WhatsApp Chat Summarization

Andrea Auletta
andrea.auletta@studenti.unipd.it

Davide Baggio
davide.baggio.1@studenti.unipd.it

Marco Bernardi
marco.bernardi.11@studenti.unipd.it

Marco Brigo
marco.brigo@studenti.unipd.it

Sebastiano Sanson
sebastiano.sanson@studenti.unipd.it

Abstract

This research project in Natural Language Processing (NLP) focuses on the development of a system for the summarization of chat messages, applicable to both group and individual conversations. The objective is to generate concise summaries of conversations, thereby obviating the need for users to review all messages individually. This application is especially pertinent in contexts where chat logs proliferate quickly, offering significant benefits in terms of time savings and cognitive load reduction for users.

1. Introduction

In the era of digital communication, chat platforms have become fundamentals in our daily interactions, whether they are personal or professional. This new type of transmission favors real-time messages exchange which allows users to engage in conversations that can quickly accumulate extensive chat logs. The large volume of messages, especially in active group chats formed by dozens or even hundreds of users, often makes it difficult for users to stay updated without dedicating significant time to review each message. This issue highlights the need for an efficient method to quickly extract essential information from chat conversations.

We aim to address this need by developing a system for summarizing chat messages, with the goal to generate concise summaries and reducing the need to sort through all messages. By doing so, users can quickly obtain the main points of discussions, significantly saving time and reducing cognitive load. The utility of this application is evident in various contexts. For instance, in corporate environments, project teams often heavily rely on group chats to coordinate tasks, share updates and exchange miscellaneous informations. Similarly, social groups frequently engage in dynamic conversations, discussing a wide range of topics, planning events, and sharing personal updates. In both scenarios, an automated summarization system can enhance productivity and ensure important information is not overlooked.

2. Datasets

In this section, we explore datasets that are essential for training and evaluating automated summarization systems for chat messages. An ideal dataset should have a variety of conversation styles and contexts to ensure the robustness and versatility of the summarization models. The datasets should contain real-life or realistic dialogues annotated with concise and accurate summaries. Additionally, the data should reflect diverse communication patterns, including informal and formal registers, and incorporate common conversational elements such as slang, emojis, eventual typos, voice messages and images. We have used SAMSum (obtained by the Hugging Face repository), a great dataset which fulfill these criteria and provide rich resources for advancing the field of conversational AI.

2.1. SAMSum

SAMSum [2] dataset contains about 16,000 messenger-like conversations with summaries splitted in:

- training: 14,732;
- validation: 818;
- test: 819.

Conversations were created and written down by linguists fluent in English: they were asked to create them similar to those they write on a daily basis, reflecting the proportion of topics of their real-life messenger conversations.

The style and register are diversified: conversations could be informal, semi-formal or formal, they may contain slang words, emoticons and typos. Then, the conversations were annotated with summaries. It was assumed that summaries should be a concise brief of what people talked about in the conversation in third person. The SAMSum dataset was prepared by Samsung R&D Institute Poland and is distributed for research purposes.

2.2. Testing dataset

To assess the performance of the selected model within our specific use case, we constructed a testing dataset by generating new examples of WhatsApp-like chat.

2.2.1 Data collection

WhatsApp provides a robust functionality for exporting chat conversations in a `.txt` file format, which encompasses all exchanged messages with timestamps and sender names. Additionally, the file includes various media types, such as images, voice messages, and videos. This comprehensive dataset is instrumental for rigorous evaluation as it reflects real-world conditions and the diverse communication forms encountered in practical applications.

2.2.2 Data preprocessing

The exported `.txt` file underwent a meticulous preprocessing phase to extract the pertinent information, including message content and sender details. This information was structured into a Python dictionary as follows:

```
dialogues = [
    {
        'text': "Hello, how are you?",
        'id': 0,
        'golden_summary': "A greeting"
    }
]
```

For conversations containing images or voice messages, the filenames of these media were embedded within the text field to facilitate subsequent processing by replacing the filenames with detailed descriptions for images and transcriptions for voice messages. The models utilized for generating these descriptions and transcriptions are discussed in the following sections. We also scraped out irrelevant information, like the date-time, in order to match the same structure used in the training dataset mentioned in the previous sections.
This preprocessing approach ensures that the dataset is both comprehensive and structured, enabling effective analysis and model evaluation.

