SUBMODULAR FUNCTIONS IN RETREIVAL AUGMENTED GENERATION

Prayas Agrawal
Department of Computer Science
IIT Bombay
{23m0757}@iitb.ac.in

ABSTRACT

The project aims to explore submodular optimisation in the LLM RAG setting, where a set of supporting evidence with respect to query is first obtained from a corpus, before being pass as context (usually) for downstream tasks. Most work employ MIPS (Maximum inner product search), or cosine distance as a similarity measure to retrieve top-k documents (sometimes with a fraction of negative documents for contrastive learning). I aim to explore the potential benefits submodular optimisation brings to the RAG literature, in the face of extremely large corpuses like wikipedia. To make the submodular optimisation tractable, a reranking approach is explored, where by a set of much smaller, potential candidate set is retrieved, over which final reranking is imposed via submod optimisation. In submodular optimisation, a relevant paper Lin & Bilmes explores submodularity in the document summarisation, which may be extended to RAG settings as well. However, there seems to be a lack of literature on applying submod in RAG settings. Additonaly, I propose, without results, potential formulation of submod techniques in retreival scheduling in the RAG setting.

1 INTRODUCTION

Owing to growing needs for LLMs to stay updated with the world knowledge, the ever increasing costs of training and updating its knowledge presents a significant challenge. To this end, a new approach has emerged to mitigate this which attempts to leverage retrieval. **Retrieval Augmented generation** (RAG), adds a retrieval component to the system to obtain relevant evidence before or during generation for answering an input query.

The metric for relevance score for a corpus document is usually the cosine similarity or bag of words similarity methods (TfIDF, BM25). In this report, I aim to explore the utility of submodular reranking techniques in the context of RAG. To my knowledge, no prior work has explored submodular techniques in the context of RAG. Since the retrieval corpus if usually is huge, I intend to adopt a reranking approach, wherein, first a larger set (100) of candidate documents are obtained, then a submodular routine optimises over these documents instead over the whole corpus. Additonaly, I propose, without results, potential formulation of submod techniques in retreival scheduling in the RAG setting.

2 PROBLEM AND DATASET

I evaluate the submodular functions on a natural language inference task (NLI). Given query q, the NLI language model LM_{NLI} must output one of entailment, neutral, contradiction. To make up for the lack of knowledge, as in commonly done in RAG settings, we use a seperate retrieval model LM_{RET} to retrieve documents $d_i \in D$ where D is a part of english wikipedia corpus, and append the documents as context to the NLI model. Its worth noting that providing a large context with a high noise to signal ratio is very likely to confuse the language model, thus its important that the retrieved documents accurately represent the given query.

Notice that there is no restriction on when the retrieval happens, and also how the documents are made to interact with the query. I work with the FEVER dataset with 200k train instances and 20K dev instances. The dataset attempts to label each instance with SUPPORTS, REFUTES, NOT ENOUGH INFO. For a given claim, we are required to retrieve evidence which justifies either of the above labels. I use 50K examples from the training set as validation.

3 Related Work

Realm Guu et al. (2020) is one of the first papers exploring the RAG setting for LM generation. RAG Lewis et al. (2021) extends Realm by optimising the retriever and finetuning on a larger set of tasks. Notice that the RAG problem can be formulated as long context problem where the entire corpus is sent as context to LM, and the LM is directly used to judge for irrelevant context, instead of first retrieving the relevant docs then providing as context. This has similarities to being a **primal-dual problem**. For this reason, research for long context optimisations is useful in the RAG setting. One such paper Unlimiformer Bertsch et al. (2023) retrieves documents per head of the transformer, instead of seperately training a retrieval model. In submodular setting, a relevant paper Lin & Bilmes applies submod optimisation for document summarisation. Surprisingly, there is a lack of literature on applying submod optimisation in RAG settings.

3.1 Dual Formulation

Disclaimer: This is just from my understanding, this not how the problem is referred to as in standard literature.
As described above, the primal problem for countering RAG is having a retrieval corpus, and filtering out relevant documents before being passed as context to the language model. Common approach for filtering is having a dedicated retriever model. The dual of this problem is having the entire corpus (or a fraction of it) provided to the LM (**possibly** as context) and its the job of the language model to determine relevant documents from the very large corpus. Bertsch et al. (2023) mimics such behaviour, although they dont term it as a dual formulation of RAG, such work often falls under **long context** understanding/approximations. A major limitation of unlimiformer is how it retrives per head per layer of the transformer, where clearly not every head necessitates retrieval. I propose a mathematical formulation, without results at the end which attempts to mitigate this possibly via a submodular formulation.

