Submodular Context Selection for Long-Document Question-Answering

Adithya Bhaskar (190050005)
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
`adithyabhaskar@cse.iitb.ac.in`

Harshit Varma (190100055)
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
`harshitvarma@cse.iitb.ac.in`

1 Introduction

Question-Answering is a task of central importance in Natural Language Processing, as it finds widespread use in eclectic domains. Typically, however, real-life documents on which questions must be answered are very long, whereas most of the attention of researchers has traditionally been concentrated on the setting where the context used to answer questions fits into the context window of Transformer-based Language Models (LMs). In this document we focus on the task of Long-Document Question Answering (LDQA), wherein the contexts run into thousands or even tens of thousands of characters, well beyond the reach of commonplace language models, save for those specifically designed to tackle long inputs. We are specifically interested in the case where individual statements are inadequate to supply an answer, but rather one needs to combine and reason over *multiple* (often far apart) sentences in order to answer a question. Given this, it will be inefficient (due to quadratic complexity of vanilla attention) and ineffective for standard LLM-based QA models to encode and work with such context directly. We design a conditional context selection module that selects the relevant sentences, conditioned on the given question. One natural way to do this is via subset selection, using submodular functions. These functions offer remarkable flexibility in choosing an objective that closely models our intentions, as these objectives are often possible to design so as to be submodular irrespective of the similarity metric used. We then pass only the relevant sentences chosen by the conditional context selection module standard QA model that then outputs the answer. Conditionally selecting the context can enable any standard QA model to: (i) focus only on the relevant parts and to not get "distracted" by the remaining large number of sentences in the document and (ii) efficiently work with long documents as context without using heuristics to decrease the complexity of the attention mechanism. An additional benefit of using submodular objectives is that we can aim to strike a balance between query relevance (so that we answer the question correctly) and representation (so that we have a decent shot even if we misclassified the important spans of text). We evaluate our methods on the challenging QuALITY dataset [14], where there still exists a large gulf between the State-of-The-Art method and human performance. To the best of our knowledge, such a conditional context selection module based on subset selection in the context of QA has not been studied in prior work, and is novel.

Preprint. Under review.

Split	No. of Articles	Avg. Article Length (Characters)	No. of Questions
`train`	300	24348.86	2523
`dev`	230	24306.59	2086

Table 1: Characterisitcs of the QuALITY dataset. Contexts often exceed 20,000 characters in length.

2 Related Work

2.1 Efficient Attention in Transformers and Efficient Non-transformer Architectures

Several methods [1, 3, 12] have attempted to reduce the complexity of the quadratic attention mechanism used in transformers via changing the computation mechanism or hardware-related optimizations. Methods that modify the attention mechanism usually suffer from an efficiency-accuracy trade-off. In contrast to these, our approach attempts to tackle the problem at the source: by reducing the number of input tokens depending on the input query. Note that both such approaches can easily be integrated with our method to provide further speedups. Since our approach simply reduces the input size, it can also be used with recently proposed non-transformer models like S4 [4].

2.2 Selector-Predictor Models

In the context of event relation extraction, Man et al. use the REINFORCE algorithm to select the sentences relevant for predicting the relation between two given input events. These sentences are then passed to a predictor network (e.g., any standard transformer-based model) that then predicts the relation. This is similar to retrieval-augmented language models like REALM [5].

2.3 Submodular Subset Selection in NLP

Submodular subset selection has been previously shown to be effective for document summarization and conditional data summarization [11, 9]. Kumar et al. also showed the effectiveness of submodular functions for diverse paraphrasing and data augmentation. However, we are not aware of any work that applied Submodular Optimization to the domain of Question Answering.

3 Problem Formulation

A question q to be answered using a document $d = \{s_i\}_{i=1}^{l}$ of sentences, given options A. Let the correct answer be $a^* \in A$, such that $a^* = \mathrm{argmax}_{a \in A} \mathrm{P}(a|q, d)$. We assume that not all sentences present in d are required or relevant for answering the given question q. We call the *smallest* subset of the document d that is relevant for answering the given question $q \in \mathcal{Q}$ as the context $c^* \subset d$, such that $\mathrm{argmax}_{a \in A} \mathrm{P}(a|q, d) \approx \mathrm{argmax}_{a \in A} \mathrm{P}(a|q, c^*)$. $c^* = \mathrm{argmax}_{c \subset d} \mathrm{P}(a^*|q, c)$. Note that the sentences s_j^* in c need not necessarily be adjacent in d. We further define a *selector* to be a function M_S that maps, given parameters θ_S and a budget b on the number of sentences, the input document-query pair to a set of size at most b:

$$M_S(d, q|\theta_S; b) := d_q \subseteq d, \quad |d_q| \leq b$$

The selections d_q are then passed to a *predictor* M_P that defines a probability distribution $M_P(a|\theta_P; q; d_q)$ over the space of all possible answers, conditioned on the parameters θ_P, selection d_q and query q. Each model M in this setting may be considered as a combination (M_S, M_P) of a selector and a predictor (with the selector possibly producing the entire context with a large enough budget, in some cases).

