Automation of PowerPoint Presentation Generation Process

Uroš Poček

1. Motivation

One of the common tasks in primary and secondary schools is creating a presentation on a given topic. Students need to research the presentation subject, find appropriate text and images, and incorporate all of this into a PowerPoint presentation that they will present at school. The most challenging part of this process for students is determining which areas to cover in the given time and finding suitable resources. This often presents a challenge, as students can be overwhelmed by the amount of available information or have difficulties in assessing the relevance and reliability of sources.

Automating this process using natural language processing techniques can significantly ease this task for students. This is not just a matter of saving time, but also of improving the quality of education. An automated system can help students focus on understanding and analysing information, rather than spending time on mechanical data collection.

Moreover, automating the process of creating presentations can be beneficial for teachers as well. They could prepare teaching materials more quickly and efficiently, leaving them more time for interaction with students and individual approach. In the long term, this technology could lead to a more personalised approach to learning, where content is automatically adapted to each student's level of knowledge and interests.

2. Research questions

The problem I wanted to tackle is the automation of generating the first draft of a school presentation related to a given topic. Creating a presentation from scratch can be a time-consuming task for students. It involves extensive research, identification of key subtopics, and summarization of relevant resources. Automating this process can significantly speed the creation of the first draft, allowing students to focus more on understanding and refining the content.

To achieve this, an automated system needs to be capable of performing several critical functions. First, it must be able to parse the main topic and identify the most important subtopics within it. This requires advanced natural language processing (NLP) techniques to analyse the topic, break it down into its constituent parts, and determine which subtopics are essential for a comprehensive understanding.

Once the key subtopics are identified, the system should then find reliable and relevant resources that provide in-depth knowledge about each subtopic. The system must then summarise these resources effectively, extracting the most important information.

Since solving this problem requires several machine learning models and natural language processing techniques, I collected and annotated the appropriate data for training each model, if the task requires it. The data was gathered in a manner consistent with how it is retrieved during model evaluation on new topics, specifically by scraping relevant articles from the internet where allowed. A Python script was developed to perform the scraping, and the retrieved data is stored in a text file on disk.

By automating these steps, the goal is to create a tool that can generate a well-structured and informative first draft of a school presentation quickly and efficiently. This not only saves time for students but also helps them to start with a solid foundation, upon which they can build and expand their presentation further.

3. Related work

There are several competitors to our product in the market. One of them is https://tome.app/, which uses LLMs and a custom presentation platform to create presentations. They recently raised $43 million, so they are doing well, but our approach is different in technology and integration with existing solutions. We choose to use Transformers for summarization and deciding what content to use in presentations, while they use LLMs. Also, they developed their own presentation standard that doesn't support PowerPoint export natively. We decided not to reinvent the wheel and instead work with PowerPoint from the beginning, respecting people's habits with PowerPoint presentations.

Another competitor is https://www.slidesai.io/. They have a different approach from us and Tome. They focus more on design, not on research and text summarization. We have a separate model that determines which category the requested theme belongs to and chooses a presentation theme accordingly, but our design is not as adaptable and diverse as SlideAI.

4. Methodology

To accomplish this task, I needed to perform several key steps. First, we gather relevant resources for a given topic. Currently, we are using the Wikipedia API, but this can be expanded to include other resources and scientific papers in the future. Once we have the raw text data returned by the Wikipedia API, I organise it into a tree-like structure to make it easier to process in the subsequent steps.

Next, I pass this tree structure to an importance classifier model, which determines the significance of each leaf or subtopic in relation to the main topic. Based on the user-specified number of slides, the model filters out the least important subtopics. I trained this classification model on over 500 samples of data that I manually annotated. The dataset structure is as follows: "topic subtopic|rank from 1 to 3." For example, "Universe Definition major subject|1."

Once we know which subtopics to include in the presentation, the next step is to summarise the texts so they fit onto the presentation slides. For this task I choose sshleifer/distilbart-cnn-12-6 pre-trained model. Besides this model I tested with my family text summarised by other larger or smaller hugging face models and by that heuristic we decided that this model provides best summarization readability and coherence for time it takes to make the summarization on average hardware. Bigger and more sophisticated models give slightly better summarizations understanding topic better and putting into summary some information that were not in text give for summarization, but they also took 2-3 times more time to perform summarization and required my more server resources which will be impractical for me and my end users in production environment. I also tested different summarization lengths and decided to use a summary between 100 and 120 words in production. Currently, we allocate one slide per subtopic, but this could be improved with more sophisticated heuristics.