4 Methodology

4.1 Reranking and SubMod

Instead of submod optimising over the whole corpus, I resort to a reranking approach. The first set 50 candidates are acquired via desnse retrieval. The second set of 20 candidates are acquired via BM25 scoring over the previous 50 candidates. The final reranking to obtain 5 best documents via submod is what is finally input to the NLI model.

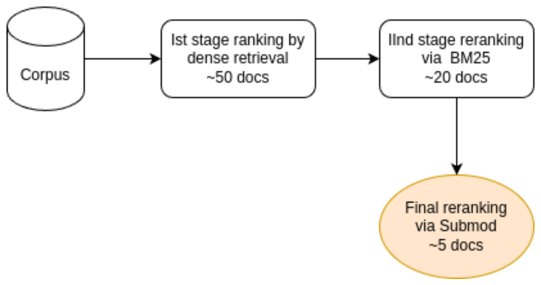
Corpus
Ist stage ranking by dense retrieval ~50 docs
IInd stage reranking via BM25 ~20 docs
Final reranking via Submod ~5 docs

Figure 1: Reranking pipeline

I test the final reranking with Facility Location MI, LogDet MI and GraphCut MI. I also define custom query relevant variant (described below) of similarity and test them over Facility Location and LogDet (**without MI**). Moreover sparse, dense, and a their convex combination is also tested (details below).

4.2 Query Relevant Similarity Measure

For the MI versions of submod objectives, its sufficient to provide seperate, interdocument and cross query-document similarity matrices. However, FacilityLocationMI objective only considers ground set - input set and ground set - query set interactions and completely ignores input set - query set interactions. LogDetMI can be tuned to favor query relevance over diversity. LogDetMI objective can be stated as:

$$logDetMI(A;Q) = \log\det(S_A) - \log\det\left(S_A - \eta^2 S_{A,Q} S_Q^{-1} S_{A,Q}^T\right) \tag{1}$$

The second term controls the query relevance. For a **single query** (thus S_Q^{-1}, is just a scalar), the relative importance of query similarity and document similarity is expressed as the difference of these similarities, as per the second term of the objective, which is roughly:

$$sim(d_i, d_j) - c * \eta^2 * sim(q, d_i) * sim(q, d_j) \tag{2}$$

For a single query, theres an alternative to instead express the relative importance as a product rather than difference.

Therefore, I experiment with directly modelling the document similarity matrix which takes into account the query and not just the retrieved documents, implying directly using the FacilityLocation and LogDet objectives **without MI**. For a **single** query q and a similarity matrix s

$$\begin{aligned} prob(b|a) &= softmax(sim(a,b);\ sim(a,v) \mid \forall v \in D) \\ s'_{ij}(d_i, d_j) &= prob(d_i|q) * prob(d_j|q) * sim(d_i, d_j) \\ s_i &= softmax(s'_i) \end{aligned} \tag{3}$$

where *sim(a,b)* may be cosine or BM25 depending on retrieval. Therefore, the similarity is weighted by query relevance with respect to a pair of documents. This metric reflects the low score of a pair of documents which are highly similar but of little relevance to query. Observe that this formulation neatly capture input set-query interactions as well, which was missing the MI variant of FacilityLocation

4.3 Sparse-Dense similarity Measure

Its commonly observed that bag of words similarity models often outperform dense retrievals in situations where entity overlap highly correlates with the queries of the downstream task. For this reason I experiment with a variant of dense and sparse retrieval which attempts to leverage the best of both worlds.

$$s_{ij} = \alpha * sparseProb(q, d_i, d_j) + (1 - \alpha) * denseProb(q, d_i, d_j) \tag{4}$$

where *denseProb, sparseProb* are probalities is dense or sparse retrievers settings respectively. The rationale behind such a combination is to benefit from both n-gram similarity and semantic similarity. The hyperparameter α is task dependent.