4 Dataset

We evaluate our methods on the recently proposed QuALITY dataset [14] that focuses on the task of single-correct multiple-choice (4 choices) question answering over long documents. The `train` and `dev` splits of QuALITY are publicly available, and have the characteristics shown in Table 1 From the `dev` split, we select 625 questions randomly to form the validation split, and 1461 to form the test split. Henceforth, we shall refer to the former as the `dev` split and the latter as the `test` split, at the risk of a mild abuse of definition.

As QuALITY consists largely of text from novels, most questions require knowledge that can only be pieced together from numerous spans, and are rather hard even for diligent, but time-pressed human annotators.

5 Methodology

5.1 Submodular Functions

Given a ground set V, a submodular function f maps each set $A \subseteq 2^{|V|}$ of V to a real number x, such that for any two sets $A \subseteq B \subseteq V$ and element $x \in V$, we have

$$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$$

It turns out that submodular functions possess helpful properties that allow efficient algorithms for their optimization in the face of budgetary constraints. Readers are referred to sources such as the CS 769 course notes of IIT Bombay or this tutorial[1] for more information. It will suffice here to say that Greedy methods are provably near-optimal for maximizing monotone submodular functions.

Of special interest to us is that a number of submodular (and often, monotone) objectives may be defined on any arbitrary similarity function that capture either the faithfulness of a selection to a query or the extent to which it represents the ground set of spans. We are, in particular, interested here in the objectives that are based on Submodular Mutual Information $I_f(A; B) = f(A) + f(B) - f(A \cup B)$.

5.2 Approach

We use the `submodlib` [7] library for the implementations of the submodular objectives. More specifically, we evaluate the Mutual Information based versions of the Facility Location, Graph Cut and Log Determinant objectives, along with the Concave-over-Modular objective. Note that for the case of a single query (which holds always for us), Graph Cut is equivalent to choosing the greedy selection of the most similar spans. For the latter two, we also experiment with an initial filtration step. This step first filters out $2.5\times$ the number of spans as specified by the budget greedily based on similarities to the query. The filtered spans are then passed to the usual submodular optimization step.

We use Facebook DPR [6] to generate embeddings for spans prior to optimization, although we also support Sentence Transformers [16]. Our spans are contiguous sets of up to five sentences. In addition, we include TF-IDF and BM25 based greedy baselines for comparison. We also support the use of other objectives with TF-IDF and BM25 based embeddings, but we do not compare to them, as we found them to anecdotally perform similar to the greedy objective. We use sklearn[2] for the former, and the rank-bm25 [2] Python library for the latter. Our budget is spans in all cases. We also compare to empty and randomly chosen contexts with the same budget.

We noticed that the best-performing methods at the QuALITY leaderboard[3] included a pre-training step on the RACE [10] dataset, so we also include this step for all of our approaches. We also include a fine-tuning step on the train split of QuALITY for each approach on selections, based on the same approach. Our training code is a modified version of the LRQA [15] code. We use a maximum input length of 512 tokens. A learning rate of $2 \cdot 10^{-5}$ was used for the RACE pretraining, with an effective batch size of 128 via gradient accumulation. Evaluation was done every 64 updates, with training proceeding until the accuracy tapered off. We observed that this typically took around 1000 gradient updates. The hyperparameters used were the same for fine-tuning on QuALITY, with the only change being a learning rate of $1 \cdot 10^{-5}$. The fine-tuning typically converged in 300-500 updates.

[1] https://theory.stanford.edu/ jvondrak/data/submod-tutorial-1.pdf
[2] https://scikit-learn.org/stable/
[3] https://nyu-mll.github.io/quality/

6 Results

Table 2 shows the quantitative performance of various methods on the QuALITY dataset. We plan to incorporate more methods and baselines in the future.

We would like to make a few remarks on the above observations. First, we notice that even without any context, DeBERTaV3 achieves pretty strong performance. Second, adding a randomly chosen context of 10 spans (around $40-50$ sentences) helps as much as going from a random baseline to the best performing system. However, our numbers for the Graph Cut (greedy most-similar) objective roughly match those reported on the leaderboard under the "DeBERTaV3 + DPR" heading. We observe that Concave-Over-Modular, especially with filtration, outperforms Graph Cut and gets the best results, albeit by a small margin. Most submodular approaches outperform BM25, but TF-IDF surprisingly achieves similar scores. We also observe that Facility Location (FL-MI) performs particularly poorly, as it highly prefers diversity over query-relevance.

