, the system still demonstrates the effectiveness of modern NLP models for extracting information from large bodies of text.