The final step is to determine the style of the presentation and select an appropriate template. I have prepared over 30 presentation templates in various categories, and my classification themes model selects the template based on the topic. I trained this model on over 550 samples with the structure: "topic|category." For example, one instance in the dataset used to train this model is "Economics|Other.". Those are themes I supported: themes = {0: "Language", 1: "Geography", 2:"History", 3: "Science", 4: "Biology", 5: "Sport", 6: "Technology", 7:"Business", 8: "Space", 9: "Art", 10: "Other"}. For this model training as for the important one I used bert-base-uncased and appropriate tokenizer from hugging face. I optimised hyper parameters for learning rate and batch size and, but model default recommended worked best: # Set up parameters bert_model_name =

'bert-base-uncased' num_classes = 11 max_length = 128 batch_size = 16 num_epochs = 4 learning_rate = 2e-5.

With all the components in place, we can create the first draft of the PowerPoint presentation. We use an open-source library to manipulate PowerPoint templates and insert data into the appropriate areas. We also use the Google Images API to add relevant images to the slides.

Finally, because this is a B2C application, we packaged all these models into an app developed in Django, making it accessible for everyone to use.

5. Discussion

To evaluate the performance of each model, I used different metrics on the test set. For the importance classifier, I used Precision, Recall, and F1 score. These are the results I got:

{'precision': array([0.28658537, 0.47598253, 0.41880342]),
'recall': array([0.34306569, 0.545 , 0.28323699]),
'f1_score': array([0.31229236, 0.50815851, 0.33793103])}

Interpreting these results shows that the model is struggling to correctly classify the importance of subsections for a given topic and is only slightly better than random guessing. This tells me that just information from the topic and name of the subsection is not enough for learning. To improve my approach, I would use a bigger model that has more understanding of language in general, and also include summary or first paragraph as input. This way, the model would have more information to work with and better determine importance for the requested topic.

Next, I evaluated the summarization model using the ROUGE metric (Recall-Oriented Understudy for Gisting Evaluation). I tested performances of that model for my use case and got these results:

ROUGE scores for topic Universe:
ROUGE-1: Score(precision=0.7587143503847895, recall=0.6404279709591135, measure=0.6945710733526731)
ROUGE-2: Score(precision=0.3677536231884058, recall=0.3103975535168196, fmeasure=0.33665008291873966)
ROUGE-L: Score(precision=0.3983703033046627, recall=0.3362628964463126, fmeasure=0.36469125569830085)
ROUGE scores for Olympic Games:
ROUGE-1: Score(precision=0.3255968169761273, recall=0.37523882307986245, fmeasure=0.3486596840049707)
ROUGE-2: Score(precision=0.058374792703150914, recall=0.0672782874617737, fmeasure=0.06251109927188778)
ROUGE-L: Score(precision=0.133289124668435, recall=0.15361100496752006, fmeasure=0.14273033907331795)

As the ROUGE metric is calculated by comparing model summarization with my own summarization, this is subjective to what I found important in those two selected topics.

Lastly, for evaluation of my theme/category classification model, I measured Accuracy, Precision, Recall and F1-score. These are my results:

{'accuracy': 0.5527272727272727,
'precision': array([0.71186441, 1. , 0.9375 , 0.73913043, 0.79365079, 1. , 0.4159292 , 0.31055901, 1. , 1. , 0.07692308]),
'recall': array([0.84, 0.02, 0.9 , 0.68, 1. , 0.18, 0.94, 1. , 0.16, 0.32, 0.04]),

'f1_score': array([0.7706422 , 0.03921569, 0.91836735, 0.70833333, 0.88495575, 0.30508475, 0.57668712, 0.47393365, 0.27586207, 0.48484848, 0.05263158])}

As a reminder, these are 11 categories that correspond to these results for mentioned metrics:

themes = {0: "Language", 1: "Geography", 2: "History", 3: "Science", 4: "Biology", 5: "Sport", 6: "Technology", 7: "Business", 8: "Space", 9: "Art", 10: "Other"}

From this, we can see that accuracy for this multi-class classification problem is relatively good. Precision across most topics is great. The main problem that is driving other metrics down is for category "Other". Even though my dataset was balanced, the model is struggling to identify when it should classify some topic as "Other". I think this is because there isn't similarity between topics now labelled as "Other". To be honest, I prefer my model to perform this way and try to classify topics in some class, rather than to be uncertain about all of them and to favour classifying anything as "Other". The same pattern repeats for 'recall' and 'f1-score', so I draw the same conclusions as described in the case study of accuracy.

The training, evaluation and experiments I used were run on my MacBook Pro with 16GB of RAM and M2 chip, so they may vary on different architectures and platforms. All of my models, Django web server and presentation templates are available for free use to everyone on https://np.bwl.rs/ .

6. References

https://huggingface.co/ 29.06.2024.

https://medium.com/@khang.pham.exxact/text-classification-with-bert-7afaacc5e49b 29.06.2024.

https://medium.com/@eren9677/text-summarization-387836c9e178 29.06.2024.