DSAI Project Seminar: Retrieval-Augmented-Generation of Discharge Documentation

Viet Dao Quoc vida00001@stud.uni-saarland.de
Xuanzhi Chen xuch00001@stud.uni-saarland.de
Universität des Saarlandes, Saarbrücken, Germany

Abstract

The challenge describes the task of ***Generating Discharge Summary Sections*** [1], aimed at reducing the time and effort clinicians spend on writing detailed notes in electronic health records (EHR). This task leverages the MIMIC-IV dataset, which includes 109,168 Emergency Department (ED) visits, and is organized into two sub-tasks: generating the "Brief Hospital Course" and "Discharge Instructions" sections of discharge summaries. The dataset is divided into training, validation, and two testing phases. The challenge is part of the 23rd Workshop on Biomedical Natural Language Processing (BioNLP) at ACL 2024. We follow the task instructions and build our system. The paper describes the data cleaning and text generation process, outlines the proposed system, reports the results achieved, and analyzes the main problems in the project and guides future research directions. The implications of our system are promising, as it has the potential to significantly improve clinical workflow efficiency in EHR documentation.

1 Introduction

In the modern healthcare environment, clinicians are required to document detailed notes in electronic health records (EHR) to track patient progress and ensure continuity of care. One critical aspect of this documentation process is the creation of discharge summaries, which encapsulate the patient's hospital course and discharge instructions. Despite their importance, preparing these summaries is time-consuming. This may contribute to clinician burnout and cause operational inefficiencies within hospital workflows.

The primary objective of the task described in this paper is to alleviate the burden on clinicians by automating the generation of discharge summary sections. Specifically, this task focuses on producing the "Brief Hospital Course" and "Discharge Instructions" sections, which are essential for summarizing patient care and providing follow-up instructions. Automating this process can significantly reduce the administrative workload on clinicians, allowing them to allocate more time to direct patient care.

This paper outlines the methodology and results of the task, providing insights into the potential of automated systems to support clinical documentation. The system is built based on the **Retrieval-Augmented-Generation** [8] method, which includes three parts: Retrieval part (patient's case comparison and search), Inference part

[1] https://stanford-aimi.github.io/discharge-me/

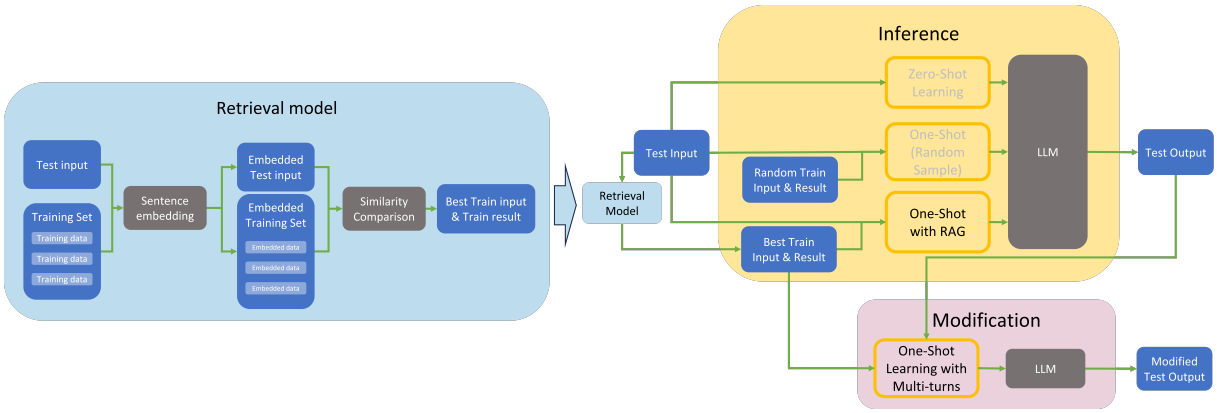

Figure 1: System Pipeline

(Few-shot-prompting and text generation) with Modification sub-part (text modification with multi-turn-prompting). By leveraging advanced natural language processing techniques and extensive healthcare datasets, we demonstrate how automation can play a vital role in modernizing healthcare workflows and improving outcomes for both clinicians and patients.

