Interactive Query-Assisted Summarization via Deep Reinforcement Learning

Ori Shapira[1,3*], **Ramakanth Pasunuru**[2], **Mohit Bansal**[2], **Ido Dagan**[1], **and Yael Amsterdamer**[1]

[1]Bar-Ilan University [2]UNC Chapel Hill [3]Amazon

Abstract

Interactive summarization is a task that facilitates user-guided exploration of information within a document set. While one would like to employ state of the art neural models to improve the quality of interactive summarization, many such technologies cannot ingest the full document set or cannot operate at sufficient speed for interactivity. To that end, we propose two novel deep reinforcement learning models for the task that address, respectively, the subtask of summarizing salient information that adheres to user queries, and the subtask of listing suggested queries to assist users throughout their exploration.[1] In particular, our models allow encoding the interactive session state and history to refrain from redundancy. Together, these models compose a state of the art solution that addresses all of the task requirements. We compare our solution to a recent interactive summarization system, and show through an experimental study involving real users that our models are able to improve informativeness while preserving positive user experience.

1 Introduction

Integrating human interaction into NLP tasks has been gaining the interest of the NLP community. Human-machine cooperation can improve the general quality of results, as well as provide a higher sense of control for the targeted consumer. We focus on the task of *interactive summarization* (INTSUMM: Shapira et al., 2021b) which enables information exploration within a document set on a topic, by means of user-guided summarization. As illustrated in Figure 1, a user can incrementally expand on a summary by submitting requests to the system, in order to expose the information of interest within the topic. A proper exploration session demands access to *all* information within the document set, and fast reaction time for smooth human engagement (Anderson, 2020; Attig et al., 2017). In addition, presented information must consider the session history to refrain from repetitiveness.

While it is worthwhile to apply recent NLP advances that excel at extracting salient and query-biased information, those advances usually come at a cost of rather small input size limits or heavy computation time. Indeed, all previous interactive summarization systems we know of either apply traditional methods or are inadequate for real-time

[*] This work was conducted prior to joining Amazon.

[1]Code and trained models at: `https://github.com/OriShapira/InterExp_DeepRL`

PREPRINT

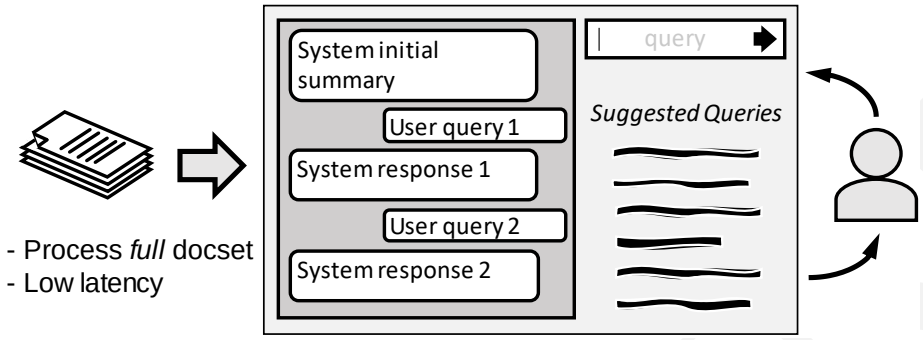

Figure 1: An INTSUMM system, ingesting a large document set. A user interactively submits queries in order to expand on the information. The system is required to process the full document set for comprehensive exploration, respond quickly, and expose non-redundant salient information that also complies to the input queries. See real example in Figure 5.

processing due to high latency (§2). Our goal is to overcome these obstacles, and leverage advanced methods to improve information exposure while keeping latency acceptable for interaction.

As depicted in Figure 1, an INTSUMM system provides an *initial generic summary* as an overview of the topic, after which a user can iteratively issue queries to the system for *summary expansions* on subtopics of interest. To support querying, the system offers a list of *suggested queries*, hinting at information concealed within the document set.

We address the INTSUMM task components through two subtasks: (1) generating the initial summary and query responses, and (2) generating lists of suggested queries. For each of the subtasks we propose a deep reinforcement learning (RL) algorithm that addresses the respective subtask requirements. To enable *comprehensive* topic exploration, our models speedily process the *full* document set, as inspired by Mao et al. (2020). Additionally, they are able to peek at session history to comply to the current state of the interaction. The model for the query-assisted summarization subtask, M_{Summ}, incorporates the query sequence by (1) encoding a query into the contextual sentence representations, (2) attending the representations using a new query-biased variant of the maximal marginal relevance (MMR: Carbonell and Goldstein, 1998) function, and (3) a dual reward mechanism for policy optimization (Pasunuru and Bansal, 2018) which we adapt to consider both reference summaries and the query (§3). The model for the suggested queries list generation subtask, M_{Sugg}, works at the phrase level, as opposed to the sentence level, to enable extraction of important phrases that serve as suggested queries. Similarly to M_{Summ}, the model learns importance with consideration to session history, but without an input query – as its role is to *suggest* such a query (§4).

The models are trained on the DUC[2] 2007 multi-document summarization (MDS) news-domain dataset, with adaptions for our task setting. For testing, we follow the INTSUMM evaluation framework of Shapira et al. (2021b) to run simulations, collect real user sessions, and assess the results, using DUC 2006. In principle, summary informativeness, i.e. general salience, could potentially come at the expense of query responsiveness, but importantly, our results show that our RL-based solution is able to

[2] https://duc.nist.gov/

significantly improve information exposure over the baseline of Shapira et al. (2021b), without compromising user experience (§5).

2 Background and Related Work

Interactive summarization facilitates user-guided information navigation within document sets. The task suffered from a lack of a methodological evaluation, until Shapira et al. (2021b) formalized the INTSUMM task with a framework consisting of a benchmark, evaluation metrics, a session collection process and baseline systems. This framework, that we leverage, enables comparison and analysis of systems, allowing principled research on the task and accelerated development of algorithms.

To the best of our knowledge, all previous works on INTSUMM have either applied more traditional text-processing methods or require costly preprocessing of inputs to facilitate seamless interaction. Leuski et al. (2003) used surface-form features for processing content, and Baumel et al. (2014) adapted classic MDS algorithms like LexRank (Erkan and Radev, 2004) and KLSum (Haghighi and Vanderwende, 2009). Christensen et al. (2014) optimized discourse graphs and Shapira et al. (2017) relied on a knowledge representation, both expensively pre-generating hierarchical summaries that limit expansions to pre-prepared information selections. Hirsch et al. (2021) applied advanced coreference resolution algorithms that take several hours for preprocessing a document set.

The two INTSUMM baseline systems of Shapira et al. (2021b) use sentence clustering or TextRank (Mihalcea and Tarau, 2004) for summarization, sentence similarity heuristics for query-responses, and n-gram frequency or TextRank for suggested query extraction. Moreover, their query-response generators strictly consider a given query, ignoring history or global informativeness. Our proposed algorithms significantly improve information exposure over the latter baselines, using advanced deep RL methods, working in real time. We next review some recent techniques in MDS, query-focused summarization and multi-document keyphrase extraction, all of which relate to the INTSUMM task and our choice of algorithms.

The subtask of query-assisted summarization. Non-interactive MDS has been researched extensively, with few recent neural-based methods that can handle relatively large inputs. For example, Wang et al. (2020) use graph neural networks to globally score sentence salience, Xiao et al. (2021) summarize using Longformers (Beltagy et al., 2020), and Pasunuru et al. (2021b) combine a Longformer with BART (Lewis et al., 2020) and incorporate graphical representation of information. Mao et al. (2020) apply deep RL for autoregressive sentence selection, and, in contrast to most other neural methods, can ingest the *full* document set.

In the query-focused summarization (QFS) task summaries are biased on a query. To accommodate a query, Xie et al. (2020) use conditional self-attention to enforce dependency of the query on source words. Pasunuru et al. (2021a) and Kulkarni et al. (2021) hierarchically encode a query with the documents. These and other QFS methods require large training sets, and limit the allowed input size (Baumel et al., 2018; Laskar et al., 2020). Relatedly, incremental update summarization (McCreadie et al., 2014; Lin et al., 2017) marks query-relevant information as reported texts stream in, avoiding repeating information marked earlier. Interactivity is not a constraining factor here, yielding solutions with relatively high computation time.

With respect to the above related work, we develop a model inspired by Mao et al. (2020), which is closest to our requirements. To facilitate an interactive setting, our model (1) enables query+history injection, (2) supports full input processing, necessary for complete information availability during exploration, (3) has low latency at inference time, and (4) requires a relatively small training set.

The subtask of suggested-queries list generation. Extracting suggested queries on a document set most resembles the multi-document keyphrase extraction (MDKE) task since it aims to identify salient keyphrases (Shapira et al., 2021a). MDKE was mostly addressed using traditional heuristics or graph-centrality algorithms applied over the documents (e.g. Mihalcea and Tarau, 2004; Florescu and Caragea, 2017). In contrast to MDKE, the suggested queries extraction subtask is a new paradigm that updates "keyphrases" with respect to session history. While previous methods for keyphrase extraction could potentially be adapted for our dynamic setting, we choose to focus in this work on a deep RL architecture for suggested queries that resonates our model for query-assisted summarization and allows sharing insights between the models.