5 Implementation Details

To avoid the time involved with pretraining, I resort to a pretrained language model **DeBERTA-v3-base**. I finetune the model to the train set of the FEVER dataset of 200k instances. For the dense retriever model I use **multi-qa-MiniLM-L6-cos-v1**, which is a model tuned for semantic search: Given a query/question, it can find relevant passages. It was trained on a large and diverse set of (question, answer) pairs. The model is an encoder transformer, which independently takes the query and a set of documents, with query relevance being measured as dot product similarity of query with every other document. For sparse retriever BM25 is used.

Thus, for dense retrieval, I encode the whole wikipedia corpus with the retriever encoder, and at test/train time I use **FAISS** Johnson et al. (2019), for fast MIPS search of candidate documents with respect to an encoded query. For sparse retrieval, I choose BM25 score.

For a submodular optimizer, I chose **Decile's Submodlib**. As mentioned above, to make the submod optimisation over corpus, a reranking approach is undertaken.

For Sparse-dense similarity experiments, α is taken as 0.7 for all objectives.

6 Results

I experiment with Deberta-v3-base(86M parameters). A comparison with BART (from RAG,Lewis et al. (2021)) is included as well. The paper experiments with BART large(406M parameters, 5x larges that Deberta). As can be seen from the table below, sparse ranking performs significantly better than dense reranking. This can be attributed to the fact that our retriever model **multi-qa-MiniLM-L6-cos-v1** is only 15M parameters. However, a Sparse-Dense similarity has a positive effect over just sparse retriever and dense retriever. This confirms my hypothesis that both n-gram and semantic similarity are useful signals for the task. However to be noted is that α is 0.7 which implies we prefer sparse over dense by a large margin. Moreover, we observe there are improvements with using a Query relevant similarity measure which model cross interaction as multiplication rather than difference as in logDetMI. Also improvements over FacilityLocationMI which ingores inputSet-query interactions completely. The best performing model is RAG BART with 0.64 label accuracy. However to be noted is that our model is 5 times smaller than BART. Future work may explore performance on larger models

Method	**Accuracy**
No Context	0.335
Dense FacilityLocationMI	0.491
Dense LogDetMI	0.522
Dense GraphCutMI	0.546
QR FacilityLocation	0.503
QR LogDet	0.531
Sparse FacilityLocationMI	0.512
Sparse LogDetMI	0.534
Sparse GraphCutMI	0.551
S-D FacilityLocationMI	0.519
S-D LogDetMI	0.541
S-D GraphCutMI	0.567
BART Large	0.64

Table 1: Results

7 Hyperparameter tuning in sparse-dense similarity

I experiment for suitable values of α in the Sparse-Dense setting. Because of the time consuming experiment, the hyperparameter is determined over 2000 examples of the dev dataset using GraphCutMI. Its observed that a value of 0.7 performs best, implying that the downstream task finds n-gram overlap more advantageous than semantic signals, however pure n-gram signals are not better, implying downstream task still finds utility for semantic signals.

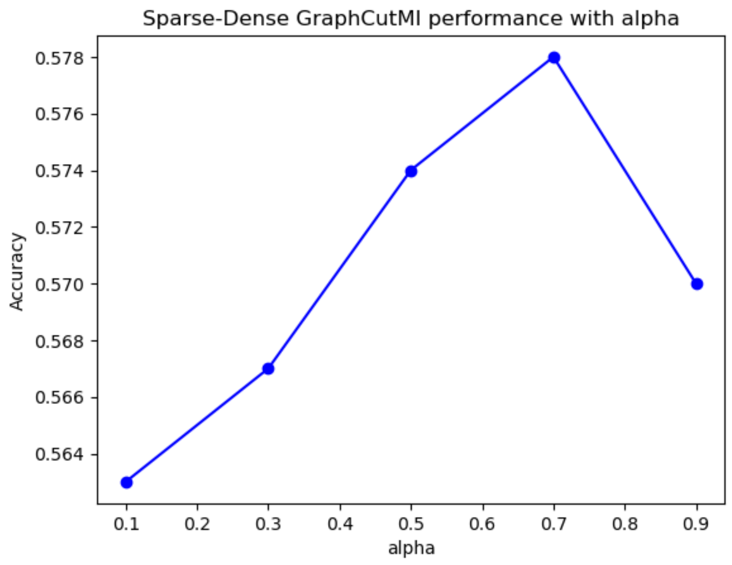

Figure 2: Sparse-dense parameter tuning

8 EXTRA, ROUGH PROPOSAL: DUAL FORMULATION BY RETRIEVAL SCHEDULING

Note: This is a rough mathematical formulation of the retrieval scheduled RAG problem using submod optimisation. I dont provide results for it. Moreover, such a formulation seems to be entirely new, atleast in RAG literature.