2 Dataset and System Pipeline

2.1 Dataset

The dataset has been split into a training (68,785 samples), a validation (14,719 samples), a phase I testing (14,702 samples), and a phase II testing (10,962 samples) dataset. Each dataset includes 5 input data files and one expected output file:

2.1.1 Input data

File Name	Content
Triage	Patient's basic physical data, including temperature, heart rate, respiratory rate, O2 saturation, systolic blood pressure, diastolic blood pressure, pain, acuity, and chief complaint.
Edstays	Patients' basic personal information.
Diagnosis	Doctor's initial diagnosis, including sequence number of ICD, ICD-code, ICD version, and ICD title.
Radiology	Radio-logical reports, including indication, examination technique, comparison, and findings.
Discharge	Complete medical records, including patients' medical history, physical exam results, brief hospital course, discharge medications, and discharge instructions.

Table 1: Description of input files and their contents

2.1.2 Expected Output

Discharge Instructions and **Brief Hospital Course**, which are the two components in dataset Discharge. Here's a structure example:

Discharge Instructions (DI)
"Dear Mr/Mrs.____, t It was a pleasure taking care of you during your recent hospital stay. We hope this information will help you take care of yourself at home. You were admitted to the hospital with ____. During your hospital stay, you ____. You have a history of ____. To care for yourself at home, please follow these instructions:____. Sincerely,____."
Brief Hospital Course (BHC)
"__(name)__ male/female with history of ____. In the examination, we found____. *Diagnosis:____ *History of Present Illness:____ *Past Medical History:____ *Physical Exam:____ *Discharge Diagnosis:____ *Medications:____ *Discharge Disposition:____"

Table 2: Description of Expected Outputs

2.2 Retrieval System

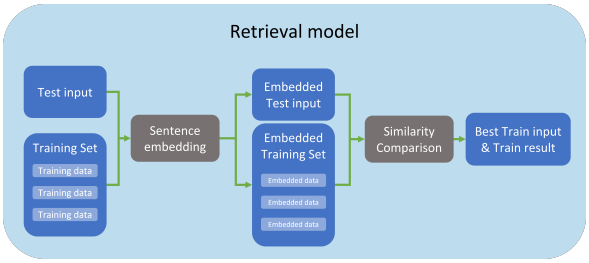

Figure 2: Retrieval System

Similar patient conditions may have similar Brief Hospital Courses and Discharge Instructions. Therefore, the goal of the retrieval model is to find the best data matching with the test data from the training set. This data is then passed to the inference system as a template helping the system generate better outputs. Among the large variety of data, it's impossible to read through all the medical records by human eyes, but the task of comparing text in bulk can be solved by using text quantization. The retrieval system is a vector matching system attach with a text-to-vector transformation model. We use Snowflake's Arctic-embed-m-long [12] sentence embedding model to convert target text into numerical vectors. These vectors capture the semantic meaning or relationships within the data. Then using cosine similarity comparisons, we can capture the similarity between two texts.

Since retrieval using all the input data was very time and space consuming, we decided to use one of the files as search reference. Because the Edstays dataset only contains patient basic information, we