3 Query-Assisted Summarization Model

The subtask of query-assisted summarization covers two main components of the INTSUMM task: the generators of an initial summary and of query-responses. The initial summary concisely specifies some central issues from the input topic (not biased on a query) to initiate the user's understanding of the topic and to motivate further exploration. Then, for each user submitted query, the query-response generator non-redundantly expands on the previously presented information with topically salient responses that are also biased around the query. We next formally define the subtask and then describe our RL model for it.

3.1 Subtask Formulation

The input to the query-assisted summarization subtask is tuple $(\mathcal{D}, q, \boldsymbol{E}^{\text{in}}, m)$, such that: $\mathcal{D}$ is a document set on a topic where the j-th sentence in the concatenation of $\mathcal{D}$'s documents is denoted s_j; q is a query, and can be empty (denoted _) for an unbiased generic summary; $\boldsymbol{E}^{\text{in}} = \{e_1^{\text{in}}, ..., e_k^{\text{in}}\}$ is a sequence of sentences from $\mathcal{D}$ termed the *history*, containing texts previously output in the session; and m is the number of sentences to output. The output is sentence sequence $\boldsymbol{E}^{\text{out}} = \{e_1^{\text{out}}, ..., e_m^{\text{out}}\}$ from $\mathcal{D}$ (extractive summarization). When inputting $(\mathcal{D}, \_, \{\}, m)$, the output is a generic summary of m sentences, that can serve as the initial summary; and when q and $\boldsymbol{E}^{\text{in}}$ are not empty, the output is an expansion on $\boldsymbol{E}^{\text{in}}$ in response to q, containing new salient information biased on q.

$\mathcal{D}$ is paired with a set of generic reference summaries $\mathcal{R}$, which is used for training or as a part of the evaluation effort.

3.2 Model Architecture

Our query-assisted summarization model, M_{Summ}, is autoregressive, outputting the requested number of summary sentences one-by-one. At time step t, a sentence e_t^{out} is output according to the current query and an encoding of the summary-so-far $\boldsymbol{E}_t = \{e_1^{\text{in}}, ..., e_k^{\text{in}}, e_1^{\text{out}}, ..., e_{t-1}^{\text{out}}\}$ to prevent information repetition. At inference time, M_{Summ}

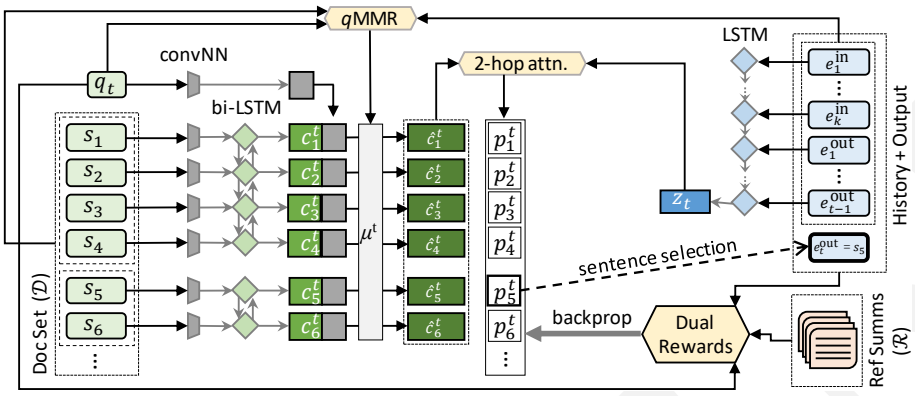

Figure 2: The M_{Summ} query-assisted summarization model architecture. Contextual sentence embeddings are concatenated to the current query embedding. The sentence+query representation is softly attended with a transformed query-focused MMR score, and a sentence selection distribution is obtained with a two-hop attention mechanism, considering a summary-so-far representation. A dual-reward mechanism, using the reference summaries and query, optimizes a policy to train the model for summary content quality and sentence-to-query resemblance. At inference time, an initial summary is generated with empty $\boldsymbol{E}^{\text{in}}$ and q_t-s, while for an expansion they are not empty.

outputs the summary sentences with the given query and history (possibly empty). At train time, we emulate a session by invoking M_{Summ} with a sequence of differing queries, $\mathcal{Q} = \{q_1, q_2, ..., q_m\}$, for which to generate the corresponding sequence of output sentences. I.e., output sentence e_t^{out} is biased on query q_t and the summary-so-far $\boldsymbol{E}_t$ at time step t. We next describe the architecture[3] of M_{Summ}, also illustrated in Figure 2.

Sentence encoding. The first step of the model is hierarchically encoding the sentences of the document set $\mathcal{D}$ to obtain contextualized representation $\boldsymbol{c}_j$ for sentence s_j $\forall j$. A CNN (Kim, 2014) encodes s_j on the sentence level and then a bi-LSTM (Huang et al., 2015) forms representation $\boldsymbol{c}_j$ on the document level, given the CNN encodings.

Query encoding. Additionally, at each time step t we prepare sentence+query representations $\boldsymbol{c}_j^t = \boldsymbol{c}_j \oplus \text{CNN}(q_t)$, i.e., obtained by concatenating a sentence representation and the CNN-encoding of the current query. This sentence+query representation influences the relevance of a sentence with respect to the current input query.

Query-MMR score weighting. MMR has been shown to be effective in MDS, where information repeats across documents. It aims to select a salient sentence for a summary, that is non-redundant to previous summary sentences. We extend standard MMR so that the importance of the sentence is in regards to both the document set *and* the query. Formally, the *query-focused MMR* function defines a score m_j^t for each s_j at time step t as follows:

[3]In general, the implementation choices weighed in the speed at which the full input document set can be processed. In comparison to other techniques (some of which are more recent), these choices gave as good or better results at lower latency. Alternative architectural choices and their behavior are discussed in Appendix B.

$$m_j^t = \lambda \cdot \textsc{BiSim}(s_j, \mathcal{D}, q_t) - (1-\lambda) \cdot \max_{e \in \boldsymbol{E}_t} \textsc{Sim}(s_j, e) \quad (1)$$

$$\textsc{BiSim}(s_j, \mathcal{D}, q_t) = \beta \cdot \textsc{Sim}(s_j, \mathcal{D}^{\oplus}) + (1-\beta) \cdot \textsc{Sim}(s_j, q_t) \quad (2)$$

where $\lambda \in [0,1]$ balances salience and redundancy and $\beta \in [0,1]$ balances a sentence's salience within its document set and its resemblance to the current query. $\textsc{Sim}(x, y)$ measures the similarity of texts x and y, and $\mathcal{D}^{\oplus}$ is a fully concatenated version of document set $\mathcal{D}$. Following findings of Mao et al. (2020), SIM computes cosine similarity between the two compared texts' TF-IDF vectors. Redundancy to previous sentences is computed as the highest similarity-score against any of the previous sentences. We set $\lambda = 0.6$ (following Lebanoff et al., 2018) and $\beta = 0.5$ (see Appendix B.3).

The query-focused MMR scores are incorporated into M_{Summ} by softly attending on the sentence representations with their respective translated query-focused MMR scores:

$$\boldsymbol{\mu}^t = \text{softmax}(\text{MLP}(\boldsymbol{m}^t)) \quad (3)$$

$$\hat{\boldsymbol{c}}_j^t = \mu_j^t \boldsymbol{c}_j^t \quad (4)$$

State representation. At time t, a representation $\boldsymbol{z}_t$ of the summary-so-far is computed by applying an LSTM encoder on $\{\boldsymbol{c}_{\text{idx}(e_1^{\text{in}})}, ..., \boldsymbol{c}_{\text{idx}(e_k^{\text{in}})}, \boldsymbol{c}_{\text{idx}(e_1^{\text{out}})}, ..., \boldsymbol{c}_{\text{idx}(e_{t-1}^{\text{out}})}\}$, i.e., on the plain sentence representations of $\boldsymbol{E}_t$, where $\text{idx}(e)$ is the index of sentence e. Then, a state representation $\boldsymbol{g}_t$ considers $\boldsymbol{z}_t$ and all sentence representations with the glimpse operation (Vinyals et al., 2016):

$$a_j^t = \boldsymbol{v}_1 \tanh(\boldsymbol{W}_1 \hat{\boldsymbol{c}}_j^t + \boldsymbol{W}_2 \boldsymbol{z}_t) \quad (5)$$

$$\boldsymbol{\alpha}^t = \text{softmax}(\boldsymbol{a}^t) \quad (6)$$

$$\boldsymbol{g}^t = \sum_j \alpha_j^t \boldsymbol{W}_1 \hat{\boldsymbol{c}}_j^t \quad (7)$$

where $\boldsymbol{v}_1$, $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are model parameters, and $\boldsymbol{a}^t$ represents the vector composed of a_j^t.