3. Models involved

3.1. BART

BART [5] is a denoising autoencoder that maps a corrupted document to the original document it was derived from. It is implemented as a sequence-to-sequence model with a bidirectional encoder over corrupted text and a left-to-right autoregressive decoder.
In the architecture there are 6 layers, both in the encoder and decoder (while the large model uses 12 layers in each). Each layer of the decoder performs cross-attention over the final hidden layer of the encoder. This pretraining enables BART to effectively learn robust representations of text.
BART is trained by corrupting documents and then optimizing a reconstruction loss between the decoder's output and the original document.
The part that interests us is the fine-tuning: this model can be used in several ways for downstream applications like sequence classification, token classification, sequence generation and machine translation tasks.

3.2. FLAN-T5

FLAN-T5 [1] (Finetuned LAnguage Net) is an enhanced version of the T5 model (Text-To-Text Transfer Transformer) that has been fine-tuned on more than 1,000 additional tasks, aiming to improve the performance on natural language understanding and generation tasks. It comes in various sizes, from 80 million parameters (Flan-T5-Small) to 11 billion parameters (Flan-T5-XXL). Due to our limited computational resources, we have chosen the base model, which includes 248M parameters.
The T5 model follows the Transformer architecture's encoder-decoder structure, which is crucial for handling text-to-text tasks and is characterized by:

- Encoder: processes the input embeddings and generates a sequence of hidden states. Each layer of it consists of a multi-head self-attention mechanism followed by a feed-forward neural network. The former allows the model to focus on different parts of the input sequence when encoding each token, while the latter consists of two linear transformations with a ReLU activation in between;

- Decoder: takes these hidden states and generates the output text, one token at a time, using autoregressive generation. It's similar to the encoder, but it includes also an additional layer of attention over the encoder's outputs to ensure that the prediction for a particular token can only depend on the known outputs before it.

Finally, the output is generated by a softmax function that converts the decoder's hidden states into probabilities reflected on the vocabulary in natural language.

The comparison between T5 and FLAN-T5, both in the base version, highlights how the fine-tuned model outperforms the former one in every relevant benchmark for our scope. Indeed, as shown in the model's paper, it performs significantly better in a wide range of tasks (like summarization, translation, Q&A), subjects and disciplines, proving that it understands and generates human language in a superior way.

3.3. Multimedia models

Because chat messages often include multimedia contents, we have decided to use specific models to convert this type of data into text that can be summarized by the previously cited model. In particular, we transform images and voice messages - which are very frequently used nowadays - while avoiding GIFs and videos due to their high computational demands. Since our project regards text chat summarization, we will not focus into the technical details of how the following models work in this paper. This approach ensures that our main objective is clearly addressed without overcomplicating the discussion with the intricate workings of the underlying models.

3.3.1 Florence-2 Large

To ensure that the images were processed by the summarization model, it was necessary to convert them into detailed textual descriptions. For this task, we employed Florence-2 by Microsoft [4], an advanced vision foundation model that utilizes a prompt-based approach to address a wide array of vision and vision-language tasks. Florence-2 is designed to interpret text prompts as task instructions and produce the desired textual outputs, including captioning, object detection, grounding, and segmentation. This model is built upon the extensive FLD-5B dataset, which comprises 5.4 billion annotations across 126 million images, thereby enhancing its multi-task learning capabilities. The sequence-to-sequence architecture of Florence-2 allows it to perform effectively in both zero-shot and fine-tuned scenarios, establishing itself as a robust and competitive vision foundation model.

3.3.2 Whisper

In addition to processing images, it was necessary to convert voice messages to text. For this task, we utilized Whisper by OpenAI [3], an automatic speech recognition (ASR) system. Whisper is trained on 680,000 hours of multilingual and multitask supervised data collected from the web. The extensive and diverse dataset enhances the system's robustness to accents, background noise, and technical language, which is particularly beneficial for our use case, as voice messages are often of suboptimal quality. Furthermore, Whisper supports transcription in multiple languages and translation from those languages into English.