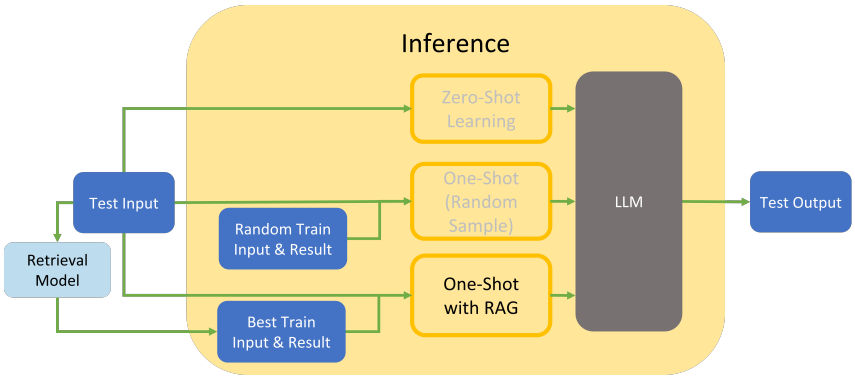

Figure 3: Inference System

select other four input data files as candidate retrieval input (triage, diagnostic, radiology, and discharge). In the triage dataset, we use simple list comparisons on chief complaint, because think it is more likely to show the cause of the patient's disease. In diagnosis, we use the same comparison method on ICD-code which is the code for preliminary diagnosis. Radiology and discharge reports are very long so we use the sentence embedding transformer, which is mentioned in the previous paragraph, transform the text to tensors and compute the cosine similarity between them. After retrieving the best matches for each sample, we look into the discharge instructions for each case to see how familiar the cases are. We finally found out that the retrieved patient cases using triage, diagnosis and radiology data, are not as similar as using Discharge data. Therefore, we select the Discharge dataset as the data for the retrieval model.

Before starting retrieval, all training data will be conversions to tensors using sentence embedding transformer. At the start of the retrieval, the discharge file of the given input data will be embedded in tensors and compared with each vector of discharge in the training set using cosine comparison. The training data with the smallest cosine value will be the most similar case to the input data. Among the sample data, the average cosine similarity is about 0.12.

2.3 Inference's Setting

We performed model assessments with **Few-shot-prompting** [11] techniques to test the knowledge of the current start-of-art transformer-based, instruction-tuned, pre-trained models for our task following some prompting- guidelines [4]. The prompt we used, has the format in Table 3:

Sections	Details
Instructions	*This section describes the instructions that we are going to give for the model to follow.
Examples	*This section is where we show the model an example with its corresponding label (in a Few-shot-prompting setting).
Content	*This section describes the details of how the generated output from the model should look like.

Table 3: Prompt's structure

Together with the following components:

- **query-input**: the target sample that we used the model to infer its expected output (query-output).
- **query-output**: the expected output of the corresponding sample (query-input).
- **sample-input**: the one example that we used as an example for the model to infer.
- **sample-output**: the corresponding label for the sample input, which is also passed into the model.

In this experiment, we focused on the assessment of Zero-shot and One-shot-prompting. Particularly, together with Retrieval Augmented Generation (RAG) introduced above, we inspected Zero-shot-prompting and 3 variations of One-shot-prompting, namely One-shot with/without RAG and One-shot with RAG plus multi-turn-prompting over a variety of different models: BioMistral-7B [7], Standford's BioMedLM [3], Qwen2-7B [1], *Meta-Llama3-70B* [13] and *Meta-Llama-3.1-70B* [5]. We are using *Instruct* version of all the models mentioned. Both *Meta-Llama3-70B* and *Meta-Llama-3.1-70B* have 70 billions parameters and are auto-regressive language model that uses an optimized transformer

architecture and trained on an offline dataset by Meta.

We chose these models from Meta because we believe that these models have seen lots of text data and have the ability to follow instructions and understand medical information better.

Zero-shot is the version in which we experiment on the model's ability to follow and understand instructions without exposing it to a single example.

One-shot with/without RAG are versions that we expose the model to one randomly sampled example. The version without RAG is where we used a fixed sample for the model, whereas the version with RAG implemented will be exposed to the most similar sample selected from the dataset using a similarity measure.

One-shot with RAG with multi-turn is the version in which we add an extra prompt to instruct the model to look at the generated output from the first inference phase to correct the false information or hallucinations produced.

As an example, the prompt for generating Discharge Instructions is as follows:

Prompt:

- This is a sample data: **sample-input**, generate a 'DISCHARGE INSTRUCTIONS' for the following data: **query-input**, which provides important medical information to help the patients manage their care when they leave the hospital.