Finally, a sentence s_j at time t is assigned a selection probability $\text{softmax}(p^t)_j$ such that:

$$p_j^t = \begin{cases} \boldsymbol{v}_2 \tanh(\boldsymbol{W}_3 \hat{\boldsymbol{c}}_j^t + \boldsymbol{W}_4 \boldsymbol{g}^t) & \text{if } s_j \notin \boldsymbol{E}_t \\ -\infty & \text{otherwise} \end{cases} \quad (8)$$

where $\boldsymbol{v}_2$, $\boldsymbol{W}_3$ and $\boldsymbol{W}_4$ are model parameters.

Reinforcement learning. As M_{Summ}'s goal is to incrementally generate a query-assisted summary, it should strive to optimize (1) non-redundant salient-sentence extraction and (2) query-to-sentence similarity, that can be appraised with ROUGE (Lin, 2004) and text-similarity metrics, respectively. A policy gradient-based RL approach (Williams, 1992) allows optimizing on such non-differentiable metrics. Specifically, we adopt the Advantage Actor Critic method (Mnih et al., 2016) for policy learning, and a dual-reward procedure (Pasunuru and Bansal, 2018) to alternate between the summary and query-similarity rewards.

At time step t, for selected sentence e_t^{out} (based on $\text{softmax}(p^t)$), reward r_t is computed and weighted into M_{Summ}'s loss function. The reward function alternates,

from one train batch to the next, between $\text{ROUGE}_\Delta(e_t^{\text{out}}, \boldsymbol{E}_t, \mathcal{R})$ and $\text{QSIM}(e_t^{\text{out}}, q_t)$. The former computes the ROUGE difference before adding e_t to $\boldsymbol{E}_t$ and after:

$$\text{ROUGE}_\Delta(e_t^{\text{out}}, \boldsymbol{E}_t, \mathcal{R}) = \\ \text{ROUGE}((\boldsymbol{E}_t \cup e_t^{\text{out}})^\oplus, \mathcal{R}) - \text{ROUGE}(\boldsymbol{E}_t^\oplus, \mathcal{R}) \quad (9)$$

A larger ROUGE_Δ value implies that e_t concisely adds more information onto $\boldsymbol{E}_t$, with respect to topic reference summaries $\mathcal{R}$. We use ROUGE-1 F_1 as the ROUGE function here. The query-similarity reward function

$$\text{QSIM}(e_t^{\text{out}}, q_t) = \\ \text{avg}(\text{SEMSIM}(e_t^{\text{out}}, q_t), \text{LEXSIM}(e_t^{\text{out}}, q_t)) \quad (10)$$

computes an average of semantic and lexical similarities between the selected sentence and corresponding query. SEMSIM computes the cosine similarity between the average of word embeddings (spaCy: Honnibal and Montani, 2021) of e_t^{out} and that of q_t. For lexical similarity,

$$\text{LEXSIM}(e_t^{\text{out}}, q_t) = \\ \text{avg}(R_1^p(e_t^{\text{out}}, q_t), R_2^p(e_t^{\text{out}}, q_t), R_L^p(e_t^{\text{out}}, q_t)) \quad (11)$$

is the average of ROUGE-1, 2 and L precision scores between sentence and query. By alternating between the two rewards, we train a sentence-selection policy in M_{Summ} to balance summary informativeness and adherence to queries.

Overall system. Our M_{Summ} model adopts its base architecture from Mao et al. (2020) (for generic MDS). Chiefly, we modify their model for handling an input query-sequence and a sentence history, and employ a different summarization reward function. The query is incorporated in the sentence representation, in the new query-focused MMR function and in the dual-reward mechanism.

3.3 Model Training

Pre-training. To provide a warm start for training M_{Summ}, a reduced version of M_{Summ} is first pre-trained for generic extractive *single-document* summarization using the large-scale CNN/Daily Mail corpus (Hermann et al., 2015), as proposed by Chen and Bansal (2018). The reduced model pre-trains the full model for contextual sentence representation and for salient-sentence selection in the single-document generic setting. See Appendix B.1 for precise technical details.

Training data. After pre-training the reduced version of M_{Summ}, we train the full model using the DUC 2007 MDS dataset, with modifications for our query-assisted MDS task. The dataset includes 45 topics (split into 35/10 train/val), each containing 25 documents and 4 reference summaries.

For each topic, we generate an "oracle" extractive summary by greedily aggregating 10 sentences from $\mathcal{D}$, that maximizes the ROUGE_Δ-1 recall against $\mathcal{R}$. Then for each sentence, we extract a bi- or trigram that is most lexically-unique to the sentence, in comparison to all other sentences in $\mathcal{D}$. This yields a sequence of 10 "queries" that could easily render the corresponding oracle summary. The intuition for this approach

is that it would teach M_{Summ} that it is worthwhile to consider a given query when selecting a sentence that is informative with respect to the reference summaries. This further assists in fulfilling the dual requirements of selecting a globally informative sentence that also adheres to the query.[4] Appendix B.3 discusses usage of different query types for training.

Validation metric. As the interactive session progresses, a recall curve emerges, that maps the ROUGE recall score (here ROUGE-1) versus the expanding summary token-length. Once the session halts, the area under the curve indicates the efficacy of the session for information exposure. A higher value implies faster unveiling of salient information. Normalizing by the final summary length allows approximate comparability between different length sessions. We hence use the average (over topics) length-normalized area under the recall curve for validating the training progress.

4 Suggested Queries Extraction Model

4.1 Subtask Formulation

We now consider the second subtask of INTSUMM: generating lists of suggested queries. The list is regenerated after every interaction, to yield queries that focus on sub-topics that were not yet explored.

Reusing the notations of M_{Summ} in §3, we define a model, M_{Sugg}, for suggested queries list generation, that receives an input tuple $(\mathcal{D}, \boldsymbol{E}^{\text{in}}, m)$ (notice that a query is not needed here). Here, the j-th **phrase** in $\mathcal{D}$ is denoted ρ_j, when the documents in $\mathcal{D}$ are concatenated, and accordingly, history $\boldsymbol{E}^{\text{in}}$ is a list of phrases extracted from the session's current accumulated summary. m is the number of suggested queries to output. The model outputs phrase sequence $\boldsymbol{E}^{\text{out}} = \{e_1^{\text{out}}, e_2^{\text{out}}, ..., e_m^{\text{out}}\}$ from $\mathcal{D}$, accounting for history $\boldsymbol{E}^{\text{in}}$. As in M_{Summ}'s setting, $\mathcal{D}$ is paired with a set of generic reference summaries $\mathcal{R}$.

4.2 Model Architecture

We adopt and adjust the architecture in §3.2 for this subtask. Similar to M_{Summ}, M_{Sugg} selects input units one-by-one considering a history, with the main difference being the absence of query injection. Additionally, inputs and outputs are processed on the phrase- rather than the sentence level.

Phrase and state representation. For the given document set, all noun phrases are extracted using a standard part-of-speech regular expression method (Mihalcea and Tarau, 2004; Wan and Xiao, 2008).

We obtain document-level contextual phrase embeddings, c_j for phrase ρ_j, with the CNN and bi-LSTM networks, and softly attend the embeddings with a *standard* MMR score:

[4] Seemingly, the most natural approach would be to train the model with queries from real sessions (collected using a different system). However, a session's queries are dependent on outputs previously produced by the used system. Hence, these do not benefit the training process more than a synthesized sequence of queries.

$$m_j^t = \lambda \cdot \textsc{Sim}(\rho_j, \mathcal{D}^{\oplus}) - (1-\lambda) \cdot \max_{e \in \boldsymbol{E}_t} \textsc{Sim}(\rho_j, e) \quad (12)$$

The MMR-based phrase representations then pass through the glimpse attention procedure, which culminates in the phrase probability distribution for selecting the next output phrase.

Reinforcement learning. The policy in M_{Sugg} is trained with a single reward function that measures how prominent the selected phrase is within the reference summaries, and how different it is from previously seen phrases. Formally, at time step t, the reward r_t of selected phrase e_t^{out} is:

$$r_t = \textsc{pf}(e_t^{\text{out}}, \mathcal{R}) - \gamma_1 \cdot \textsc{pfMax}(e_t^{\text{out}}, \boldsymbol{E}^{\text{in}}, \mathcal{R}) - \gamma_2 \cdot \textsc{pfMax}(e_t^{\text{out}}, \boldsymbol{E}_t \setminus \boldsymbol{E}^{\text{in}}, \mathcal{R}) \quad (13)$$

$$\textsc{pf}(e_t^{\text{out}}, \mathcal{R}) = \text{avg}_{r \in \mathcal{R}} \left(\text{avg}_{w \in e_t^{\text{out}}} \textsc{tf}(w, r) \right) \quad (14)$$

$$\textsc{pfMax}(e_t^{\text{out}}, \boldsymbol{L}, \mathcal{R}) = \max_{e \in \boldsymbol{L}} \textsc{pf}(e_t^{\text{out}} \cap e, \mathcal{R}) \quad (15)$$

where $\textsc{tf}(w, r)$ is the relative frequency of word w in reference summary r. Namely, PF computes the average term frequency of a phrase over its words and across the reference summaries, as an estimate of the phrase importance within the topic. PFMAX computes the highest PF against a list of phrases, which is used to lower the reward of a phrase that is redundant to phrases used earlier. Different weights are given to the PFMAX against the input history (γ_1) and that of the phrases output so far (γ_2).

4.3 Model Training

Similarly to M_{Summ}, we first **pre-train** the base model to get a warm start on embedding formation and salience detection. The reduced architecture of M_{Summ} and M_{Sugg} for pre-training are identical.

We use the same DUC 2007 **training data**, with document sets and reference summaries, and additionally prepare three "histories" per topic: one empty and two non-empty. An empty history mimics generating a session's initial list of suggested queries, while a non-empty history trains the model to consider previously known information. Training with two non-empty histories per topic prepares a model for varying informational states. These are curated from a generic summary (from a trained M_{Summ} model) that is truncated at two random sentence-lengths between 1 and 12. Overall, the model is trained on three versions of each topic, each time with a different history.

Similarly to M_{Summ}, **validation** is guided by the average normalized area under the recall curve. Here, the accumulating r_t scores from Equation 13 are used as the recall of the expanding suggested queries list. I.e., a higher reward means better suggested queries are output earlier. The AUC is normalized with the total token-length of all suggested queries to mitigate for lengthy phrase extractions.

PREPRINT

M_{Summ} Model Configuration					Simulation Results († = informativeness metric, R_1 = ROUGE-1)				
#	Query in Encoding	Query in MMR	Query in Reward (Dual)	Query in MMR at Inference	† Initial Summ Norm $R_1^{F_1}$ ($\times 10^{-3}$)	Initial Summ Token-Length	† Expansion Norm $R_{1\Delta}^{\text{recall}}$ ($\times 10^{-3}$)	Expansion Token-Len.	QSIM Query Responsiveness
i.	yes	yes	yes	yes	3.09 (±0.11)	86.7 (5.6)	0.913 (±0.055)	49.7 (2.7)	0.488 (±0.021)
ii.	yes	yes	no	yes	3.04 (±0.12)	87.4 (8.3)	0.897 (±0.054)	49.4 (2.7)	0.482 (±0.022)
iii.	yes	no	no	yes	3.00 (±0.14)	88.0 (8.7)	0.892 (±0.058)	50.1 (2.9)	0.479 (±0.020)
iv.	no	yes	yes	yes	2.98 (±0.17)	85.1 (6.5)	0.892 (±0.057)	51.3 (2.8)	0.462 (±0.025)
v.	no	no	no	yes	3.05 (±0.12)	85.4 (8.1)	0.955 (±0.046)	51.8 (2.9)	0.423 (±0.027)
vi.	no	no	no	no	3.05 (±0.12)	85.4 (8.1)	0.988 (±0.056)	52.8 (4.0)	0.311 (±0.023)
S_2 Baseline (Shapira et al., 2021b)					2.75 (±0.20)	85.1 (21.8)	0.799 (±0.040)	49.1 (2.8)	0.601 (±0.021)

Table 1: Simulation results on previously collected sessions, yielding a partial ablation of our M_{Summ} model, and the results on the baseline system which was originally used to collect those sessions. Intervals at 95% confidence.

5 Experiments

We ran several experiments for the assessment of our M_{Summ} and M_{Sugg} models, applying the INTSUMM evaluation framework of Shapira et al. (2021b). The goals of the experiments are to compare varying configurations of our models and to evaluate against an INTSUMM baseline system. The experiments include both simulations and interactive sessions with human users.

5.1 Compared Algorithms

The M_{Summ} model architecture (§3.2) has several configurable components: encoding the query into sentences, considering the query in the MMR function (both at train and inference time), and the dual reward mechanism. We compared several variations of these using simulations, presented in §5.2.

In addition, we compare, both via simulations (§5.2) and real sessions (§5.3), against the (better-performing) baseline system in (Shapira et al., 2021b), named S_2. S_2's initial summary algorithm is TextRank, and the query-response generator extracts sentences via lexical+semantic similarity to the query, somewhat resembling QSIM in Equation 10, fully neglecting the summary-so-far, in contrast to M_{Summ}. S_2's suggested queries list contains TextRank's top salient topic phrases. Since these too do not account for the summary-so-far, they are computed at the session beginning and are not updated along the session, in contrast to M_{Sugg}.

5.2 Simulated Experiments

The INTSUMM task involves human users by definition. Nevertheless, running on simulated query lists and session histories is pertinent for efficient system evaluation and comparison of methods.

To simulate the query-assisted summarization algorithms, we utilize the real sessions recorded by Shapira et al. (2021b): 3-4 user sessions on 20 topics from DUC 2006 collected with S_2. In our simulation, each summary-so-far from a recorded session is fed as input to the system together with the following recorded user query. We then measure $R_{1\Delta}^{\text{recall}}$ (difference of ROUGE-1 recall incurred by the query response compared with the input summary-so-far). Additionally, we use $R_1^{F_1}$ (ROUGE-1 F_1) for initial summary informativeness. Both are measured w.r.t. the reference summaries, normalized by the output length, and averaged per session recording, and then over all sessions and topics, to get an overall system informativeness score. We also measure system query-responsiveness using the QSIM metric.

PRELIMINARY

Metric	Ours	S_2 Baseline
R^r_1 AUC @ $[106, 250]$	43.42 ($\pm$1.54)	40.01 ($\pm$1.52)
$R^{F_1}_1$ @ initial	0.256 ($\pm$.011)	0.231 ($\pm$0.014)
$R^{F_1}_1$ @ 250	0.396 ($\pm$.015)	0.378 ($\pm$.015)
QSIM query-resp.	0.471 ($\pm$.028)	0.623 ($\pm$.023)
Manual query-resp.	3.96 ($\pm$0.19)	4.03 ($\pm$0.23)
Manual UMUX-Lite	78.9 ($\pm$2.5)	78.6 ($\pm$3.4)

Table 2: Average scores of our system (configuration *v*) and baseline system S_2 on actual user sessions. Our system exposes topical information better, while the user experience is very good despite the slight degradation in query-responsiveness. Intervals at 95% confidence.

Table 1 presents a representative partial ablation of the M_{Summ} model. All variants were configured to output sentences of up to 30 tokens, initial summaries are 75 tokens, and query responses are 2 sentences. Configurations *i-iv* use the query in training, while *v* and *vi* do not. Each configuration is measured for informativeness (columns marked with †), and for query-responsiveness (QSIM column). Out of configurations *i-iv*, config. *i*, where we employ all mechanisms for query inclusion, yields the best overall scores in both informativeness and query-responsiveness, despite the inherent tradeoff between the two. In the second set of configurations (*v-vi*), we observe that ignoring the query at train time substantially degrades query-responsiveness, and this is expectedly further exacerbated when also ignoring the query at inference time. However, disregarding the query gives more informative expansions with respect to reference summaries, since the model was trained only to optimize content informativeness, and is less likely to sidetrack to the query-related information.

Compared to S_2 (last row), our model significantly improves informativeness. Query-responsiveness is better in the S_2 baseline since its query-response generator simply invokes a function similar to QSIM, but for the price of lower informativeness. Still, this does not lead to inferior overall user experience, see § 5.3.

5.3 Real Session Collection and Evaluation

We collect real user sessions via controlled crowdsourcing (which provides high quality work, see Appendix D) with the use of an INTSUMM web application[5] running either our M_{Summ}+M_{Sugg} models or the S_2 baseline algorithms, enabling a comparative assessment of the two systems. Notably, our algorithms have the low latency required for the interactive setting (Attig et al., 2017), i.e., responding almost immediately.[6]

Using the DUC 2006 INTSUMM test set, we prepared two complementing user sets of 20 topics, each with 10 of the topics to be run on our system and the other 10 on the baseline. We apply the evaluation metrics of Shapira et al. (2021b): (1) The area under the sessions' ROUGE recall curves, in a common word-length interval across all sessions and topics, which demonstrates how fast salient information is exposed in sessions. (2) ROUGE F_1 at the initial summary and at 250 tokens, that indicate how effectively the interactive system can generate summaries at pre-specified, comparable lengths. (3)

[5]Minimally modified from (Shapira et al., 2021b) to support updating the suggested queries list after each interaction.

[6]M_{Summ} generates summaries in under a second and M_{Sugg} prepares the list of suggested queries in a few seconds. See Appendix E.2 for more details.

Manually assigned query-responsiveness score (1 to 5 scale), which expresses how well users think the system responded to their requests. And (4) manual UMUX-Lite (Lewis et al., 2013) score for system usability (effectiveness and ease of use), where 68 is considered "acceptable" and 80.3 is considered "excellent". We also measure automatic query-responsiveness with QSIM.[7]

We conducted two such comparative collection and assessment experiments, either employing M_{Summ} configuration *v* or *i*, namely the best of the two configuration sets. In both cases, the M_{Sugg} model used was set with $\gamma_1 = 0.5$ and $\gamma_2 = 0.9$ after some hyperparameter tuning (Appendix B.4). The first experiment (with configuration *v*) is described here, and the other in Appendix E.1.

We hired 6 qualified workers using the controlled crowdsourcing procedure, and collected 2-3 sessions per topic per system (111 total sessions). In the sessions, users explore their given topic by submitting queries with a common generic informational goal in mind (Appendix D).

Overall system assessment. Table 2, presenting average scores over the collected sessions, shows that our system is significantly more effective for exposing salient information, as depicted in the first three rows. Users indicate a slight degradation in query-responsiveness of our system, consistent with QSIM scores (row 4-5). Note that the observed difference in QSIM scores, between simulations and user sessions, partly stems from the fact that they were computed over different sets of queries. The varying queries issued by the users in user sessions form a less stable query responsiveness comparison than the one in Table 1, where QSIM scores are computed using consistent queries for all systems. Despite the gap in QSIM scores between our system and S_2 in Table 2, the overall usability scores are slightly better (last row). This may suggest that users appreciate the informativeness of the produced summary even when they are aware that the summary is less biased on their queries; thus our system improves informativeness while still providing a favorable user experience.

Assessment of suggested queries functionality. We analyzed the types of queries users submitted throughout their sessions, to assess the utility of updating suggested queries, with M_{Sugg}, as opposed to a static list of suggestions, with S_2. To that end, we tallied suggested query clicks and query submissions via other modes, binning the tallies to three sequential temporal segments within their respective sessions (Appendix E.3). We found that, on average, the usage of suggested query clicks increased by ~13% when nearing the end of a session with M_{Sugg}, and conversely decreased by ~24% with S_2. While the decrease in use of the static list is expected, since appealing queries are likely exhausted earlier in a session, it is encouraging to witness the usefulness of updated queries as the session progresses. This behavior suggests that the updated list contains suggested queries that are indeed engaging for learning more about the topic.

6 Conclusion

Interactive summarization for information exploration is a task that requires compliance to user requests and session history, while comprehensively handling a large input document set. These requirements pose a challenge for advanced text processing

[7] While QSIM is a reasonable automatic measure for estimation of query-responsiveness, it is left for future work to assess its true reliability for such use.

methods due to the need for fast reaction time. We present novel deep reinforcement learning based algorithms that answer to the task requirements, improving salient information exposure while satisfying user queries and keeping user experience positive.

We note that while M_{Summ} is designed for the INTSUMM task, it may potentially be serviceable for standard MDS, QFS, update summarization and combinations thereof. This can be accommodated by a proper choice of input, e.g., QFS can be addressed by giving M_{Summ} as input a query, an empty history and target summary length. In future work, we may study the performance of our solutions for such tasks, as well as strive to further improve their performance on both ends of the INTSUMM task – selecting topically salient information and responding to user queries.

Acknowledgements

We thank the anonymous reviewers for their constructive comments and suggestions. This work was supported in part by Intel Labs; by the Israel Science Foundation (grants no. 2827/21 and 2015/21); by a grant from the Israel Ministry of Science and Technology; by the NSF-CAREER Award #1846185; and by a Microsoft PhD Fellowship.

References

Shaun Anderson. 2020. How Fast Should A Website Load? https://www.hobo-web.co.uk/your-website-design-should-load-in-4-seconds. Accessed: 2021-09-25.

Christiane Attig, Nadine Rauh, Thomas Franke, and Josef F. Krems. 2017. System Latency Guidelines Then and Now – Is Zero Latency Really Considered Necessary? In *Engineering Psychology and Cognitive Ergonomics: Cognition and Design*, pages 3–14, Cham. Springer International Publishing.

Tal Baumel, Raphael Cohen, and Michael Elhadad. 2014. Query-Chain Focused Summarization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 913–922, Baltimore, Maryland. Association for Computational Linguistics.

Tal Baumel, Raphael Cohen, and Michael Elhadad. 2016. Topic Concentration in Query Focused Summarization Datasets. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2573–2579. AAAI Press.

Tal Baumel, Matan Eyal, and Michael Elhadad. 2018. Query Focused Abstractive Summarization: Incorporating Query Relevance, Multi-Document Coverage, and Summary Length Constraints into seq2seq Models. *arXiv preprint arXiv:1801.07704*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. *arXiv preprint arXiv:2004.05150*.

John Brooke. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7.

Jaime Carbonell and Jade Goldstein. 1998. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in*

Information Retrieval, SIGIR '98, page 335–336, New York, NY, USA. Association for Computing Machinery.

Yen-Chun Chen and Mohit Bansal. 2018. Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686, Melbourne, Australia. Association for Computational Linguistics.

Janara Christensen, Stephen Soderland, Gagan Bansal, and Mausam. 2014. Hierarchical Summarization: Scaling Up Multi-Document Summarization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 902–912, Baltimore, Maryland. Association for Computational Linguistics.

Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-Based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 22(1):457–479.

Corina Florescu and Cornelia Caragea. 2017. PositionRank: An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1115, Vancouver, Canada. Association for Computational Linguistics.

Dan Gillick and Yang Liu. 2010. Non-Expert Evaluation of Summarization Systems is Risky. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 148–151, Los Angeles. Association for Computational Linguistics.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring Content Models for Multi-Document Summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370, Boulder, Colorado. Association for Computational Linguistics.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1693–1701, Cambridge, MA, USA. MIT Press.

Eran Hirsch, Alon Eirew, Ori Shapira, Avi Caciularu, Arie Cattan, Ori Ernst, Ramakanth Pasunuru, Hadar Ronen, Mohit Bansal, and Ido Dagan. 2021. iFacetSum: Coreference-based Interactive Faceted Summarization for Multi-Document Exploration. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 283–297, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Matthew Honnibal and Ines Montani. 2021. Linguistic Features - spaCy Usage Documentation. `https://spacy.io/usage/linguistic-features#vectors-similarity`. Accessed: 2021-11-01.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *CoRR*, abs/1508.01991.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Sayali Kulkarni, Sheide Chammas, Wan Zhu, Fei Sha, and Eugene Ie. 2021. CoMSum and SIBERT: A Dataset and Neural Model for Query-Based Multi-document Summarization. In *Document Analysis and Recognition – ICDAR 2021*, pages 84–98, Cham. Springer International Publishing.

Md Tahmid Rahman Laskar, Enamul Hoque, and Jimmy Xiangji Huang. 2020. WSL-DS: Weakly Supervised Learning with Distant Supervision for Query Focused Multi-Document Abstractive Summarization. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5647–5654, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Logan Lebanoff, Kaiqiang Song, and Fei Liu. 2018. Adapting the Neural Encoder-Decoder Framework from Single to Multi-Document Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4131–4141, Brussels, Belgium. Association for Computational Linguistics.

Anton Leuski, Chin-Yew Lin, and Eduard Hovy. 2003. iNeATS: Interactive Multi-Document Summarization. In *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*, pages 125–128, Sapporo, Japan. Association for Computational Linguistics.

James R. Lewis, Brian S. Utesch, and Deborah E. Maher. 2013. UMUX-LITE: When There's No Time for the SUS. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, page 2099–2102, New York, NY, USA. Association for Computing Machinery.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Jimmy Lin, Salman Mohammed, Royal Sequiera, Luchen Tan, Nimesh Ghelani, Mustafa Abualsaud, Richard McCreadie, Dmitrijs Milajevs, and Ellen Voorhees. 2017. Overview of the TREC 2017 Real-Time Summarization Track.

Yuning Mao, Yanru Qu, Yiqing Xie, Xiang Ren, and Jiawei Han. 2020. Multi-document Summarization with Maximal Marginal Relevance-guided Reinforcement Learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1737–1751, Online. Association for Computational Linguistics.

PREPRINT

Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2014. Incremental Update Summarization: Adaptive Sentence Selection Based on Prevalence and Novelty. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, page 301–310, New York, NY, USA. Association for Computing Machinery.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA. PMLR.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-Reward Reinforced Summarization with Saliency and Entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 646–653, New Orleans, Louisiana. Association for Computational Linguistics.

Ramakanth Pasunuru, Asli Celikyilmaz, Michel Galley, Chenyan Xiong, Yizhe Zhang, Mohit Bansal, and Jianfeng Gao. 2021a. Data Augmentation for Abstractive Query-Focused Multi-Document Summarization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13666–13674.

Ramakanth Pasunuru, Mengwen Liu, Mohit Bansal, Sujith Ravi, and Markus Dreyer. 2021b. Efficiently Summarizing Text and Graph Encodings of Multi-Document Clusters. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4768–4779, Online. Association for Computational Linguistics.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence Level Training with Recurrent Neural Networks. In *ICLR*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Ori Shapira, Ramakanth Pasunuru, Ido Dagan, and Yael Amsterdamer. 2021a. Multi-Document Keyphrase Extraction: A Literature Review and the First Dataset. *arXiv preprint arXiv:2110.01073*.

Ori Shapira, Ramakanth Pasunuru, Hadar Ronen, Mohit Bansal, Yael Amsterdamer, and Ido Dagan. 2021b. Extending Multi-Document Summarization Evaluation to the Interactive Setting. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 657–677, Online. Association for Computational Linguistics.

PREPRINT

Ori Shapira, Hadar Ronen, Meni Adler, Yael Amsterdamer, Judit Bar-Ilan, and Ido Dagan. 2017. Interactive Abstractive Summarization for Event News Tweets. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 109–114, Copenhagen, Denmark. Association for Computational Linguistics.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order Matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*.

Xiaojun Wan and Jianguo Xiao. 2008. Single Document Keyphrase Extraction Using Neighborhood Knowledge. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, page 855–860. AAAI Press.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. 2020. Heterogeneous Graph Neural Networks for Extractive Document Summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6209–6219, Online. Association for Computational Linguistics.

Ronald J Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3):229–256.

Wen Xiao, Iz Beltagy, Giuseppe Carenini, and Arman Cohan. 2021. PRIMER: Pyramid-based Masked Sentence Pre-training for Multi-document Summarization. *arXiv preprint arXiv:2110.08499*.

Yujia Xie, Tianyi Zhou, Yi Mao, and Weizhu Chen. 2020. Conditional Self-Attention for Query-based Summarization. *arXiv preprint arXiv:2002.07338*.

PREPRINT

A Ethical Considerations

Datasets. The DUC 2006 and 2007 datasets were obtained according to the DUC website (`duc.nist.gov`) requirements. It was not possible for others to reconstruct the document sets and reference summaries of the dataset from the crowdsourcing tasks.

The datasets are composed of new articles mainly from the late 1990s from large news outlets, compiled by NIST. All data exposed by our systems are directly extracted from those articles. For extraction, we do not intentionally add in any rules for ignoring or boosting certain information due to an opinion.

Crowdsourcing. Due to the need for English speaking workers, a location filter was set on the Amazon Mechanical Turk (`https://www.mturk.com`) tasks for the US, UK and Australia. All tasks paid according to a $10 per hour wage, according to the estimated required time of each task. The payment was either paid per assignment, or as a combination with a bonus.

Compute resources. Our M_{Summ} and M_{Sugg} models required between 2 and 20 hours of training (usually around 4 hours), depending on the configuration. We trained on one NVIDIA GeForce GTX 1080 Ti GPU with 11GB memory. The pretrained base model was trained once and reused in all subsequent training. Outputting at inference time is computationally cheap: M_{Summ} runs upto about 1 second, but mostly in a few hundred milliseconds, and M_{Sugg} runs upto about 7 seconds, but mostly in under 4 seconds. Training with a batch size of 8 used about 3GB GPU memory for M_{Summ}, and about 9GB memory for M_{Sugg} (since there are many more input units per document set, i.e., all noun phrases versus sentences).

B Implementation Details

B.1 Pre-training Technicalities

To provide a warm start for training M_{Summ} and M_{Sugg}, a reduced version of the models, which is the same for both, is first pre-trained for generic extractive *single-document* summarization using the CNN/Daily Mail corpus (Hermann et al., 2015) with about 287k samples, as proposed by Chen and Bansal (2018). In this reduced model, $\hat{\boldsymbol{c}}_j^t$ is replaced by $\boldsymbol{c}_j$ in Equations 5, 7 and 8. Furthermore, there is a single reward function for learning the policy, computed per selected sentence e_t^{out} as ROUGE-L F_1 w.r.t. the (single) reference summary's sentence at index t. The reduced model pre-trains the full model for contextual sentence representation and for salient-sentence selection in the single-document generic setting. This allows training M_{Summ} and M_{Sugg} with a relatively small dataset for their final purposes.

B.2 Training Technicalities

Following (Mao et al., 2020), the pre-trained base model is the rnn-ext + RL model from Chen and Bansal (2018), and is trained like in Lebanoff et al. (2018). Both M_{Summ} and M_{Sugg} are further trained on our adjusted DUC 2007 data using an Adam optimizer with a learning rate of 5e-4 and no weight decay. A discount factor of 0.99 is used for the reinforcement learning rewards. The batch size was 8. Training was halted once 30 consecutive epochs did not improve the validation score.

The MMR function within our models uses TF-IDF vector cosine similarity for all SIM instances (in Equations 1 and 12). The TF-IDF vectorizer is initialized with the document set on which the MMR score is computed.

As is commonly practiced, selection of an output sentence/phrase e_t^{out} is done by sampling probability distribution $\boldsymbol{p}^t$ (in Equation 8) at train time, and by extracting the maximum scoring sentence/phrase at inference time.

The MLP in Equation 3 transforms the MMR score with a feed-forward network with one-hidden layer of dimension 80 following (Mao et al., 2020).

B.3 Query-Assisted Summarization Model

Model configurations. The architecture of the M_{Summ} model and its training allowed for much creativity in the configuration process. Other than the combinations mentioned in the paper in Table 1, we also experimented with other components. We list here many of the experiments, without formal results. Anecdotes are taken by looking at validation scores and some eyeballing.

(1) The β **value in the query-focused MMR** function in Equation 2, that impacts the weight of the query on a sentence versus the document set on the sentence. We tried out a few β values and mainly noticed that a value of 0.5 kept validation results more stable across configurations, or kept training time shorter. In our experiments, to cancel out this component (both at training and inference time), we simply set $\beta = 1$ so that the query is not considered.

(2) Different **summary reward functions**. ROUGE$_\Delta$ recall (instead of F_1) was also a good alternative, but gave somewhat less stable results across configurations. ROUGE (not as Δ) was also less stable with recall and F_1, and gave too short and irrelevant sentences with precision. We also tried sentence level ROUGE-L, like in (Mao et al., 2020), eventually outputting sentences that were much less compliant to queries.

(3) Using only the **query similarity reward** instead of the dual reward mechanism worked surprisingly well. This may be due to the queries on which the model was trained on. These queries were very relevant to the gold reference summaries, hence possibly implicitly providing a strong signal to salient sentences within the document set. Still, this was less productive than our final choice of reward.

(4) **Adding training data** (additional DUC MDS datasets) did not impact the results. Importantly, since DUC 2007 is most similar to the test DUC 2006 set, it seems to be more beneficial to include DUC 2007 in the training set.

(5) We also tried **representing the query in the input** by concatenating it's raw text to each input sentence before get the sentence representations.

(6) To **represent the sentences**, we also tried using average w2v vectors (Honnibal and Montani, 2021) and Sentence-BERT (Reimers and Gurevych, 2019) instead of the CNN network. These did not show any apparent improvements, and were notably expensive in terms of execution time.

(7) For the **sentence similarity in the query-MMR component**, we tried w2v and Sentence-BERT representations instead of TF-IDF vectors. Similarly to (6), they did not show improvements over using TF-IDF, and were very time-costly.

(8) Instead of the **dual-reward mechanism** that alternates between the two rewards from batch to batch, we also considered using a weighted average of the two rewards, consistently over all batches. Further experimentation is required on this technique for a more conclusive judgment.

PREPRINT

Queries used for training. The queries used for training the M_{Summ} model can affect the way it learns to respond to a query. Seemingly, the most natural approach would be to train the model as close as possible to the model's use at inference time. This would mean training M_{Summ} with queries from real sessions. However, a session's queries are dependent on outputs previously produced by the used system. It is therefore not certain that the sequence of queries from a different system's usage would necessarily benefit the training process when compared to a synthesized sequence of queries. I.e., it's not actually possible to train with "real sessions" in a conventional way.

Also, as stated in §3.3, the synthetic queries we eventually used direct the model to select salient sentences, which can support our dual-objectives: to get a sentence that is both globally salient to the topic, as well as responsive to the query. We tried training on other query types, synthesized with various keyphrase extraction techniques, and found that our final choice of queries more consistently gave good results overall.

Sentence length. We segmented the sentences in the document sets with the NLTK[8] sentence tokenizer, and removed sentences that contain quotes in them or do not end with a period.

During training we did not constrain the input sentences in any way. Some of the configuration experiments described above were done to check how the configuration might influence the length of the selected sentences. The best configurations, including the one we eventually used in our tests, tended to output somewhat longer sentences. Very long sentences are usually tedious for human readers, and we hence limited the sentences to 30 tokens at inference time. We found that this length constraint caused a slight degradation in simulation score results of our models, however still gave superior informativeness results compared to the baseline system.

Initial summary length. Sentences are accumulated until surpassing 75 tokens. Therefore summaries are not shorter than 75 tokens, but mostly not much longer than that.

B.4 Suggested Queries Extraction Model

Model configurations. We experimented with different configurations and hyper-parameter fine-tuning in the M_{Sugg} model as well. Tuning was performed in accordance to the validation scores and generic keyphrase extraction scores on the *MK-DUC-01* multi-document keyphrase extraction dataset of Shapira et al. (2021a).

(1) In the **reward function** in Equation 13, we set $\gamma_1 = 0.5$ and $\gamma_2 = 0.9$, i.e., the preceding output phrases are more strongly accounted for than the phrases in the session history. We tested several values between 0 and 1 for both hyper-parameters.

(2) We implemented **altered versions of the reward function** in Equation 13. Instead of phrase unigram-level frequency, we tried computing the full phrase frequency and computing partial phrase frequency, i.e., a maximal phrase template match within a reference summary. All functions tested were adequate overall, though our final choice of reward function was closest to the keyphrase extraction task unigram overlap metric, and gave best results overall.

(3) We also attempted **noun phrase extraction** with the spaCy[9] noun chunker and named entity recognizer. This combined approach misses some noun phrases within the

[8] https://www.nltk.org

[9] https://spacy.io/

text, but mainly is also more computationally heavy than the simple POS regex search that we use.

Extracting phrases with regular-expression. We extracted all noun-phrases from the document set by first mapping all tokens to their part-of-speech tags, and then applying a regular-expression chunker with regex: `{(<JJ>* <NN.*>+ <IN>)? <JJ>* <NN.*>+}`. These steps were accomplished with NLTK.

Phrase length. There is no limit set on the phrase length. We tried training and inferring with a phrase length constraint of 4 words, but found that this gave worse results overall.

History sentences to phrases. M_{Sugg} works on the *phrase* level. Meanwhile, in our extractive interactive setting, the history is a set of *sentences* already presented to the reader. Therefore, when extracting phrases from $\mathcal{D}$, we also link each phrase to its source sentence, and obtain $\boldsymbol{E}^{\text{in}}$ by compiling the phrases linked from the history sentences.

C Dataset Notes

While DUC 2006 (our test set) and 2007 (our train/validation set) were originally designed for the query-focused summarization task, they contain excessive topic concentration due to their long and descriptive topic queries (Baumel et al., 2016). Hence, their reference summaries can practically be considered generic.

D Session Collection

Controlled crowdsourcing protocol. We followed the controlled crowdsourcing protocol of Shapira et al. (2021b), which includes three steps: (1) a trap task for finding qualified workers; (2) practice tasks for explaining the interface and the purpose, as well as reiterating the generic information goal (see below) during exploration; (3) the session collection tasks. We used the Amazon Mechanical Turk HITs prepared by Shapira et al. (2021b).

Process cost. We paid \$0.40 for a trap task assignment, with 400 assignments released, and \$0.90 for a practice task assignment, with 28 assignments completed. The session collection assignment paid \$0.70, and a bonus mainly according to the length of interaction and additional comments provided. The bonus was between \$0.15 and \$0.35. A total of 111 sessions were recorded from 6 high quality workers. The full process cost about \$385 in total (including the Mechanical Turk fees) for the experiment including configuration *v* in Table 1.

The second round of experiments done on another variant of our system (configuration *i*) also included 28 practice tasks and compiled 10 final workers for a total of 180 collected sessions. Bonuses ranged from \$0.10 and \$0.40 on the session collection task. The full process cost of the second experiment was about \$475 in total (including the Mechanical Turk fees).

Session collection data preparation. We used the same 20 test topics as Shapira et al. (2021b), and created 2 batches of tasks. For the first batch, in alternating order of topics, 10 topics were paired with our system, and the other 10 were paired with the S_2 baseline. The other batch consisted of the complementing topic-system pairings. The workers were assigned a batch to work on such that half of the workers would work on each batch.

User informational goal. Since all sessions on a topic are evaluated against the same reference summaries, it is important that users aim to explore similar information. Following Shapira et al. (2021b), during practice tasks all users received a common informational goal to follow, so that the sessions are comparable. The emphasized description was: "produce an informative summary draft text which a journalist could use to best produce an overview of the topic".

Sessions filtering. In the first experiment, we filtered out 7 sessions that accumulated less than 250 tokens (from 2 different workers).

In the second experiment, 9 of the 10 workers completed at least 19 of the 20 topics One worker completed only 3 tasks and we disregarded those sessions. We also threw away 9 sessions that accumulated less than 250 tokens.

INTSUMM user interface. We used the same user interface developed by Shapira et al. (2021b) with a small change to enable suggested query list updates after each interaction (the interface was designed for the baselines, where the suggested-query list is static). To refrain from any possible user experience bias, we made the UI change as least apparent as possible.

System response time. M_{Summ} is able to generate summaries mostly in under a second, and M_{Sugg} prepares the list in a few seconds. The summary expansion is hence presented to the user almost immediately after query submission, and the suggested queries list is shown shortly afterwords, before the user finishes reading the expansion. The small delay in suggested query updating is hence almost unnoticed. The baseline summarizer responds similarly fast to M_{Summ}, making response-time difference unperceivable between the systems.

User feedback. Many of the users provided feedback about the session collection tasks after finishing their assignment batch. The overall impression was that there was no strong preference for either system. For example, one user wrote: *"I did not discern a consistent difference between the two systems that would result in having a clear preference."* This kind of comment was repeated by several users. Generally, there were no explicit comments about the difference in quality of the summary outputs, and topics were mostly scored or commented on similarly between the two systems since the complexity of the topic influenced the ability of the systems to comply to the user.

A comment in favor of updating suggested queries during interaction said: *"It was nice to have a new list as you progressed through the task, it helped me think of where to go next if I got stuck..."* This specific comment was written by a user that explored topics quite deeply. On the other hand, a user that explored more shallow liked that used suggested queries in the static list were marked: *"I did notice...the red font color on the used queries. That was helpful."* It therefore seems that updating suggested queries

are more useful for lengthy exploration, but for quick navigation, the static list might naturally be enough.

E More Results

E.1 Overall System Assessment

Metric	Ours	S_2 Baseline
R_1^r AUC @ $[106, 250]$	42.52 (± 1.65)	40.34 (± 1.40)
$R_1^{F_1}$ @ initial	0.260 ($\pm .011$)	0.231 (± 0.014)
$R_1^{F_1}$ @ 250	0.390 ($\pm .015$)	0.382 ($\pm .014$)
QSIM query-resp.	0.527 ($\pm .016$)	0.603 ($\pm .022$)
Manual query-resp.	3.66 (± 0.29)	3.79 (± 0.25)
Manual UMUX-Lite	73.8 (± 3.6)	75.8 (± 2.9)

Table 3: Average scores of our system and a baseline INTSUMM system on real user sessions, in an experiment using a different M_{Summ} configuration (configuration *i*) compared to the experiment of Table 2 (configuration *v*). Our system exposes topical information better, while the overall user experience is not significantly harmed. Intervals at 95% confidence level.

We conducted two comparative session collection and analysis experiments, one using M_{Summ} model configuration *v* (from Table 1), as presented in §5.3 and Table 2, and another with M_{Summ} model configuration *i*. As explained in §5.2, these two configurations performed best, on simulations, out of their respective configuration sets.

We show here results of the second experiment, where we used M_{Summ} model configuration *i*, with the same M_{Sugg} model as in the first experiment. The S_2 baseline was similarly used for comparison. We also kept the same AUC length limits (106 to 250 tokens) for easy comparability to Table 2. Table 3 shows the results. Here too, while less substantially, informativeness is improved with our system without significantly harming the user experience. Overall, it seems that users were somewhat more satisfied with the INTSUMM system that uses M_{Summ} configuration *v* than configuration *i*. Interestingly, it seems the users may have appreciated the slightly better informativeness of configuration *v* even if the query-responsiveness was not as good as in configuration *i*, as shown through the QSIM score. In addition, we see that absolute manual scores in Table 3 are lower than in Table 2, but trends are generally similar. It is common that scaling of manually supplied scores can fluctuate (e.g. Gillick and Liu, 2010).

Figures 3 and 4 show the averaged (per topic and then over all topics) recall curves of the collected sessions in the experiment described in §5.3 and above, respectively. The x-axis is the accumulating token-length of the session, and the y-axis is the ROUGE-1 recall. The points on the curve are the average interpolated values from all the sessions. The vertical dashed lines are the intersecting bounds of the sessions, from 106 tokens to 250. The area under the curve (AUC) is computed for each of the curves, and reported in the first row of Tables 2 and 3. The higher AUC scores obtained from the recall curves of our models, compared to those of the S_2 baseline, highlight the ability to expose more salient information earlier in the session.

PREPRINT

E.2 Execution Time of Systems

Systems that are made for interacting with humans must respond quickly in order to keep the user's engagement. The exact amount of time does not affect the user experience as long as it does not surpass some limit, after which the user starts losing interest or feeling irritated (Attig et al., 2017; Anderson, 2020).

As mentioned in Appendix D, M_{Summ} generates summaries in under a second and M_{Sugg} prepares the list in a few seconds. The baseline summarizer also responds in under a second. The difference between the systems is virtually unperceivable during interaction. There were no comments from the users in our experiments that stated any issue with execution time.

E.3 Assessment of Suggested Queries Functionality

In this analysis, we assessed what modes of query submission users relied on over the course of a session. To that end, (1) we divided each session to three segments (first, second and third part of the session), and counted the types of queries. The types are "suggested query", "free-text", "highlight" (a span from the summary text) and "repeat" (repeating the last submitted query). (2) We then computed the percentage of each mode in each segment. (3) The percentages over all sessions and all topics were computed for each of the three segments.

This process was conducted only for sessions between 4 and 20 interactions, as the few long and short sessions often show different behavior. For the first experiment, this left 43 sessions with avg. 8.63 (std. 2.32) interactions for our system, and 50 sessions with 8.44 (2.48) interaction for S_2. For the second experiment, it left 72 sessions with 10.24 (4.82) interactions for our system, and 74 sessions with 9.59 (4.42) interactions for S_2.

We focus here on the use of suggested queries versus all other query types. In the first experiment we observe a change of +9% from the first to the third segment in our system, and -20% in S_2. In the second experiment we see +18% and -28% in S_2. As discussed in §5.3, this suggests the effectiveness of updated suggested queries, especially by the end of a session.

F Further Explanations on Evaluation Metrics

The **normalized AUC score for the validation metric** (explained in §3.3) is computed over the recall curve produced from the accumulating summary expansions. Each point on the curve marks an accumulating token-length (x-axis) and an accumulating recall score (y-axis) of an interactive state, as depicted in Figures 3 and 4 (although these figures show the *averaged* session recall curves with bounds, whereas during validation the curve is for a single session and there are no bounds set). By computing the area under the full curve, and dividing by the full length, the normalized AUC score is obtained. The normalization gives an approximate absolute value that can be compared at different lengths (although at large length differences this is not comparable due to the decaying slope of the curve).

The **manual query-responsiveness score**, reported in Tables 2 and 3, is obtained by asking users, at the end of a session, "During the interactive stage, how well did the responses respond to your queries?", for which they rate on a 1-to-5 scale. The scores

are averaged over the topic and then over all topics. This follows the evaluation defined in Shapira et al. (2021b).

The **UMUX-Lite score** (Lewis et al., 2013), reported in Tables 2 and 3, is obtained by asking users to rate (1-to-5) two statements at the end of a session: (1) "The system's capabilities meet the need to efficiently collect useful information for a journalistic overview" and (2) "The system is easy to use". The first question refers to the users' informational goal that they received, in order to follow a consistent objective goal during their exploration. The final score is a function of these two scores, and is used as a replacement for the popular SUS metric (Brooke, 1996) (with a much longer questionnaire), to which it shows very high correlation, thus offering a cheaper alternative. This also follows the evaluation defined in Shapira et al. (2021b).

All **confidence intervals** in Tables 1, 2 and 3 are computed as margins-of-error, on the topic-level, over the standard error of the mean with 95% confidence.[10]

The **token-length values** in Table 1 are averages with standard deviations.

G A2C Policy Learning

A policy gradient-based reinforcement learning approach (Williams, 1992) allows optimizing on non-differentiable metrics, and eliminates the exposure bias that occurs with traditional training methods, like cross-entropy, on generation tasks (Ranzato et al., 2016).

Specifically, we use the Advantage Actor Critic (A2C) policy gradient training method. See technical explanations in the appendix of (Chen and Bansal, 2018). At a high level, an output reward (subtracted by a baseline reward – computed on a version of the model without MMR attention) is used to weight the output selection in the loss function. In so, outputs with higher rewards increase the likelihood of those outputs and lower rewards decrease the likelihood. Since the reward function is not differentiable, it is used as a weight on the probability of the selected output, which is then given to the loss function.

H INTSUMM Example

We show in Figure 5 an example of an INTSUMM system using the web application of Shapira et al. (2021b) and our our M_{Summ} (configuration *i* from Table 1) and M_{Sugg} models in the backend.

[10] E.g., see `https://www.calculator.net/standard-deviation-calculator.html`

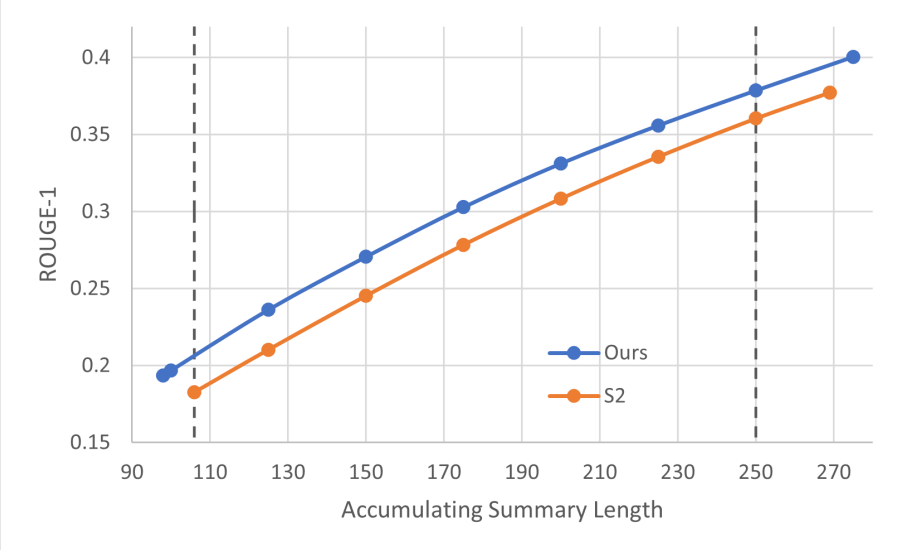

Figure 3: Averaged recall curves of our system and the S_2 baseline system in the experiment described in §5.3 and Table 2 (using M_{Summ} configuration *v* from Table 1). The intersecting range is bounded by dashed lines (between 106 and 250 tokens).

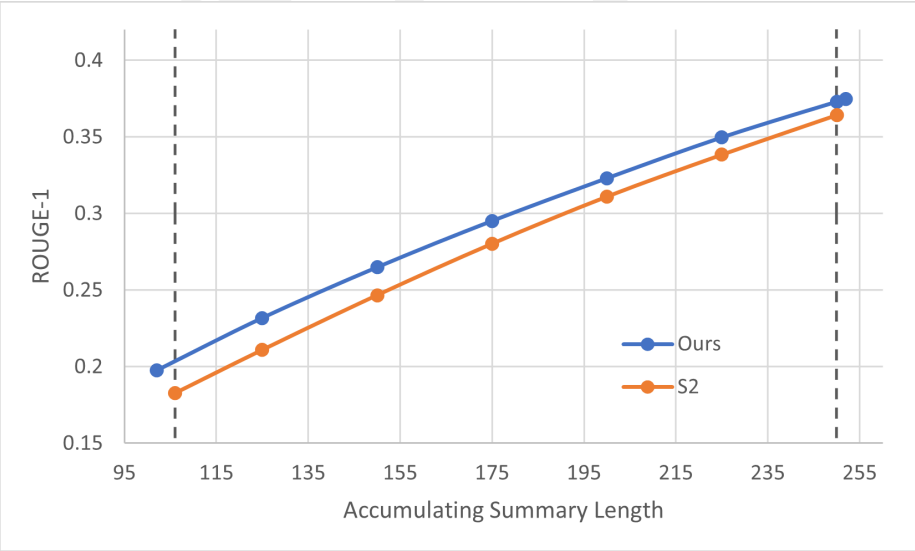

Figure 4: Averaged recall curves of our system and the S_2 baseline system in the experiment described here in Appendix E.1 and Table 3 (using M_{Summ} configuration *i* from Table 1). The intersecting range is bounded by dashed lines (between 106 and 250 tokens).

Figure 5: An INTSUMM system using the web application of Shapira et al. (2021b), with our M_{Summ} and M_{Sugg} models run in the backend, on one of the topics in DUC 2006 with 25 news documents about "Global Warming". Sub-figure (a) shows the initial summary and the initial list of suggested queries. Sub-figure (b) shows the result of clicking the "carbon dioxide gas" suggested query (with the query response and updated suggested queries list). Sub-figure (c) shows the result of subsequently submitting the query "water level". Query responses should be informative for the general topic, while also complying to the user queries. System summaries and expansions must be output fast in order to allow smooth interaction and human engagement.