Method	Description	Accuracy
No Context	Only the question is provided as input	44.55
Random	Context is selected randomly	50.38
TF-IDF	TF-IDF similarity is used to select sentences	55.44
BM25	The Okapi BM25[4] [17] is used to retrieve the sentences	52.91
GraphCut-MI	$I_f(A;Q) = 2\lambda \sum\limits_{i\in A} \sum\limits_{j\in Q} s_{ij}$	55.10
FL-MI	$I_f(A;Q) = \sum\limits_{i\in V} \min(\max\limits_{j\in A} s_{ij}, \eta \max\limits_{j\in Q} s_{ij})$	50.24
LogDet-MI	$I_f(A;Q) = \log\det(S_A) - \log\det(S_A - \eta^2 S_{A,Q} S_Q^{-1} S_{A,Q}^T)$	54.55
F+LogDet-MI	Filtration followed by LogDet-MI	52.29
CoM-MI	$I_{f_\eta}(\mathcal{A};\mathcal{Q}) = \eta \sum_{i\in\mathcal{A}} \psi(\sum_{j\in\mathcal{Q}} s_{ij}) + \sum_{j\in\mathcal{Q}} \psi(\sum_{i\in\mathcal{A}} s_{ij})$	55.99
F+CoM-MI	Filtration followed by CoM-MI	56.14

Table 2: Results on the QuALITY dataset

7 Conclusion

In this work, we propose a conditional context selection module to select the sentences relevant for answering a given question. We do this is via subset selection using submodular functions. The selected sentences are then passed as context to a standard QA model that then outputs the answer. We evaluate our methods on the challenging QuALITY dataset [14], demonstrating that the subset selection module outperforms multiple baselines. In the future, we also plan to evaluate our methods on more datasets from the SCROLLS benchmark [18].

Submission structure

Configuration parameters for subset selection can be modified in `config.py`, while global variables are in `utils/globals.py`. Submodular optimization functions defined in `utils/submodular.py` call upon embedders in `models/embedder.py` to embed sentences. In turn, helper functions in `models/subset_selection.py` use the assistance of `utils/data.py` to load the data, then call the appropriate functions in `utils/submodular.py` to select and save spans of sentences. The file `main.py` brings together all the pieces of subset selection and allows one to change the objective used for the same. Once the subsets have been selected for each data instance, scripts in `training/scripts/` may be used to prepare the same for model training or inference. Both proceed by calling `training/lrqa/run_lrqa.py` with a configuration file inside `training/configs/`. Prior to calling `run_lrqa.py`, add `training/` to your `PYTHONPATH`, if not calling from the same directory.

Note: As GitHub as well as Moodle have restrictions on size, we are unable to upload the datasets or the subsets we obtain from our methods. Running `main.py` with the appropriate parameters generates the subsets under `data/`, and the scripts in `training/scripts/` (once modified to use the above paths) prepare the training data under `training/data/`.

References

[1] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer. *ArXiv*, abs/2004.05150, 2020.

[2] D. Brown. The `rank-bm25` Python package. `https://pypi.org/project/rank-bm25/`, 2023. [Online; accessed April 26, 2023].

[3] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. R'e. Flashattention: Fast and memory-efficient exact attention with io-awareness. *ArXiv*, abs/2205.14135, 2022.

[4] A. Gu, K. Goel, and C. R'e. Efficiently modeling long sequences with structured state spaces. *ArXiv*, abs/2111.00396, 2021.

[5] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang. Realm: Retrieval-augmented language model pre-training. *ArXiv*, abs/2002.08909, 2020.

[6] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. tau Yih. Dense passage retrieval for open-domain question answering, 2020.

[7] V. Kaushal, G. Ramakrishnan, and R. Iyer. Submodlib: A submodular optimization library, 2022.

[8] A. Kumar, S. Bhattamishra, M. Bhandari, and P. P. Talukdar. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. In *North American Chapter of the Association for Computational Linguistics*, 2019.

[9] L. Kumari and J. A. Bilmes. Submodular span, with applications to conditional data summarization. In *AAAI Conference on Artificial Intelligence*, 2021.

[10] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy. Race: Large-scale reading comprehension dataset from examinations, 2017.

[11] H.-C. Lin and J. A. Bilmes. A class of submodular functions for document summarization. In *Annual Meeting of the Association for Computational Linguistics*, 2011.

[12] L. Madaan, S. Bhojanapalli, H. Jain, and P. Jain. Treeformer: Dense gradient trees for efficient attention computation. *ArXiv*, abs/2208.09015, 2022.

[13] H. Man, N. T. Ngo, L. N. Van, and T. H. Nguyen. Selecting optimal context sentences for event-event relation extraction. In *AAAI Conference on Artificial Intelligence*, 2022.

[14] R. Y. Pang, A. Parrish, N. Joshi, N. Nangia, J. Phang, A. Chen, V. Padmakumar, J. Ma, J. Thompson, H. He, and S. Bowman. Quality: Question answering with long input texts, yes! In *North American Chapter of the Association for Computational Linguistics*, 2021.

[15] J. Phang. The `lrqa` GitHub repository. `https://github.com/zphang/lrqa`, 2023. [Online; accessed April 26, 2023].

[16] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[17] S. E. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3:333–389, 2009.

[18] U. Shaham, E. Segal, M. Ivgi, A. Efrat, O. Yoran, A. Haviv, A. Gupta, W. Xiong, M. Geva, J. Berant, and O. Levy. SCROLLS: Standardized CompaRison over long language sequences. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 12007–12021, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. URL `https://aclanthology.org/2022.emnlp-main.823`.