4. Metrics

4.1. ROUGE

We evaluated the results provided by different NLP models using the ROUGE set of metrics, which stands for Recall-Oriented Understudy for Gisting Evaluation. We chose this specific metric because it is currently the most effective way to assess the performance of automatic summarization systems. It compares the generated summary with one or more human made references by measuring the overlap of n-grams in them.
In particular, we have used the following metrics:

- **ROUGE-1**: as the number suggests, it evaluates the overlap of unigrams, meaning that it compares the single words between the summaries;
- **ROUGE-2**: this time it evaluates the overlap of bigrams, that are two consecutive words;
- **ROUGE-L**: this one instead evaluates the longest common subsequence, meaning that it considers the structure of the sentences and the word order, but not requiring the words to be consecutive;
- **ROUGE-LSum**: it is related to the previous one, but it uses a slightly different calculation method since it applies the ROUGE-L's method at the sentence level and then aggregates all the results for the final score. This metric is seen as more suitable for tasks where sentence level extraction is valuable such as extractive summarization tasks.

Each metric is calculated as follows:

- **Precision**: it measures the accuracy of the positive predictions made by the model. It is the ratio of correctly predicted positive instances to the total predicted positive instances

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

- **Recall**: it measures the ability of the model to find all relevant positive instances in the dataset. It is the ratio of correctly predicted positive instances to the total actual positive instances.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

- **F1-Score**: is the harmonic mean of precision and recall. It provides a single metric that balances both precision and recall, which is useful when the dataset

has an uneven class distribution or when the cost of false positives and false negatives is different.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The value returned for each ROUGE metric is the F1-Score, since it provides a balanced measure of a model's performance by combining both precision and recall. By doing so, it avoids scenarios where a model might have high precision but low recall values, or vice versa. This is especially important when both false positives and false negatives carry significant costs.

4.2. Human evaluation

The group evaluated the quality of each generated summary by assigning a vote: 1 if the summary was deemed good and 0 otherwise. The final score for each summary was determined by summing the votes from all team members.

5. Methods

After reviewing the state-of-the-art literature on the summarization task, we decided to proceed as follows:

- chose abstractive over extractive summarization because the former provides more human-like and fluent summaries;
- selected small-sized models, based on our study of state-of-the-art models, due to computational cost considerations;
- fine-tuned the models with the SAMSum dataset, as it best suited our needs in terms of dialogue structure.

6. Fine tuning

6.1. BART hyperparameters selection

We started by using BART, where we selected BART-large-cnn over BART-large-xsum available on Hugging Face by Meta. We opted for that model because it generates longer summaries that retain more information, which is essential for our task, whereas BART-large-xsum tends to produce shorter, more concise summaries that may not contain important details from the dialogues. We monitored the training loss and the validation loss of the model through the logging hyperparameters specified in the Seq2SeqTrainingArguments. While experimenting with the model we kept a frequent logging to monitor the progress of the losses. We started by changing the learning rate, noticing that a slight increase from the default value improved performance. However, using an higher learning rate like 1e-03 or 1e-04, caused the loss functions to increase. After noticing this behaviour, we decided to fix the learning rate and experiment with the other hyperparameters. We decided to experiment with the `num_train_epochs`, gradually increasing it, in order to put more epochs to help the model learn better keeping an eye on avoiding overfitting. We noticed that the training loss decreased significantly and then stabilized at each epoch, while the validation loss decreased at the beginning but then remained relatively constant. We decided to set the number of epochs to 3 to ensure the model learns the important structure of the data. Then, we increased `per_device_train_batch_size` and `per_device_eval_batch_size` to have a more stable gradient estimates, even though it required more memory, making the most of Colab free resources. Using higher batch sizes also decreased the validation loss and improved the Rouge metrics. Knowing the Colab free limitations we decided to exploit the `gradient_accumulation_steps` hyperparameter in order to increases the batch size without requiring more memory, as explained in the notebook. As a final step, we decided to set `fp16=True` in order to speed up training and reducing memory usage, allowing us to use larger batch sizes.

6.2. FLAN-T5 hyperparameters selection

After obtaining almost good results by tweaking all the possible hyperparameters, we decided to test another model, T5 by Google, which we discovered while researching state-of-the-art models for abstractive summarization. The function used in T5 for preprocessing the data uses the tokenizer, but here we decided to avoid accounting for padding in the loss and to add, as stated in the model's paper, the prefix for the summarization task. After this, we used the same approach to identify the best combination of hyperparameters while also monitoring the ROUGE metrics during each epoch. This time, differently from our approach with BART hyperparameters selection, we decide to set `fp16=False`, since the FLAN-T5-base has less parameters than BART-large-cnn, and so it requires less memory usage. In the end, we chose to pick the FLAN-T5 model as the final fine-tuned model, as it achieved the best ROUGE metrics during the testing phase at the conclusion of the notebook.

6.3. Results

Before fine-tuning the two selected models, we conducted a test to obtain the ROUGE metrics for both of them. The results are as follows:

Model	R-1	R-2	R-L	R-LSum
BART	0.30	0.10	0.23	0.23
FLAN-T5	0.45	0.21	0.37	0.37

Table 1: ROUGE metrics before fine-tuning

After succesfully completing the fine-tuning on the SAMSum dataset, we obtained the following results:

Model	R-1	R-2	R-L	R-LSum
BART	0.43	0.21	0.33	0.33
FLAN-T5	0.47	0.23	0.37	0.37

Table 2: ROUGE metrics after fine-tuning

As observed, BART demonstrated the greatest improvements while FLAN-T5 achieved the highest ROUGE values, resulting in our final model. The fine-tuned models have been uploaded to the following Hugging Face's repositories:

- BART: `https://huggingface.co/Seba213/bart-large-cnn-samsum`
- FLAN-T5: `https://huggingface.co/Seba213/flan-t5-base-samsum`

7. Testing

After completing all the steps mentioned before, we assessed the practical performance on the custom dataset by voting on whether the generated summaries met our expectations, as detailed in the Human evaluation section. The results are shown in the following table, where the first row represents the chat ID and the second row represents our quality rating for that summary:

1	2	3	4	5	6	7	8	9	10	11	12
2	5	0	4	5	2	0	5	5	5	5	1

Table 3: Results of the human evaluation

Furthermore, we calculated the ROUGE scores for all the examples in the custom dataset and obtained the following results by averaging the values.

ROUGE-1	ROUGE-2	ROUGE-L
0.41	0.18	0.39

Table 4: ROUGE metrics on the custom dataset

Here is an example of the model's output on a chat from our custom dataset:

- Golden summary: In the group chat, Mia asks the group about plans for tonight. Alex suggests dinner and a movie, and Chris agrees.They decide to meet at 7 PM in an italian restaurant. For the movie, they choose Tenet, and agree on booking tickets in advance for the 9 PM show.
- Model summary: Mia, Alex, Chris and Taylor will meet at the Italian restaurant downtown at 7 pm for dinner and then a movie. They are going to see the new action movie Tenet at 9 pm. Mia will book 5 tickets for the 9 pm show.

8. Conclusions

In conclusion, we successfully developed a specialized model capable of summarizing chat dialogues and integrating textual representations of images and voice messages. The main challenges we encountered were the limited resources provided by Google Colab free tier, which prevented us from utilizing larger and better performing models, and the dependency on image conversion since the model used did not always provide accurate descriptions due to hallucinations.
We achieved all the objectives outlined in our work plan, including the non-mandatory tasks.

N	Task	Mand	Status
1	Study of the state-of-the-art	Yes	Completed
2	Dataset colletion	Yes	Completed
3	Data preprocessing	Yes	Completed
4	Summarization of text messages	Yes	Completed
5	Summarization of images	No	Completed
6	Summarization of voice messages	No	Completed
7	Test and evaluation of the model	Yes	Completed ROUGE: 0.416 > 0.40
8	Documentation	Yes	Completed

Table 5: Deliverables status

Additionally, we got really close to the time schedule proposed in the work plan as shown in the following table.

Task	Aul	Bag	Ber	Bri	San	Tot
1	5	5	5	5	5	25
2	5	3	3	2	2	15
3	3	4	3	0	0	10
4	17	16	16	21	22	91
5	2	5	5	2	1	15
6	3	3	5	2	2	15
7	10	10	10	15	15	60
8	5	5	5	4	4	23
Tot	50	51	52	51	51	255

Table 6: Time schedule

References

[1] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022.

[2] Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *CoRR*, abs/1911.12237, 2019.

[3] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022.

[4] Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks, 2023.

[5] Mike Lewis Yinhan Liu Naman Goyal Marjan Ghazvininejad Abdelrahman Mohamed Omer Levy Ves Stoyanov Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019.