Closing the Gap with Idioms and LLMs
A Hybrid Approach for Abstractive Dialogue Summarization

Samuele Vanini
Politecnico di Torino
s318684@studenti.polito.it

Susanna Olivero
Politecnico di Torino
s334303@studenti.polito.it

Hadi Ibrahim
Politecnico di Torino
s313385@studenti.polito.it

***Abstract*—Abstractive Dialog Summarization involves generating concise summaries from chat dialogues, capturing the essence and critical points of the conversation. The architecture we took as a reference, SICK/SICK++, is built around the novel approach of commonsense injection; this paper aims to enhance it through two experimental approaches. The first approach focuses on improving performance by addressing idiomatic expressions within dialogues and augmenting the datasets with additional relevant data. The second approach involves modifying the base architecture and investigating the application of large language models to refine summarization capabilities further. Our experimental techniques demonstrate competitive performance compared to the reference ones, indicating the effectiveness of our enhancements.**
The code of the work can be found at the following link here [1].

I. Introduction

The Natural Language Processing (NLP) task addressed in this paper is abstractive dialogue summarization, which involves generating a concise summary from a chat dialogue between different speakers. Unlike traditional document-to-document summarization, such as for articles and scientific publications, dialogue-to-document summarization faces challenges due to the differences in input and output formats, making it harder to learn the mapping patterns. Summarizing dialogues presents different challenges. First of all, the model has to capture the context of a conversation, which often involves understanding the nuances and implicit meanings behind the speakers' words. Detecting unspoken intentions and hidden meanings is essential for the general understanding of the dialogue. The model should also correctly identify and attribute statements to the respective speakers and create a coherent and fluent summary, even if the dialogue itself may be fragmented and unclear. Working with dialogue means that the model has to deal with informal language, slang, abbreviations, and figures of speech, such as idioms.
The starting point of our work is the paper "Mind the Gap" [1], where the authors integrated commonsense knowledge to enhance dialogue summarization. They used COMET [9], a commonsense knowledge model that generates social interaction inferences (e.g., Xintent, Xwant) and event-centered inferences (e.g., Xreason, Xneed). Their framework injects commonsense knowledge into the chat dialogue before producing the summaries. With this research, we propose an extension to their work and an alternative solution, but in both cases, we start by utilizing the datasets with the injected commonsense knowledge.

[1]https://github.com/hadi-ibra/SICK_Summarization

The first extension we propose aims to improve the models suggested by S. Kim et al. [1]. It involves detecting and augmenting idiomatic expressions within dialogue datasets, providing the model with additional context to better understand and summarize the intended messages in dialogues. Idioms are a significant component of natural language, often used to express complex ideas succinctly and colorfully. However, summarization models can overlook their figurative meanings, potentially leading to inaccurate or incomplete summaries.
Since abstractive dialogue summarization is a real challenge for traditional language models, we chose to explore the usage of Large Language Models (LLMs) for our second experimentation. The LLMs have exploded in the last few years, becoming increasingly widespread across various domains and NLP applications. We decided to use them because they are renowned for generating good human-like text and are characterized by expansive pre-trained knowledge, and so they should capture the nuances and implicit meanings behind the speakers' words more easily. Additionally, while the SICK framework works with only one summary at a time, the prompting method allows us to provide the LLM with multiple examples, each consisting of a gold summary corresponding to the input dialogue. To conclude our experimentation, we tried combining the two extensions. We performed prompting on our LLM using dialogues where we added the idioms' meaning as additional context.

II. Related Works

A. *Abstractive Dialogue Summarization*

The reference paper [1] introduces two innovative architectures, SICK and SICK++, that enhance dialogue summarization by incorporating commonsense knowledge into the process. Compared to other popular models, the illustrated framework presents competitive results.
The principal architecture, SICK, consists of four main components: a commonsense knowledge generator, a commonsense picker, a dialogue encoder, and a summary decoder. SICK++ further refines this architecture by leveraging an additional decoder and a combined loss function to force the summarization decoder to encode the injected commonsense better. The base model used for commonsense knowledge generation

is COMET, but the authors have also used an improved version called ParaCOMET. This discourse-aware model incorporates paraphrasing capabilities, allowing the model to generate more diverse and contextually appropriate inferences. This latter extraction model is the one we have used, as it performs better, as shown in [1].

B. *Idioms Elaboration*

The study of idiomatic expressions has garnered significant attention in recent years, with several datasets and methodologies being developed to facilitate the detection and analysis of idioms in natural language.
Notably, the work of Haagsma et al. [6] presents a comprehensive corpus, the MAGPIE dataset, that identifies and categorizes potentially idiomatic expressions across various contexts. Similarly, the EPIE Dataset [5] offers an extensive collection of idiomatic phrases. These datasets are instrumental in examining the variability and usage patterns of idioms across different domains and genres.
In terms of computational approaches, the work by Wang et al. [11] showcases a state-of-the-art methodology for idiom detection. Utilizing pre-trained language models, this study demonstrates the effectiveness of advanced neural networks in accurately identifying idiomatic expressions within text. The pipeline introduced in this work has shown considerable promise in enhancing the precision of idiom detection tasks, and we decided to use a similar one in our extension.

C. *Large Language Models*

The LLM we have chosen to utilize for our experiments is LLaMA, specifically version 7B of LLaMA2-Chat. The choice was driven by the fact that the models of the LLaMA family, with their extensive training on vast corpora, have demonstrated an unparalleled ability to capture the nuances of language and consistently outperform other open-source chat models on various benchmarks [4].
The LLaMA family, presented in [4], consists of a collection of pre-trained and fine-tuned LLMs where the range of the parameters is between 7 and 70 billion. We decided to use the version *Chat* because LLaMA2-Chat is a refined version of LLaMA2 fine-tuned explicitly to excel in dialogue use cases. Prompting is the method used to provide input to our LLM to generate desired outputs [10]. It involves giving specific instructions or queries, known as prompts, to guide the model's responses. The choice and formulation of prompts play a crucial role in shaping the model's behavior and output. There are two ways of prompting, the zero-shot and the few-shot, based on the number of examples provided as input in the prompt [10].

III. Methodology

In this section, we are going to describe the two pipelines of our extensions, that use the SICK architecture as a starting point.

A. *Idioms Detection and Augmentation*

This process was developed to annotate the datasets to test with idiom meanings. The implemented function processes each conversation, identifies idiomatic expressions using the detection model, and augments the dialogue with the meanings of the detected idioms. To better visualize how the dialog is augmented with the idiom meaning, refer to Figure 3 in the Appendix.
This annotation process involves different steps.
First of all we need an extensive list of idioms and their meanings to create a comprehensive set of idiomatic expressions. We have utilized four sources: the MAGPIE dataset [6], the EPIE dataset [5], and two additional online sources: ”EF Education First: English Idioms” [12] and ”The Phrase Finder: Idioms and Phrases” [13]. These sources collectively provided a comprehensive set of idiomatic expressions, which were essential for the augmentation process.
After consolidating the idiom datasets, we computed the embeddings for all idioms using the ”mixedbread-ai/mxbai-embed-large-v1” model. This model was chosen for its optimal balance between performance and size, ensuring efficient and accurate embedding generation. These embeddings were precomputed to facilitate quick similarity searches during the augmentation process.
Subsequently, we employed the ”idiom-xlm-roberta” model for idioms' detection in the dialogues. This model was fine-tuned on MAGPIE and EPIE datasets, to accurately detect idioms within the dialogue text, accounting for prepositions and other context-specific variations.
After computing the embeddings for each detected idiom, we perform a similarity search to find the best matching idiom from the precomputed embeddings for each detected idiom.
To conclude we augment the original dialogues by inserting the idioms meaning, thereby providing additional context for the summarization model. A conceptual schema of the pipeline is present in Figure 1.

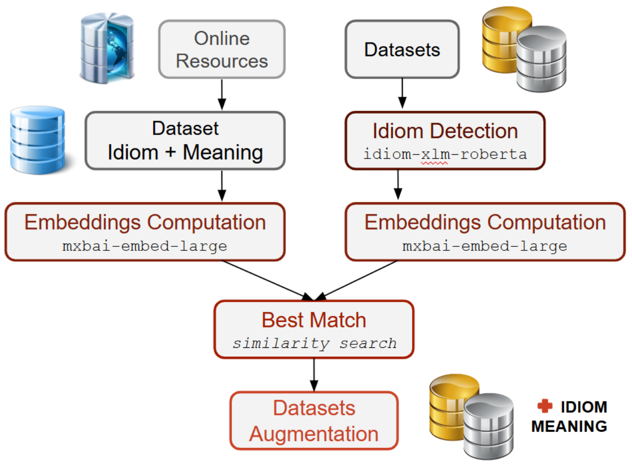

Fig. 1. Idiom Annotation Pipeline

B. Prompting on LLaMA

We use the LLM via the prompting method, which supplies input to the LLaMA model to generate desired summaries. Unlike the SICK/SICK++ frameworks, this approach does not involve any training. There are two main types of prompting: zero-shot, where the model is given a task without any examples, relying on its pre-existing knowledge, and few-shot, where we provide the model with a few examples to guide its responses. This last method strikes a balance between generality and task specificity since we give the model more context on how to solve the required task.
After a brief section on prompt engineering, in which we selected the type of prompt that proved to be the most effective and robust among those tested, we began the experiments focusing on two main variables: the number of input examples and the 'temperature' value. The latter is a parameter that controls the randomness of the output; it affects the probability distribution of the predicted tokens. A lower temperature (closer to 0) makes the model more deterministic, producing more predictable and focused outputs. Instead, a higher temperature (closer to 1) increases randomness, allowing the model to explore a broader range of possible tokens, resulting in more diverse and creative outputs.
Of course, since we prompt the dialogues presented in the test set of the datasets that we have to summarize, we used as additional input dialogues and summaries randomly taken from the train set. A conceptual schema is present in Figure 2. An important detail to consider about LLaMA is that the maximum input token length is 4096, so we can provide a few dialogue examples as input.

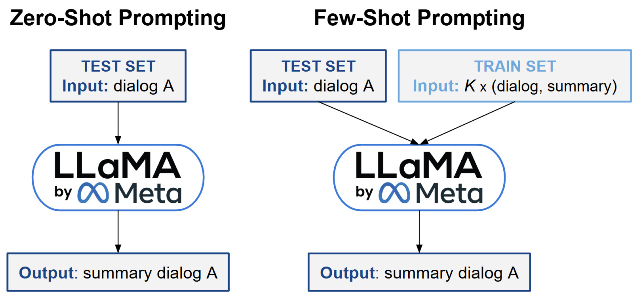

Fig. 2. Prompting Pipeline

IV. Experiments

Datasets

We conducted experiments on two datasets, SamSum [3] and DialogSum [2]. SamSum is the most commonly used resource for the abstractive dialogue summarization task, consisting of natural, messenger-like conversations in English created by linguists with manually annotated gold summaries. DialogSum, a more recent dataset, presents a more challenging task due to its lower compression ratio (ratio of the length of summary divided by the length of dialogue) and includes multi-turn dialogues from real-life scenarios collected from three different dialogue corpora. We provide some data statistics in Table I, including information about the presence of idioms. For both our extensions, we have not used the original datasets but the ones with injected commonsense extracted by the PARACOMET model described in the reference paper [1].

TABLE I
Statistics of Dialog Summarization Datasets

	SamSum	DialogSum
train set	14732	12460
validation set	818	500
test set	819	500
# tokens / dialogue	82.57	121.56
# tokens / summary	20.30	22.64
compression rate	0.35	0.20
# idioms train set	2268	2860
# idioms validation set	123	131
# idioms test set	132	103

Implementation details

The main libraries that we used are the transformers package from Hugging Face and nltk for text preprocessing. The main models utilized are from Hugging Face: *bart*, *idiom-xlm-roberta* [14], *mxbai-embed-large-v1* [16] and *Llama-2-7b-chat-hf* [15]. Other minor libraries, like *rouge_score* and *bert_score*, were used for the evaluation. The Python framework for sentence embeddings is instead 'SentenceTransformers'. For the training of SICK and SICK++ with and without the idiom extension, we used the parameters related to the best performances according to the reference paper. However, due to computational limits, we have trained all for 5 epochs instead of 20/25 epochs as in [1].
All experiments were run on Colab using one NVIDIA A100 GPU, and each took an average of 4 hours to execute.

Evaluation metrics

To evaluate the summary generated by our model against the reference summary, we used two automatic evaluation metrics: ROUGE [8] and BERTScore [7]. The ROUGE metrics assess the overlap with the gold summary. Specifically, we used ROUGE-1 (R-1) and ROUGE-2 (R-2), which compare word-level uni-grams and bi-grams, and ROUGE-L (R-L), which considers the longest common subsequence. In contrast, BERTScore (BS) computes contextual similarity by representing tokens with BERT contextual embeddings rather than relying on exact matches. We report F1 scores for all metrics.

Results

In this section, we present the results obtained, following a brief explanation of the experimental setup.
Table II compares the results obtained on SICK/SICK++ architecture, by leveraging or not leveraging the idiom information in the dataset preprocessing. We can observe that the results are similar: SICK performs slightly better with idiom annotations, while SICK++ without them.
Table III shows the results of the experiments performed using LLaMA, where we asked the model to summarize the chat

TABLE II
IDIOM EXTENSION

	SamSum				DialogSum			
	R-1	**R-2**	**R-L**	**BS**	**R-1**	**R-2**	**R-L**	**BS**
Experiments Configuration								
SICK (default configuration)	52,28	27,15	43,14	71,28	44,43	**19,73**	36,69	70,37
SICK & idiom preprocessing	**52,40**	**27,34**	**43,20**	**71,31**	**44,43**	19,67	**36,88**	**70,44**
SICK++ (default configuration)	**52,63**	**27,55**	**43,62**	**71,38**	**44,53**	19,79	**36,77**	**70,55**
SICK++ & idiom preprocessing	52,57	27,15	43,29	71,21	44,51	**19,81**	36,61	70,29

TABLE III
PROMPTING ON LLAMA

		SamSum				DialogSum			
		R-1	**R-2**	**R-L**	**BS**	**R-1**	**R-2**	**R-L**	**BS**
Input Examples	*Temperature*								
0	0	33,12	9,63	28,78	63,37	24,08	6,82	21,02	**60,65**
	0	**39,51**	**15,33**	**35,01**	**67,01**	29,17	8,60	26,07	57,48
2	0.4	38,89	14,45	34,40	66,69	**30,94**	**9,49**	**27,45**	59,57
	0.8	38,11	14,09	33,64	66,68	25,79	6,63	22,99	55,89
	0	38,52	14,60	34,55	65,77	27,77	8,05	24,51	54,62
4	0.4	35,99	13,06	31,98	63,98	23,88	6,14	21,47	50,17
	0.8	24,53	7,46	21,94	55,59	22,26	5,13	19,94	50,34

TABLE IV
EXPERIMENT ON FEW-SHOT PROMPTING

	SamSum				DialogSum			
	R-1	**R-2**	**R-L**	**BS**	**R-1**	**R-2**	**R-L**	**BS**
Few-Shot Experiments Configuration								
with commonsense	39,51	15,33	35,01	67,01	30,94	9,49	27,45	59,57
without commonsense	35,79	13,97	32,01	64,48	29,77	8,81	26,30	58,27
with commonsense and idioms annotations	**40,05**	**15,82**	**35,50**	**67,66**	**33,39**	**10,39**	**29,31**	**61,40**

dialog using a single sentence. In the zero-shot case, we used only a null value for the temperature parameter, as this type of prompting is the least robust. For the few-shot case, we tested different temperature values: 0, 0.4, and 0.8. The number of input examples used was either 2 or 4.

Analyzing the results obtained, we observed that the performance related to zero-shot prompting is generally the lowest, as expected. In the few-shot cases, we noticed that a higher temperature decreases performance due to the introduction of too much randomness, as evidenced by the significantly lower ROUGE metrics. Additionally, concerning the number of input examples, we observed that adding more examples to the model also increases the noise introduced, thereby decreasing the performance.

To conclude our investigation, we combined the previously described extensions. We performed the few-shot prompting using the augmented idiom dataset with the best configurations previously established: for Samsum dataset, 2 input examples and null temperature, while for DialogSum dataset, 2 input examples but a positive temperature (0.4).

Out of experimental curiosity, we also tried few-shot prompting using the original datasets without commonsense injection. Table IV presents these last results, where we can easily spot the importance of additional idiom information. Indeed, the best performances for few-shot prompting were obtained when the datasets were augmented with both idioms and commonsense annotations.

V. CONCLUSIONS

In this report, we introduced a summarizing framework for chat dialogues. We began with the baseline presented in [1], where commonsense is exploited to inject information into the dialogue, and subsequently enhanced it using different strategies.

Specifically, we enriched the datasets by incorporating additional information about the meaning of idioms. We observed that this preprocessing aids the prompting on the LLaMA model, but it has negligible impact within the context of the SICK++ architecture. This condition is likely attributable to the smaller number of training epochs used and the combined loss specific of SICK++. To support this assertion, consider that the extraction of commonsense was performed on dialogues where idioms and metaphors are present but have yet to be explained, and the introduction of a second decoder (SICK++ pipeline) could introduce some noise when the commonsense does not align with the idiom's meaning.

Regarding the utilization of a LLM via few-shot learning techniques, we found that the results obtained are competitive with the reference ones. This is noteworthy, as no training or fine-tuning was performed on the model used.

A. Future Work

We present a few aspects that could improve the overall performance in the future. When working with models like LLaMA, we should start by better tokenizing special characters such as emojis, as well as those used to indicate additional commonsense. Another improvement that we identified is the choice of examples used in the model's prompt. For example, given that different dialogues can vary substantially in length, we should provide a more consistent input where the dialogues are more similar to the one being summarized.

REFERENCES

[1] K. Seungone, J. Se June, C. Hyungjoo, K. Chaehyeong, H. Seung-won and Y. JinyoungG., “Mind the Gap! Injecting Commonsense Knowledge for Abstractive Dialogue Summarization”, International Committee on Computational Linguistics, 2022.

[2] Y. Chen, Y. Liu, L. Chen and Y. Zhang, “DialogSum: A Real-Life Scenario Dialogue Summarization Dataset”, Association for Computational Linguistics, 2021.

[3] B. Gliwa, I. Mochol, M. Biesek and A. Wawer. ”SamSum corpus: A human annotated dialogue dataset for abstractive summarization”, ACL Workshop, 2019.

[4] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave and G. Lample, “LLaMA: Open and Efficient Foundation Language Models”, February 2023.

[5] Prateek Saxena and Soma Paul, “EPIE Dataset: A Corpus For Possible Idiomatic Expressions”, 2020.

[6] H. Haagsma, J. Bos and M. Nissim, “MAGPIE: A Large Corpus of Potentially Idiomatic Expressions”, European Language Resources Association, May 2020.

[7] T. Zhang, V. Kishore, F. Wu, K.Q. Weinberger and Y. Artzi, “BERTScore: Evaluating Text Generation with BERT”, Feb 2020.

[8] Chin-Yew Lin, “ROUGE: A Package for Automatic Evaluation of Summaries”, Association for Computational Linguistics, 2004.

[9] A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Celikyilmaz and Y. Choi., “COMET: Commonsense Transformers for Automatic Knowledge Graph Construction”, Association for Computational Linguistics, 2019.

[10] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi and G. Neubig, ”Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing”, ACM Computing Surveys, 2023.

[11] Z. Chu, Z. Yang, Y. Cui, Z. Chen and M. Liu, ”HIT at SemEval-2022 Task 2: Pre-trained Language Model for Idioms Detection”, Association for Computational Linguistics, July 2022.

[12] Online Website, ”EF Education First: English Idioms”, ”https://www.ef.com/wwen/english-resources/english-idioms/”.

[13] Online Website, The Phrase Finder: Idioms and Phrases”, ”https://www.phrases.org.uk/idioms”.

[14] Mia Mohammad Imran, ”idiom-xlm-roberta (Revision aecca71)”, ”https://huggingface.co/imranraad/idiom-xlm-roberta”, 2022

[15] ”https://huggingface.co/meta-llama/Llama-2-7b-chat-hf”, 2023.

[16] S. Lee, A. Shakir, D. Koenig and J. Lipp, ”Open Source Strikes Bread - New Fluffy Embeddings Model”, ”https://huggingface.co/mixedbread-ai/mxbai-embed-large-v1”, 2024.

APPENDIX

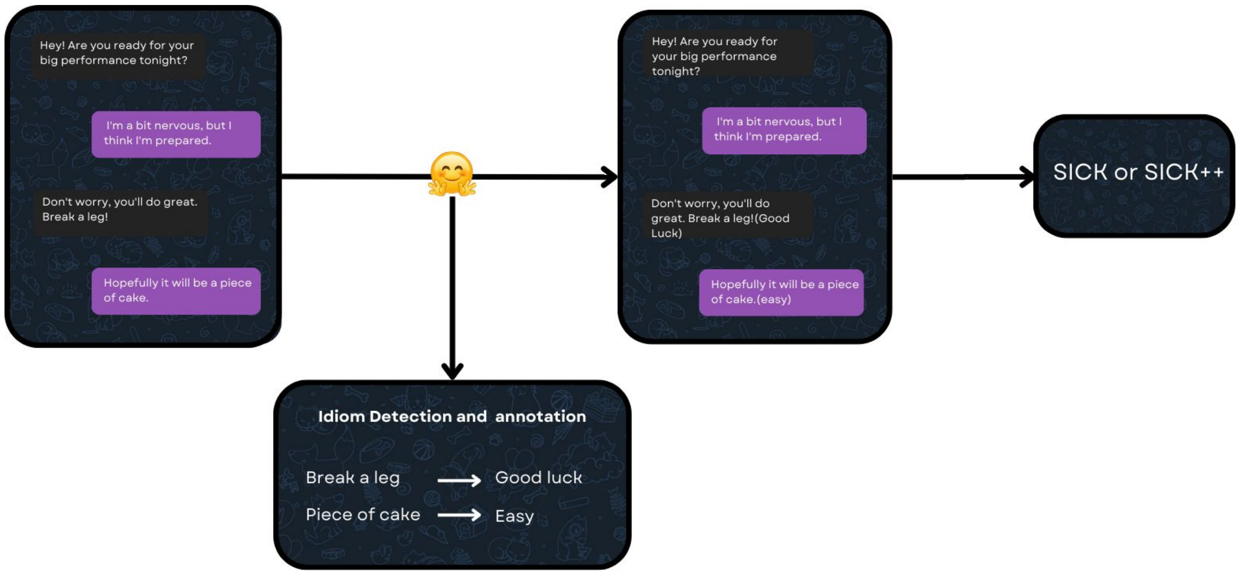
Hey! Are you ready for your big performance tonight?
I'm a bit nervous, but I think I'm prepared.
Don't worry, you'll do great. Break a leg!
Hopefully it will be a piece of cake.
Hey! Are you ready for your big performance tonight?
I'm a bit nervous, but I think I'm prepared.
Don't worry, you'll do great. Break a leg!(Good Luck)
Hopefully it will be a piece of cake.(easy)
SICK or SICK++
Idiom Detection and annotation
Break a leg → Good luck
Piece of cake → Easy

Fig. 3. Idiom Detection and Annotation