The output of using this prompt was really poor, had a lot place-holders. We added more instructions to the prompts, to make the model understand more the task. Moreover, we did some analysis on a few samples of the *Discharge* dataset to find relevant information of the Discharge Summary that appears in its Discharge Instructions and Brief Hospital Course. Since this dataset was sampled with different writing style of the doctors, we found out that the parts that are consistent over all writing styles with most relevant information, are *'history-present-illness'*, *'past-medical-history'* and *'discharge-medication'* (only for Discharge Instructions). Therefore we added instructions in the prompts to ask the model to look explicitly into these specific part of Discharge Summary to extract the information for the task. We then arrived at this lengthy prompt, e.g for generating Discharge Instructions:

Prompt:

- *Instructions*:
  Generate 'DISCHARGE INSTRUCTION' for the patient based on the following data: **query-input**.
  \n The 'DISCHARGE INSTRUCTION' must provide actionable information to help them take care of themselves when they leave the hospital. \n

- *Examples*:
  The following is just a sample data, containing an 'EXAMPLE DISCHARGE INSTRUCTION' for the 'EXAMPLE INPUT'. \n
  'EXAMPLE INPUT': **sample-input** 'EXAMPLE DISCHARGE INSTRUCTION': **sample-output** \n

- *Content*:
  The 'DISCHARGE INSTRUCTION' of each patient of 'subject-id' and 'hadm-id' should be an overview of the conditions of the corresponding patient. \n
  The content of 'DISCHARGE INSTRUCTION' must contain information from 'discharge-summary'.\n
  Look into 'history-present-illness', 'past-medical-history' and 'discharge-medication' sentence by sentence to extract information on diagnosis and treatments administered. \n
  Avoid using any place holders. \n
  Avoid using '[insert diagnosis(s) e.g., pneumonia, heart failure]', instead provide specific diagnoses and treatments. \n
  The 'DISCHARGE INSTRUCTION' must be written in simple terms. \n Provide information about medical jargon \n
  Start your response with: 'Dear

Ms. /Mr‘ followed by ‘ ‘, which is based on the ‘gender’ from the ‘EXAMPLE INPUT‘. \n
Give an answer to a given question in a natural, human-like manner. \n
Avoid adding any personal information about the patient such as name and date of birth. \n”’

3 Evaluation metrics

The metrics for this task are based on a combination of textual similarity and factual correctness of the generated text. Specifically, we will consider the following metrics:

3.1 MEDCON

MEDCON [16] calculates the F1-score to determine the similarity between the Unified Medical Language System (UMLS) concept sets in both candidate and reference clinical notes. The extraction of UMLS concepts from clinical notes is performed using a string match algorithm, which is applied to the UMLS concept database through the QuickUMLS package [15], which is a tool for fast, unsupervised biomedical concept extraction from medical text.

3.2 BERTScore

BERTScore [18] use contextual embedding to represent the tokens, and compute matching using cosine similarity, optionally weighted with inverse document frequency scores. Here we use *distilbert-base-uncased* as base model to embed the texts for our results. It addresses two common pitfalls in n-gram-based metrics: fail to robustly match paraphrases and capture distant dependencies (e.g. BLEU and METEOR [2]).

3.3 ALIGNScore

ALIGNScore [17] is built by fine-tuning the lightweight RoBERTa [10] models for alignment. The alignment function is defined by unifying a large diversity of data sources, then combining the alignment function with a new context/claim splitting and aggregation strategy.

3.4 BLEU

BLEU [14] was originally used to compare the merits of translated texts. But it can also be used to show the similarity between texts. Counting every n-gram in candidate text shown in the reference text, then averaged over the whole corpus to reach an estimate of the translation's overall quality. Intelligibility or grammatical correctness are not taken into account.

3.5 ROUGE-L

ROUGE-L [9] uses Longest Common Subsequence (LCS) algorithm to measure the similarity between sequences by finding the maximum-length common sub-sequence. It has applications in translation lexicon construction and text similarity.

Among the five evaluation metrics, BLEU and ROUGE-L are traditional evaluation metrics focusing more rationally focusing on matching vocabulary and sentence structures. The other three evaluation methods have semantic comparison ability. Different from BERTScore and ALIGNScore, MEDCON is trained with medical data which has a better understanding of medical records.

4 Results and Discussion

As BioMistral-7B, BioMedLM, Qwen2-7B show that they have the poor ability to follow instructions, we mainly measure *Meta-Llama3-70B* and *Meta-Llama-3.1-70B* performance with the above-mentioned metrics.

First, using the previously mentioned version of the prompt, we assessed the *Meta-Llama3-70B* and *Meta-Llama3.1-70B* models. We added instructions to remove the Discharge Medications part for generating **Discharge Instruction** and **Brief Hospital Course** across 3,200 samples. We take the average of the scores from the evaluation for each task, together with the overall score being an average of all scores.

With *Meta-Llama3-70B*, we encountered some scenarios where the total tokenized input-tokens including the prompt exceeds max-input-tokens of 8192 tokens. We have tried to address the problem by reducing the total with the following approaches:

- **Splitting query-input or sample-input** into smaller chunks, constructed smaller prompts for those particular parts and then combine the smaller outputs into one output. Limitation: the run-time *doubles* for every prompt for the model (also with *Meta-Llama3.1-70B*). Therefore, we concluded this approach is not realistic.
- **Summarization of sample-input**, we constructed another prompt to ask the model to

capture important informations of the **sample-input** and then summarize it. The prompt for this approach is zero-shot based. Litmitation: therefore the output in this case contain lots of hallucinations (up to 70 percent).

The above mentioned approaches were somewhat promising but with our technical settings and capability of the models, we decided just to filter out instances that were causing the problem.
We further inspected that **sample-input** were not the cause for exceeding max-input-tokens but **query-input**. We decided using a filter that filters out instances of **query-input** that together with the prompt when tokenized has the length over 8192 tokens for version of **One-shot with/without RAG**. Meanwhile there is another filter for **One-shot with RAG and with 1-Turn**.
It is worth to notice that the run-time of the Inference Pipeline for each query-input from the test-phase-II dataset, using HuggingFace is approximately around 89 second for both model. The total run-time would be 734412 seconds which is roughly 8.5 days. To speed up the process, we reorganized our code and then used *vLLM* [6] a is a fast and easy-to-use library for LLM inference. It then reduced the run-time to roughly less than 5 second for each single prompt, equivalent to 13 hours for the entire test-phase-II and 3 hours for 3200 samples of test-phase-II.

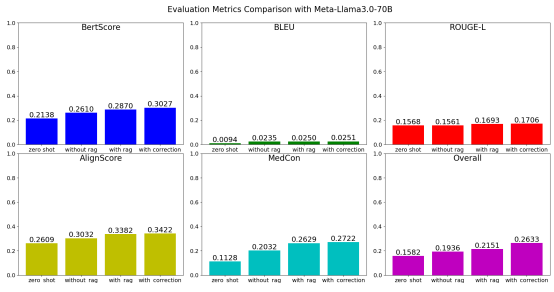

Figure 4: Overall scores with Meta-Llama3-70B

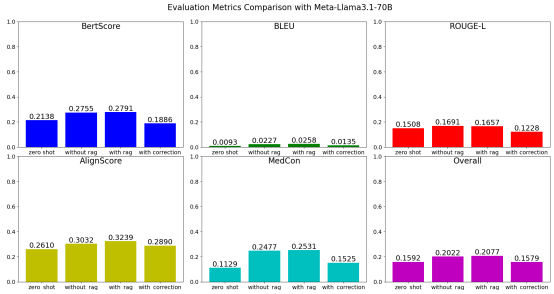

Figure 5: Overall scores with Meta-Llama3.1-70B

Based on the results above, we can see that the RAG model did not significantly improve the quality of the text. We have a few hypotheses regarding this:

- Firstly, the RAG system can introduce greater variability in the text, which may result in generated content that aligns less closely with the reference text. In many instances, the training examples and the expected test outputs originate from different physicians. Consequently, the generated response may deviate from the expected output. Additionally, we observed that the expected outputs sometimes omit certain information, such as medication details, in the discharge instructions. When our generated output includes these medications, the evaluation score tends to decrease.

- Secondly, the retrieval system identifies the best matches by comparing only superficial etiologies. For instance, in one case from our data, a man in the test set has a foot infection due to an injury, while another man has a foot infection caused by diabetes. Although both cases involve infections, the underlying causes differ, leading to distinct Discharge Instructions. Consequently, the retrieved example may inadvertently mislead the Large Language Models, resulting in less accurate outputs.