Motivation and background: Observe that the RAG problem poses no limitation on when to retrieve, and how to retrive. For example in my experiments above, I only retrive once and append it as context. However its entirely possible to retrive per token, and also even retrive amidst forward pass of the transformer. Here, I propose, without results a possible implementation of Submodular optimisation for Retrieval Scheduling in the RAG setting. As described above, Unlimiformer,Bertsch et al. (2023) experiments with the dual variant of the RAG problem, commonly called as long context learning. The paper retrieves top-k encoder embeddings per head, per layer from the decoder of an encoder-decoder model. **No citation of the paper** so far has yet considered retrieval scheduling, ie selectively choosing which heads to enforce retrieval upon. Moreover, one of the popular papers in retrieval scheduling in the RAG setting, SelfRAG Asai et al. (2023), uses a finetuning dataset to ellicit the LM to output special tokens ([RET]) during generation, which fire retrievals for the subsequent generation. It doesnt seem like there exists derivative work which directly attempts to induce retrieval scheduling in a more structured way, rather than just finetuning over relevant datasets.

Method: For a N layer decoder with h heads per layer, consider a **trainable** model $J \in R^{N*h}$, where each element denotes the probability of a being active for retrieval. It seems reasonable to model it is a MLP with inputs the input-query embeddings, layer, and head indices, and output the probability of being active for retrieval
For a head h_{ij} where i is the layer index, and j is the head index at the ith layer. Consider the set $\{h_{ij}\}$ of all the heads in a layer. Define the similarity between two heads (within same layer) as a measure of increase in model accuracy if the heads were active. $Acc_{active}(i, \{j, k\})$ denote the accuracy measure if heads j,k at ith layer were active for retrieval. Then, one such similarity measure could be:

$$s_i(j, k) = Acc_{active}(i, \{j, k\}) - Acc_{no\_retrieval} \tag{5}$$

Thus, our first step is to create a training data. The training data is simply $(q_i, \{r_{jk}\})$, where for the ith query, we determine per layer j the ideal retriever heads k. These heads are obtained via submod optimisng the above formulation.

Now to train the J network, we simply minimise the cross entroy loss over the ideal heads and heads what J thinks are appropriate.

Note that the similarity matrix is constructed per layer. Thus we have N similarity matrices, one per layer.
Note however that calculating Acc_{active} can be expensive for the whole dataset, and for every pair of head, thus a reasonable option is to approximate it via markov samples over train data or even a subset selection problem itself. Also observe that the number of attention heads is usually small. Example: 32 for LLama70b, thus making the problem somewhat tractable.
Notice that J is the only trainable component, we dont update the underlying LM. At inference, we sample a multinomial from the J and threshold appropriately to reduce the number of retrieval steps, thus in effect pursuing retrieval scheduling. Retrieval scheduling in RAG settings (using a mathematical formulataion) and modelling head indices directly as part of scheduling submodular objectives seems to be new.

9 Conclusion

In this report, I detail the utility and results of applying submodular optimisation in the RAG setting. As noted before, there seems to be a lack of work in the area which attempts to appyling submod optimisation in RAG setting. To this end I test FacilityLocationMI, LogDetMI and GraphCutMI objectives. Additionaly, I propose variants to the similarity matrices which further improve the accuracies. To be noted is that our Deberta system is competitive with the 5x larger BART system, which further motivates me to evaluate the technniques further for other tasks and dig deeper on the submod objectives used. In particular, one may evaluate the task on SNLI, which is a a large dataset of 570K instances. Moreover, as noted before, since there is no restriction on when the retrieval happens, exploring submod techniques for retrieval scheduling is another exciting direction to explore

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection, 2023.

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R. Gormley. Unlimiformer: Long-range transformers with unlimited length input, 2023.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training, 2020.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization.