Conditional Generation with a Question-Answering Blueprint

Shashi Narayan¹, Joshua Maynez¹, Reinald Kim Amplayo¹, Kuzman Ganchev¹, Annie Louis², Fantine Huot¹, Anders Sandholm², Dipanjan Das¹, Mirella Lapata¹

¹Google DeepMind, UK  ²Google Research
shashinarayan@google.com, joshuahm@google.com, reinald@google.com, kuzman@google.com, annielouis@google.com, fantinehuot@google.com, sandholm@google.com, dipanjand@google.com, lapata@google.com

Abstract
The ability to convey relevant and faithful information is critical for many tasks in conditional generation and yet remains elusive for neural seq-to-seq models whose outputs often reveal hallucinations and fail to correctly cover important details. In this work, we advocate planning as a useful intermediate representation for rendering conditional generation less opaque and more grounded. We propose a new conceptualization of text plans as a sequence of question-answer (QA) pairs and enhance existing datasets (e.g., for summarization) with a QA blueprint operating as a proxy for content selection (i.e., what to say) and planning (i.e., in what order). We obtain blueprints automatically by exploiting state-of-the-art question generation technology and convert input-output pairs into input-blueprint-output tuples. We develop Transformer-based models, each varying in how they incorporate the blueprint in the generated output (e.g., as a global plan or iteratively). Evaluation across metrics and datasets demonstrates that blueprint models are more factual than alternatives which do not resort to planning and allow tighter control of the generation output.

1 Introduction
Neural generation models are often prone to hallucination (Song et al., 2018; Maynez et al., 2020; Kryscinski et al., 2020; Gabriel et al., 2021), repetition and redundancy (Li et al., 2018; Suzuki and Nagata, 2017), and struggle to identify which content units are salient (Tan et al., 2017a). These phenomena are amplified when generating long-form text, i.e., documents with multiple paragraphs (Wiseman et al., 2017), when dealing with non-linguistic data (e.g., database tables), or very long input—which is common when summarizing multiple documents (Liu and Lapata, 2019; Perez-Beltrachini et al., 2019), books (Kryściński et al., 2021), or dialogue (Chen et al., 2022; Zhong et al., 2021). An additional challenge concerns the blackbox nature of deep learning systems, which hides the inherent complexity of modeling multiple interconnected linguistic phenomena in text generation, and makes it difficult to examine model decisions and attribute errors to specific components. The lack of modularity further affects controllability as these systems cannot be easily tailored to individual needs.

Attempts to remedy some of these issues focus on changing the way entities are represented (Puduppully et al., 2019b; Iso et al., 2019), allowing the decoder to skip low-confidence tokens to enhance faithful generation (Tian et al., 2019), modeling graph connections between document elements to better capture salience (Tan et al., 2017b; Liu and Lapata, 2019), encoding documents hierarchically (Celikyilmaz et al., 2018; Liu and Lapata, 2019; Rohde et al., 2021), learning latent alignments between the input and the target text (Xu et al., 2021), adopting sparse attention mechanisms (Child et al., 2019; Beltagy et al., 2020), and introducing content selection (Gehrmann et al., 2018; Dou et al., 2021) and planning components (Puduppully et al., 2019a; Moryossef et al., 2019b; Narayan et al., 2021; Wiseman et al., 2018).

In this paper we also aim to render conditional generation more modular via an intermediate, plan-based representation. While autoregressive models of language predict one token at a time, there is evidence that in humans some degree of planning occurs at a higher level than individual words (Levelt, 1993; Guhe, 2007). A long tradition in natural language generation views
planning as a central component to identifying important content and structuring it appropriately (Reiter and Dale, 2000), however, there is less agreement on how plans should be represented. Common examples include discourse trees (Mellish et al., 1998), entity transitions (Kibble and Power, 2004; Barzilay and Lapata, 2008), sequences of propositions (Karamanis, 2004), and schemas (McKeown, 1985).

Our work proposes a new conceptualization of text plans as a sequence of question-answer pairs. Specifically, we draw inspiration from the “Questions under Discussion” (QUD) theory of discourse structure, which posits that one way of articulating the structure of a text is to identify the questions and sub-questions that are raised and answered by subsequent spans of text (Carlson, 1983; Ginzburg, 1994; Van Kuppevelt, 1995; Larson, 2002; Roberts, 2012; Riester, 2019). Theoretical models of QUD assume that discourse contains implicit questions for each of the assertions made, which are thereby turned into answers. These questions and answers can be understood in terms of their use in moving a discourse forward to achieve communicative goals. We propose to make QUDs explicit by exploiting state-of-the-art question generation technology (Alberti et al., 2019; Lu and Lu, 2021) and use them as an intermediate representation layer for conditional generation, i.e., a question-answering (QA) blueprint operating as a proxy for both content selection (i.e., what to say) and planning (i.e., in what order).

Table 1 illustrates a plan for generating a Wikipedia abstract from the AQuaMuSe dataset (Kulkarni et al., 2020). We enhance existing datasets (e.g., for summarization) with similar blueprints which we obtain automatically. We then convert input-output pairs into input-blueprint-output tuples and propose to learn encoder-decoder models from these augmented annotations. We develop three models that vary in how they integrate blueprints in the generation process and their ability to handle long outputs. Aside from generating blueprints and their corresponding text in one go, we propose a new architecture that iteratively plans and generates a sentence at a time, conditioning on the input and the output sentences generated so far. We do not generate a global blueprint, rather, our planning process is incremental and informed by generation, which we argue affords greater control over the output and its fluency. Moreover, the model is better equipped for long-form generation, since it does not have to (autoregressively) decode the blueprint and its summary in one go, avoiding the risk of exceeding the maximum decoder length.

We instantiate our models with a Transformer (Vaswani et al., 2017) encoder-decoder architecture and perform experiments on summarization datasets representing different information seeking tasks, application domains, and user requirements.\(^1\) In all cases, we empirically demonstrate that blueprint models are more factual than alternatives which do not resort to planning; we also observe that QA blueprints are a better representation compared to plans based on entity chains (Narayan et al., 2021), allowing tighter control of the output, and providing a comprehensive explanation for model predictions (if the plan is erroneous, then the summary will be too).

### 2 Related Work

**Questions under Discussion** The QUD-based approach to discourse structure assumes an open-ended inventory of possible questions and sub-questions (Van Kuppevelt, 1995). Recent efforts (De Kuthy et al., 2018; Westera et al., 2020;\(^1\)Our models, training data and predictions are available at [https://github.com/google-research/google-research/tree/master/text_blueprint].
 Riester, 2019) have nevertheless shown that it is possible to manually annotate documents with QUDs, i.e., to formulate a question for every assertion expressed in a text. De Kuthy et al. (2020) even go as far as to partially automate QUD annotation in German by automatically generating all potentially relevant questions for a given sentence. Related work (Ko et al., 2020) focuses on the generation of inquisitive questions that reflect general text understanding and free-form open-ended questions (Ko et al., 2021). Our work builds upon QUD and related discourse structure theories, although, we do not directly implement any of them in particular. We adopt question answering as a good way of spelling out the connection between the information structure of a sentence and the discourse in which the sentence can function.

**QA Pairs as a Proxy for Annotation Labels**

Question-answer pairs have been previously used as a proxy for expressing semantic content. QA-SRL (He et al., 2015) is a representation based on QA pairs that has been shown to capture the vast majority of arguments and modifiers in PropBank (Palmer et al., 2005) and NomBank (Meyers et al., 2004). Instead of using a pre-defined role lexicon, QA-SRL labels semantic roles with questions whose answers denote the argument bearing the role. Follow-on work uses QA pairs to represent discourse relations (Pyatkin et al., 2020) and to capture overlap or redundancy at the propositional level (Brook Weiss et al., 2021). We also employ QA pairs as an abstraction of propositional content, however, we do not target specific relation types, or make any linguistic assumptions about them (e.g., discourse relations vs semantic roles).

**Question-Answering in Summarization**

QA pairs have been used for evaluating summaries (Deutsch and Roth, 2021b; Eyal et al., 2019; Durmus et al., 2020; Wang et al., 2020), specifically as a means of estimating the information overlap between a reference summary and a system-generated one. QA-based signals have also been incorporated in the training of summarization models, using reinforcement learning (Arunmae and Liu, 2018, 2019; Scialom et al., 2019) or as a way of identifying salient content in the input document (Deutsch and Roth, 2021a). Cao and Wang (2022) introduce the task of hierarchical question-summary generation, where a source document is condensed into multiple summaries, each answering a different question. Questions are organized hierarchically into broad questions and more specific sub-questions that are learned from manual annotations. Our model outputs a QA-based plan and a single summary for a given document, although it is possible to generate different summaries from different plans for the same document. Our QA pairs are obtained automatically and they are not structured.

**Planning in Encoder-Decoder Models**

Various recent efforts have developed planning modules in the context of data-to-text generation. In most cases, the plans are specific to the input, which varies from tables and records to RDF tuples. For instance, Puduppully et al. (2019a) learn a plan corresponding to a sequence of records, and generate a summary conditioned on it. Narayan et al. (2020) treat content selection as a task similar to extractive summarization; they first extract sentence plans and then verbalize them one-by-one. Morossef et al. (2019a,b) propose a symbolic planning stage followed by a neural realization stage. Other work (Puduppully and Lapata, 2021; Puduppully et al., 2022) advocates macro planning, where document content is organized into a sequence of paragraph plans which are verbalizations of tabular input. Our work is closest to Narayan et al. (2021), who also target summarization applications and learn an intermediate plan to guide generation. We adopt a more elaborate plan representation based on QA blueprints, and interface decoding with plan generation similarly to Narayan et al. (2020).

**3 Text Generation with Blueprints**

**3.1 Problem Formulation**

Let \( \text{d} \) denote the input to the model which could be a document (or multiple documents), a dialogue history, or even database tables. The model will learn to generate blueprint \( \text{b} \) for output \( \text{s} \) (e.g., a summary) and the output itself. The blueprint \( \text{b} \) is an ordered set of question-answer pairs \( \{(q_1, a_1), (q_2, a_2), \ldots, (q_m, a_m)\} \). Unsurprisingly, such blueprints are not naturally occurring in existing datasets that typically consist of \( (\text{d}, \text{s}) \) pairs. In the following we explain how we automatically augment training examples \( (\text{d}, \text{s}) \) into tuples \( (\text{d}, \text{b}, \text{s}) \) with blueprints (Section 3.2) and then
describe how we devise blueprint models based on them (Section 3.3).

### 3.2 Blueprint Annotation

We first explain how question-answer pairs are automatically (over-)generated for output s, and subsequently filtered to create blueprint b. We illustrate the different filtering stages via the example in Table 2.

**Question-Answer Generation** We generate QA pairs following an approach similar to Honovich et al. (2021, 2022). We convert the SQuAD reading comprehension dataset (Rajpurkar et al., 2018b) to a question generation dataset by concatenating the answer and context (with separators) and fine-tuning a sequence-to-sequence transformer model to predict the question. Specifically, we fine-tune the TS-11B checkpoint from Raffel et al. (2020); questions are decoded with a beam size of 4. During training, answer candidates are the answers provided in the SQuAD annotation. At inference time, answer candidates are the answers provided in the SQuAD trained system. This procedure yields a large list of QA pairs (see in Table 2 the questions generated for the summary at the bottom), which we reduce using the filtering explained below.

**Question-Answer Blueprints** Initially, we apply a Round-trip Consistency check (Alberti et al., 2019), which discards questions if they yield answers different from those used to generate them. In Table 2, Q11 is discarded as the answer it is paired with is wrong (1968 was the final year that Shelby American built the Mustang, not 1970). The same is the case for Q13, where the answer to the question ought to have been the introduction of the fifth generation Ford Mustang.

To decrease the number of QA pairs further, we chunk the text (bottom block in Table 2) into propositions—a proposition is a sub-sentential unit which represents a single claim or fact (Stanovsky et al., 2018; Ernst et al., 2022). We use propositions instead of sentences since the latter can be too long and contain multiple facts. We split text into propositions based on punctuation (period, comma, and semicolon), coordination (e.g., and, but), relative pronouns (e.g., that, who), and prepositions (e.g., at, by). Following this simple approach, the summary in Table 2 is split into six propositions, shown within square brackets. We next match each proposition to a
single QA pair heuristically, following a two-stage approach.

We first find the question whose answer is at the rightmost position within a proposition. If there are multiple such questions, we select the one with the longest answer. This first stage, which we call Rheme, is motivated by the theme-rheme structure (Vallduví and Vilkuna, 1998) of natural language sentences: Already known information (i.e., the theme) is usually placed first while new information (i.e., the rheme) is placed later in a sentence or phrase (Kruijff-Korbayová and Steedman, 2003). Following this idea, Rheme selection prioritizes new-information seeking questions. As can be seen in Table 2, it eliminates several questions (e.g., Q1–Q4) as their answers are not the right most element in the obtained propositions. Questions Q5 and Q6 are identical, however we retain Q5 as it yields the longest answer.

The second stage, which we call Coverage, prioritizes the selection of informative QA pairs by selecting non-overlapping ones. Specifically, we first convert s to a bag of tokens and select the QA pair with the highest lexical overlap. We then remove the overlapping tokens from s, and repeat this greedy selection process until the bag is empty or the overlap is zero. Table 2 shows how Coverage further eliminates QA pairs Q5 and Q8. The remaining four QA pairs constitute the final blueprint b. Rather than defaulting to a random order, we sort these based on the location of the answer spans in s (see the final order in Table 1).

3.3 Blueprint Models

We devised three seq-to-seq models, which differ in the way the output and its blueprint are generated.

End-to-End Model A straightforward approach would be to take d as input and learn to first predict blueprint b as \( p(b|d) \), and then generate output s as \( p(s|b) \). However, this approach crucially relies on the blueprint being accurate and capturing all required information, which might be overly optimistic, given that blueprints (for training) are generated automatically. Moreover, pipeline architectures are known to suffer from error propagation, which in our case would undoubtedly affect generation performance, the final stage of the pipeline.

Rather than modeling the blueprint and output generation stages separately, we train an encoder-decoder model to encode d and generate b/s (i.e., the concatenation of the blueprint and output sequence) in one go. Essentially, the decoder first predicts blueprint b and then continues to generate output s, using both b and d. We prefix b and s with special markers “Plan:” and “Summary:”, respectively. In particular, we predict b as \( a_1; q_1; \ldots; a_m; q_m \), namely, a (concatenated) sequence of answer-question pairs. The model is trained with the standard maximum-likelihood objective to generate the augmented target b/s. Interestingly, in this end-to-end model the blueprint functions as a macro-plan, i.e., a global sketch of the content and organization of the output.

Multi-task Model It is generally challenging for encoder-decoder models to generate long output sequences (Ko and Li, 2020; Tan et al., 2021). The end-to-end model sketched above further amplifies this problem because it ultimately aims to generate sequence b/s rather than just s, increasing the sequence length by 220% (see Table 3).

To mitigate this problem, we propose a multi-task model optimized to perform two separate tasks. Let a and q denote an ordered sequence of answers \( (a_1, \ldots, a_m) \) and corresponding questions \( (q_1, \ldots, q_m) \), in blueprint b. The model is trained to generate (a) the answer plan concatenated with output sequence a/s, and (b) the answer plan concatenated with questions a/q. In particular, we train a single encoder-decoder model to encode input d, while the decoder first predicts answer plan a (as \( p(a|d) \)) and then continues to generate output s (as \( p(s|a, d) \)) or corresponding questions q (as \( p(q|a, d) \)), depending on the task. We prefix a, q, and s with special markers “Plan:”, “Questions:”, and “Summary:”, respectively. We further prefix input d with “Generate Summary:” or “Generate Questions:” to instruct our model to generate output s or questions q, respectively. We sample data points from these two tasks with equal probability and train the model with the standard maximum-likelihood objective.

During inference, we use a two-step process to generate output s’ and its blueprint b’ for input d.\

\footnote{Predicting b as \( q_1; a_1; \ldots; q_m; a_m \) is more natural, but, it lead to inferior performance. See the ablation experiments in Section 5.3.}
We first prefix $d$, answer spans sequence of (Narayan et al., 2021) with the plan being a se-
it can be viewed as an extension of FROST answers only, not question-answer pairs. As such, quality, since the model now conditions on the
by learning to generate model alleviates the length issue discussed above
s
a
predicted answer plan
s
Generate Questions:
q
ing questions
plan (i.e., answer plan $a$)
Iterative Model

Rather than predicting a global plan (i.e., answer plan $a$ or blueprint $b$) prior
to generating output $s$, we employ an incremental approach that interleaves planning with text generation. Let output $s$ consist of $n$ sentences $\{s_1, s_2, \ldots, s_n\}$; then, the corresponding blueprint $b$ can be represented as $\{b_1, b_2, \ldots, b_n\}$, where $b_i : \{(a_{j+1}^i, q_{j+1}^i) \ldots (a_{j+k}^i, q_{j+k}^i)\}$ consists of $k$ question-answer pairs for sentence $s_i$. We train our model to iteratively plan and generate one sentence at a time, conditioning on the input and the output sentences generated so far. In particular, we train an encoder-decoder model where the en-
coder first encodes input $d$, while the decoder takes summary $\{s_1, \ldots, s_i\}$ generated so far as a prompt and generates blueprint $b_{i+1}$ for next sentence $s_{i+1}$, followed by sentence $s_{i+1}$ itself.

The iterative model is trained on quadruples $\{(d, \phi, b_1, s_1), \ldots, (d, s_{i-1}, b_{i+1}, s_{i+1}), (d, s_1, \ldots, s_i)\}$, where $\phi$ is an empty context placeholder used to predict the first blueprint $b_1$ and corresponding first sentence $s_1$. $(n + 1)$ is the blueprint length, and $s_{1,i} = \{s_1, \ldots, s_i\}$ are the output sentences generated so far; $b_{end}$ and $s_{end}$ are special tokens marking the end of the output prediction. We prefix $s_{1,i}$, $b_i$, and $s_i$ with special markers “Context:”, “Plan:”, and “Next Sentence:”, respectively. We train the model with the standard maximum-likelihood objective to predict $s_{1,i}; b_i; s_i$, however, we do not compute the loss for predicting context $s_{1,i}$ to avoid over-optimizing for sentences that appear at the beginning of the output.

The iterative approach does not create a global macro plan. Rather, it learns micro content plans and verbalizes them one-by-one, conditioning on previously generated sentences but not on previously generated QA pairs. Although it does not have a global document view like the end-to-end model, the iterative decoder cannot exceed the output sequence length as it plans and predicts one sentence at a time as $b_i; s_i$, instead of generating $b; s$ in one go. And unlike the multi-task model, each sentence $s_i$ is generated by conditioning on the full blueprint $b_i$ (consisting of questions and answers).