- Another reason why a RAG model might generate sub-optimal text is due to hallucinations in Large Language Models (LLMs). Hallucinations occur when the model generates content that is not based on the provided data or is factually incorrect. When the retrieval system provides examples that are incomplete or not perfectly aligned with the query, the LLM may fabricate details to fill in the gaps, leading to inaccurate or misleading output, which can compromise the quality and reliability of the generated text.

We saw that with **One-shot with RAG** and **One-shot without RAG** both model have approximately same values. But there is a significant difference with **One-shot with RAG with 1-Turn** displayed in the figures. We then inspected deeper

in the scores of *Meta-Llama-3.1-70B* of the prompt with extra instructions of removing Discharge Medication from Discharge Instructions when doing modification during 1-Turn Modification.

Figures 6, 7 and 8 indicate the scores of **Meta-Llama-3.1-70B**.[2]

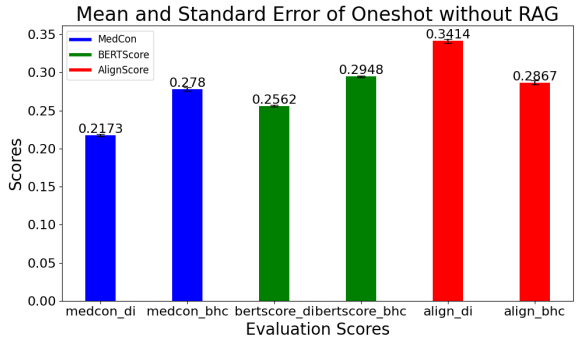

Figure 6: One-shot-prompting without RAG

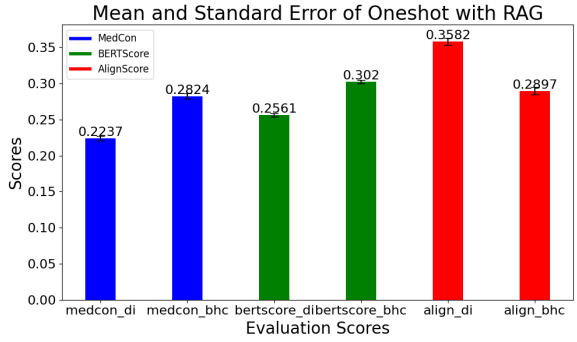

Figure 7: One-shot-prompting with RAG

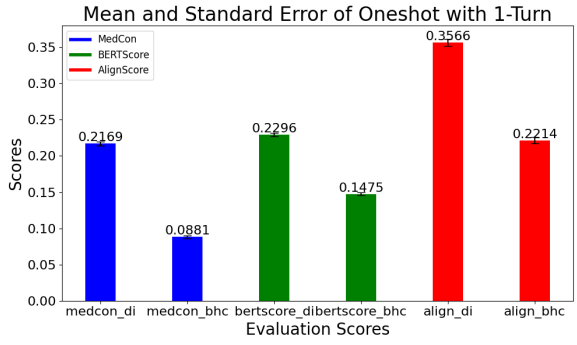

Figure 8: One-shot-prompting with RAG with 1-Turn

We noticed that the problem lies in **One-shot with RAG with 1-Turn**. In comparison to other versions, the scores experienced a decrease. This is because with the prompt for modification, the model introduced some hallucinations into the corrected/modified Brief Hospital Course, whereas removing the Discharge Medications from the Discharge Instructions may also result in a decrease in the scores.

This result gave us insight that there are things that effect the output of our system that we can not control. This can come from the choice of model for inference, for BERTScore, for ALIGNScore, and for Embedding model in the retrieval system, we do not know if the models have been trained on some medical dataset or not. It can also be about the choice of instructions that we used for our prompt, which is the choice of language, phrases and instructions that we used. But without fine-tuning the model we achieved an overall score of 0.2 with our system, which is 0.1 away from the leader-board from the shared task, and maybe gives enough insights for developments for new evaluation metrics specifically for this task.

5 Challenges

During the project, we encountered and overcame several challenges. In the retrieval model, we discovered that the data was disorganized: doctors' reports were in various formats, and some were even missing information. To address this, we developed a Python program using Pandas to clean and organize the data. When we scored the result, we found that there is currently no perfect text comparison system. BLEU and ROUGE-L cannot compare texture meaning. Although BERTScore and ALIGNScore provide meaningful text measurements, they fail to demonstrate a strong understanding of medical information due to their training on everyday language and lack of medical knowledge.

MEDCON can handle medical information, but it is difficult for non-medical staff to use because of the need to adjust medical parameters. Moreover, the optimal prompt for our language model remains elusive, and we are continuously refining it to enhance the model's comprehension of instructions. While operating the system, we realized that our computer configuration was insufficient, causing significant delays in processing the large volume of data. Fortunately, our tutor assisted us by running the code on a server. Furthermore, LLM hallucinations are still a big problem for us, and we are actively seeking solutions to this issue.

Throughout the project, we found that the system

[2]di: Discharge Instructions, bhc: Brief Hospital Course

demonstrates significant potential, but it is not yet suitable for deployment without expert review. The generated outputs may still somehow contain hallucinations that could lead to incorrect conclusions or actions if not carefully examined. Therefore, we would recommend that the system be used with expert validation. To enhance its reliability, further refinement and training are necessary, potentially including more advanced retrieval methods, improved handling of edge cases, and fine-tuning to reduce errors. Additionally, establishing a feedback mechanism, where experts can correct and refine the system's outputs, would contribute to its ongoing improvement.

6 Conclusion and Outlook

In this project, we demonstrated the effectiveness of our system in generating Discharge Summary Sections, significantly reducing the time and effort required by doctors. This highlights the potential of AI-driven solutions to transform healthcare documentation, alleviate clinician burnout, and improve overall hospital efficiency.

There is still considerable room for further exploration. Future development can be trying more accurate language models tailored to the medical field, such as BioMedLM, and developing the system for multi-shot-prompting, which could lead to even better output generation.

Throughout this project, we've glimpsed the promising future of AI in the medical field and feel confident about the industry's potential for continued innovation and impact.

References

[1] Qwen2 technical report. 2024.

[2] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

[3] Elliot Bolton, Abhinav Venigalla, Michihiro Yasunaga, David Hall, Betty Xiong, Tony Lee, Roxana Daneshjou, Jonathan Frankle, Percy Liang, Michael Carbin, et al. Biomedlm: A 2.7 b parameter language model trained on biomedical text. *arXiv preprint arXiv:2403.18421*, 2024.

[4] Sondos Mahmoud Bsharat, Aidar Myrzakhan, and Zhiqiang Shen. Principled instructions are all you need for questioning llama-1/2, gpt-3.5/4. *arXiv preprint arXiv:2312.16171*, 2023.

[5] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[6] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[7] Yanis Labrak, Adrien Bazoge, Emmanuel Morin, Pierre-Antoine Gourraud, Mickael Rouvier, and Richard Dufour. Biomistral: A collection of open-source pretrained large language models for medical domains, 2024.

[8] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[9] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[10] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[11] Ben Mann, N Ryder, M Subbiah, J Kaplan, P Dhariwal, A Neelakantan, P Shyam, G Sastry, A Askell, S Agarwal, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 1, 2020.

[12] Luke Merrick, Danmei Xu, Gaurav Nuti, and Daniel Campos. Arctic-embed: Scalable, efficient, and accurate text embedding models. *arXiv preprint arXiv:2405.05374*, 2024.

[13] AI Meta. Introducing meta llama 3: The most capable openly available llm to date, 2024. *URL https://ai. meta. com/blog/meta-llama-3/. Accessed on April*, 26, 2024.

[14] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[15] Luca Soldaini and Nazli Goharian. Quickumls: a fast, unsupervised approach for medical concept extraction. In *MedIR workshop, sigir*, pages 1–4, 2016.

[16] Wen-wai Yim, Yujuan Fu, Asma Ben Abacha, Neal Snider, Thomas Lin, and Meliha Yetisgen. Aci-bench: a novel ambient clinical intelligence dataset for benchmarking automatic visit note generation. *Scientific Data*, 10(1):586, 2023.

[17] Yuheng Zha, Yichi Yang, Ruichen Li, and Zhiting Hu. Alignscore: Evaluating factual consistency with a unified alignment function. *arXiv preprint arXiv:2305.16739*, 2023.

[18] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.