4 Experimental Setup

4.1 Datasets

We evaluated our model on benchmarks representative of long-form question answering and summarization. Our datasets vary in terms of the

| Source | AQuM | WCSum | SS-FD |
|--------|------|-------|-------|
| # queries | 8,162 | — | — |
| # examples | 6,599 | 165,000 | 3,673 |
| train | 714 | 8,723 | 338 |
| dev | 849 | 9,166 | 337 |
| test | — | — | — |
| # docs | 6.46 | 135.56 | 1.00 |
| # words | 12,986.88 | 7455.75 | 8051.74 |
| # sentences | 339.62 | 307.80 | 804.01 |
| # words/doc | 2,008.38 | 52.02 | 8051.74 |
| # sentences | 114.07 | 115.61 | 126.73 |
| # words | 3.65 | 4.74 | 5.26 |
| novel unigrams | 0.02 | 0.13 | 0.17 |
| novel bigrams | 0.13 | 0.54 | 0.66 |
| novel trigrams | 0.24 | 0.78 | 0.92 |
| novel 4-grams | 0.31 | 0.86 | 0.98 |
| target (original) | 8.16 | 9.56 | 28.10 |
| target (+blueprint) | 272.68 | 291.28 | 597.90 |

Table 3: Summary statistics for the datasets used in this work (AQuM, WCSum, and SS-FD are shorthands for AQuaMuse, WikiCatSum, and ScreenSumm-FD, respectively). We report on the number of queries, size of training, development, and test set, and average source and target length (in terms of documents, words, sentences, and words per document). We quantify the abstract-

We first prefix $d$ with “Generate Summary:” and generate $a'$; $s'$, i.e., answer plan $a'$ followed by output sequence $s'$. We then prefix $d$ with “Generate Questions:”, prompt our decoder with the predicted answer plan $a'$ and generate corresponding questions $q'$ for blueprint $b'$. The multi-task model alleviates the length issue discussed above by learning to generate $a; s$ instead of $b; s$. However, this comes at the expense of generation quality, since the model now conditions on the answers only, not question-answer pairs. As such, it can be viewed as an extension of FROST (Narayan et al., 2021) with the plan being a sequence of answer spans rather than entity chains. This model also creates a macro-plan of the output, however, less detailed compared to the end-
to-end model.

**Iterative Model** Rather than predicting a global plan (i.e., answer plan $a$ or blueprint $b$) prior
input given to the generation model (e.g., multiple documents or one, web pages, or dialogue transcripts), the user’s information need (e.g., answering a question or aggregating information), and summary style (e.g., genuinely abstractive vs extractive). Common features among them are very long inputs and multi-sentence output summaries. We summarize various dataset statistics in Table 3.

**AQuaMuSe** (Kulkarni et al., 2020, 2021) is a query-focused multi-document summarization dataset; it was created with the intent of simulating how a search engine might synthesize documents of high relevance to a user query. It consists of Google Natural Questions (Kwiatkowski et al., 2019) paired with web documents extracted from Common Crawl and long-form answers from Wikipedia. We approach this task as a generative QA problem where we take the query and associated web documents and generate a long-form answer to the query. We work on the split from Kulkarni et al. (2021); on average, each instance has 6.46 web documents (2,008 tokens per document), leading to very long input (12,987 tokens).

**WikiCatSum** (Perez-Beltrachini et al., 2019) is a topic-focused multi-document summarization dataset where the goal is to generate Wikipedia abstracts (i.e., lead article sections) from a large set of webpages related to an entity or a topic. It focuses on three entities, namely, Films (59,973 instances), Companies (62,545 instances), and Animals (60,816 instances). In experiments, we collate the different data subsets into one, which we refer to collectively as WikiCatSum. The input webpages are truncated to the first 800 tokens.

**SummScreen-FD** (Chen et al., 2022) is a recently released dialogue summarization dataset. It contains transcripts of TV episodes (e.g., Game of Thrones, CSI Las Vegas) and corresponding (community authored) summaries. The original dataset is divided into two complementary subsets; we use the ForeverDreaming (FD) subset released as part of the SCROLLS benchmark (Shaham et al., 2022), which incorporates episodes from 88 different shows. SummScreen-FD is a challenging testbed for several reasons. Plot details are often expressed indirectly in conversations between characters and are scattered across the entire transcript. The summarization task is highly compressive, a transcript the size of a book (on average 8,000 tokens; see Table 3) is condensed into a few sentences, and the evaluation of such summaries comes with its own challenges (e.g., it is not realistic to expect humans to read the transcript to be able to assess their quality).

We further analyze the characteristics of these datasets in Table 3. Long-form answers in AQuaMuSe are mostly extractive with only 2%, 13%, 24%, and 31% novel unigrams, bigrams, trigrams, and 4-grams, respectively. In comparison, summaries in WikiCatSum and SummScreen-FD are more abstractive; WikiCatSum abstracts have 13% novel unigrams, 54% bigrams, 78% trigrams, and 86% 4-grams, whereas in SummScreen-FD summaries 17% unigrams, 66% bigrams, 92% trigrams, and 98% 4-grams were not seen in the training. Interestingly, SummScreen-FD summaries have far more propositions than AQuaMuSe or WikiCatSum targets, leading to a much higher number for QA pairs in their blueprints (28.10 vs 8.16 or 9.56). This in turn makes the generation task for end-to-end models very challenging. The average summary length together with the blueprint annotations (i.e., \(b; s\)) for SummScreen-FD is almost twice the size of WikiCatSum and AQuaMuSe (597.90 vs 291.28 and 272.68). The majority of questions in AQuaMuSe and WikiCatSum are what questions (76.0% and 74.2%, respectively), followed by who, where, when, and how questions. For SummScreen-FD, what and who questions are most popular (50.1% and 42.9%, respectively).

### 4.2 Comparison Systems

All our experiments used *LONGT5* (Guo et al., 2021), an extension of the original T5 encoder (Raffel et al., 2020) with global-local attention sparsity patterns to handle long inputs. We compared a vanilla *LONGT5* model (xl, 3B parameters) fine-tuned on our datasets (with a maximum input sequence length of 4,096 tokens and a maximum output length of 512 tokens) against several blueprint variants. These include an end-to-end *LONGT5* model (E2E) which first decodes blueprint \(b\) and then continues to decode output \(s\); a *LONGT5* multitask model (MULTITASK) which jointly learns to predict the answer plan.

---

4We used the publicly released checkpoints from [https://github.com/google-research/longt5](https://github.com/google-research/longt5).
followed by either the output $s$ or the questions in $b$; and a **LONGT5 iterative model (ITERATIVE)** which plans and generates one sentence at a time.

In addition, we implemented a two-stage model (2-STAGE), which first creates blueprint $b$ given input $d$ and then generates output $s$ given $b$ and $d$ as input. Finally, we also fine-tuned T5 (xl, 3B parameters) on our datasets with a maximum input sequence length of 1,024 tokens and a maximum output length of 256 tokens, as a baseline. We present these comparisons in Table 4 together with the performance of various state-of-the-art systems.

We fine-tuned all our models with a leaning rate of 0.001 and a batch size of 128, for 50K steps. We select best checkpoints using average Rouge performance on validation sets. During inference, we use beam search with size 5 and alpha 0.8.

## 5 Automatic Evaluation

In this section we present experimental results using automatic evaluation metrics that assess overall summary (and blueprint) quality. Moreover, we quantify the extent to which automatically generated output is grounded to the blueprint and faithful to the input document/s.

### 5.1 Metrics

**Summary and Blueprint Quality** We evaluate summary quality automatically using (summary-level) Rouge F1 (Lin and Hovy, 2003). We report only RougeLSum$^5$ in Table 4 for the sake of brevity. We also use RougeLSum to evaluate the quality of the automatically generated blueprint, i.e., the QA pairs and their order against the reference blueprint.

**Informativeness and Grounding** We evaluate informativeness using QA-based metrics. Specifically, following the reading comprehension literature (Rajpurkar et al., 2016, 2018b), we quantify the extent to which the generated text can answer all questions from its reference (Informativeness) and predicted blueprint (Grounding).

Following Stelmakh et al. (2022), we use a RoBERTa model (Liu et al., 2019) fine-tuned on SQuAD-V2 for question-answering in both cases.  

---

### Table 4: Results on AQuaMuSe, WikiCatSum, and SummScreen-FD test sets. Baseline and earlier SOTA models are presented in the top block and all blueprint models are shown in the bottom block. Models marked with * generate extractive summaries. HiBERT, TextRank, and SiBERT results on AQuaMuSe are taken from Kulkarni et al. (2021). BART and REFFLECT (extract-then-abstract) results are taken from Song et al. (2022). Hybrid r2T-BART (content selection + generation) results are taken from Chen et al. (2022). Best results for each task are boldfaced. Scores that are not significantly different (using paired bootstrap re-sampling; $p < 0.05$) from the best score in each column are marked with a dagger (†).

Given generated text $s'$ and question-answer pair $(q_i, a_i)$ from the (reference or predicted) blueprint, we apply our question-answering model to $s'$ to predict answer $a_i'$ to question $q_i$. We then compute the token-level F1 score between predicted answer $a_i'$ and ground truth answer $a_i$, and report the average.

**Faithfulness** Hallucinations are a widely known issue with neural abstractive summarization (Song et al., 2018; Maynez et al., 2020; Kryscinski et al.,...
2020; Gabriel et al., 2021), especially when a sentence combines content from multiple sources (Lebanoff et al., 2019).

Following previous work (Maynez et al., 2020; Falke et al., 2019; Narayan et al., 2022; Honovich et al., 2022; Dušek and Kasner, 2020), we quantify the extent to which generated summaries are faithful to their input using textual entailment. We resort to textual entailment for two reasons; firstly, it is a relatively intuitive metric, all information in a summary should be entailed by the source or at least not conflict with it; secondly, recent studies (Maynez et al., 2020; Fischer et al., 2022) have shown that it correlates with human judgments of faithfulness across summarization datasets and tasks.

Following Honovich et al. (2022), we trained an entailment model by fine-tuning T5-11B (Raffel et al., 2020) on the Adversarial NLI dataset (ANLI; Nie et al., 2020). For each sentence (hypothesis) in the summary, we compute its entailment probability given the input (premise) and report the average across all sentences to obtain an overall score (Maynez et al., 2020).

More formally, let $E$ denote a textual entailment model that predicts $E(a, b)$, namely, that text $b$ is entailed by text $a$. The faithfulness score $F$ of summary $s$ containing sentences $s_1, \ldots, s_2$ with respect to input $D$ is computed as:

$$F(s) = \frac{1}{n} \sum_{i=1}^{n} E(D, s_i)$$

where $n$ is the number of sentences in the summary. If the input is longer than the T5 maximum encode length, we split it, calculate the entailment probability per split, and take the maximum.

We further validated our ANLI entailment scores against human judgments of faithfulness elicited as part of SummEval (Fabbri et al., 2021), a recently released dataset for assessing automated summarization metrics. Our entailment predictions correlate well with human ratings, achieving a Spearman’s rank correlation of $\rho = 0.774$.

5.2 Results

Why LONGT5 for Blueprint Models

All the tasks we are dealing with require modeling input of highly complex nature, which is often very long (see Table 3). Our results in Table 4 (see Rouge/summary column) demonstrate that T5 models always fall behind LONGT5, underscoring the importance of sparse attention mechanisms for modeling long inputs. In fact, LONGT5 sets a new state of the art on AQuaMuSe and SummScreen-FD. On WikiCatSum, it is slightly worse than Reflect (Song et al., 2022), an extract-then-abstract model which has a dedicated content selection module. Similar content selection techniques could also benefit LONGT5, however, we leave this to future work. We henceforth use LONGT5 as a base model for fine-tuning our blueprint models.

Blueprint Models and Rouge

Compared to LONGT5, blueprint variants slightly underperform on AQuaMuse, but score better on WikiCatSum and SummScreen-FD (see MULTITASK model). All differences between LONGT5 and blueprint models are statistically significant using paired bootstrap resampling; $p < 0.05$). For a fair comparison, we always use a maximum decoder length of 512 tokens. With the exception of AQuaMuse, E2E is inferior to other blueprint models, which is not surprising since it has to generate much longer text (recall it predicts $b; s$ rather than simply $s$). Overall, MULTITASK is significantly better than other blueprint models on WikiCatSum but on par with Iterative on SummScreen-FD.

Similar patterns emerge when evaluating the predicted blueprints against reference QA pairs, with Iterative significantly outperforming the other two variants on SummScreen-FD. This could be due to the fact that SummScreen-FD summaries have far more propositions than AQuaMuSe or WikiCatSum targets; it is better to predict them one sentence at a time, rather than all together. With regard to WikiCatSum, the difference between MULTITASK and Iterative is not significant (although MULTITASK has a slight numerical advantage) and both systems are significantly better than E2E. On AQuaMuSe, MULTITASK is significantly better than E2E and Iterative.

Note that all 2-STAGE models are significantly worse in comparison to blueprint variants, when evaluating either their blueprints or summaries (in terms of Rouge). While our models learn to optimize blueprints and summaries together, 2-STAGE models are faced with the harder task of predicting the blueprint solely based on the input (text-to-data). Since the blueprints learned by the
first stage are of poor quality, the summaries generated in the second stage are also inferior.

**Blueprint Models and Informativeness** Our blueprint annotation of reference summaries naturally provides a more principled alternative to Rouge. We can now use QA pairs in reference blueprints to evaluate the informativeness of predicted summaries. Results follow a pattern overall similar to Rouge, however, this approach reveals the complexity of the different generation tasks better than Rouge. While we were able to achieve reasonably high Rouge across datasets, we are far from generating informative summaries. On SummScreen-FD, in particular, we achieve a maximum Rouge score of 31.88, but are able to answer correctly only 7.59% of reference questions using the predicted summaries.

Across datasets, LONGT5 performs on par with MULTITASK, the difference between the two models is not statistically significant, and the same is true of ITERATIVE on SummScreen.

**Blueprint Models and Grounding** The E2E and ITERATIVE variants are significantly better than MULTITASK in generating texts grounded to their predicted blueprints (see ground. column in Table 4). This is because both models generate text conditioned on their blueprints; E2E first predicts blueprint b and then continues to generate output s using both b and the input, whereas ITERATIVE plans and generates one sentence at a time as b_i; s_i. This is not the case with MULTITASK, which generates s conditioned on answer spans only. E2E performs slightly better than ITERATIVE on AQuaMuSe and WikiCatSum (differences are not statistically significant) but struggles on SummScreen-FD, where summaries are longer with more facts/propositions, requiring inference over long-range dependencies, and common sense reasoning. ITERATIVE seems the best option for grounded generation without sacrificing informativeness (ITERATIVE is most informative amongst blueprint models on SummScreen-FD, second best on AQuaMuSe, and third best on WikiCatSum).

**ITERATIVE Is Most Faithful Model** As far as faithfulness is concerned, ITERATIVE performs consistently better than E2E and MULTITASK, as well as T5 and LONGT5 models where text is generated from scratch without any planning (pairwise differences between ITERATIVE and comparison systems are all significant with the exception of E2E on AQuaMuse). On SummScreen-FD, ITERATIVE brings large gains on faithfulness without sacrificing informativeness (both in terms of Rouge and QA-F1). The ANLI score for ITERATIVE is 20.84, whereas it is below 10 for E2E and MULTITASK. E2E outperforms LONGT5 on AQuaMuSe and WikiCatSum, but gains are smaller compared to ITERATIVE.

We show examples of system output in Table 5, highlighting propositions that are not grounded to the input in red. Generated questions from blueprint models are not shown due to space constraints.

### Table 5: System output and reference summary for SummScreen-FD (CSI S6.E9, “Dog Eat Dog”).

| Propositions not grounded to the input |
|----------------------------------------|
| E2E sums are shorter, which is somewhat expected; the model has to decode both the plan and the summary and in cases where the blueprint is large (e.g., in SummScreen-FD), |
Table 6: Example of plan/summary generated by our E2E blueprint model as answer to the question “What is the difference between an Old English Bulldog and an English Bulldog?” (AQuaMuse test set); user edits to the plan and updated summary are shown in red.

Table 7: Example of plan/summary generated by the E2E blueprint model as answer to the question “What section of the world or country is Hinduism usually found in?” (AQuaMuse test set); the part of the plan which is removed by the user is highlighted in red; the shorter summary generated from the elided plan is shown in red.
Table 8: Controllability results on the AQuaMuSe, WikiCatSum and SummScreen-FD test sets. Lighter blue color means more control. Best results for each metric are boldfaced. Scores that are not significantly different (using paired bootstrap resampling; \( p < 0.05 \)) from the best score for each column are marked with a dagger (†).

| Models          | Rouge (RLSum) | QA-F1 | ANLI entail. |
|-----------------|---------------|-------|--------------|
|                 | sum.         | bluepr. | inform |          |
| AQuaMuSe        |               |        |            |
| E2E             | 58.96†       | 54.16† | 34.59     | 83.64†   |
| +drop           | 58.74†       | 52.71† | 34.64     | 83.98†   |
| MULTITASK       | 59.24†       | 54.88† | 37.87†    | 83.37†   |
| +drop           | 59.25†       | 53.79† | 37.57     | 84.98†   |
| ITERATIVE       | 56.48        | 52.92† | 36.04     | 84.77†   |
| +drop           | 56.30        | 46.62  | 36.03     | 85.50†   |
| +Q1             | 55.57        | 52.29† | 34.97     | **85.74**|
| WikiCatSum      |               |        |            |
| E2E             | 39.78        | 45.40† | 19.22†    | 60.56     |
| +drop           | 37.35        | 36.00  | 17.74     | 64.38     |
| MULTITASK       | **42.34**†   | 46.95† | **20.55** | 52.09     |
| +drop           | 40.65        | 36.45  | 19.29†    | 56.71     |
| ITERATIVE       | 39.53        | 46.35† | 18.75     | 64.37     |
| +drop           | 38.80        | 41.78  | 18.14     | **66.56** |
| +Q1             | 38.80        | 43.81  | 18.91     | 64.42†    |
| SummScreen-FD   |               |        |            |
| E2E             | 28.22        | 34.49  | 6.13†     | 8.53†     |
| +drop           | 25.50        | 23.22  | 5.28      | 10.00     |
| MULTITASK       | **31.88**†   | 35.23  | 7.06†     | 9.57†     |
| +drop           | 30.40†       | 24.75  | 6.10      | 12.23†    |
| ITERATIVE       | 29.86†       | **39.36**† | 7.55†    | 20.84†    |
| +drop           | 28.10        | 32.25  | 6.43†     | 24.12†    |
| +Q1             | 27.94        | 29.47  | 6.63†     | **24.80** |

Table 9: System output from ITERATIVE and ITERATIVE+Q1 generating WikiCatSum abstract on "Abraham Verghese."

| E2E             | Rouge (RLSum) | QA-F1 | ANLI entail. |
|-----------------|---------------|-------|--------------|
|                 | sum.         | bluepr. | inform |          |
| QA Plan, Rheme, Covg, Sorted | 48.75 | 39.06 | 44.31 |
| AQ Plan, Rheme, Covg, Sorted  | **50.86** | **39.95** | **45.60** |
| —Sorted, Random   | 50.79 | 36.08 | 43.43 |
| —Rheme            | 47.16 | 40.70 | 44.19 |
| —Coverage         | 47.02 | 41.37 | 44.79 |
| —Rheme, —Coverage | 18.05 | **42.54** | 40.90 |

Table 10: E2E model trained on AQuaMuSe with different selection and sorting (validation set).

5.3 Ablation Studies

As described in Section 3.2, we construct blueprint annotations using the Rheme- and Coverage-based selection strategies. Table 10 presents various ablations that provide rationales for these annotation choices. For the sake of brevity, we report experiments with the E2E model trained (for 50,000 steps) on AQuaMuSe. We observe very similar trends on the other two datasets. As can be seen, it is empirically better to form blueprints from answer-question pairs rather than predicting the questions first and then their answers which is more natural (at least to humans). We further assessed whether sorting the QA pairs based on how they appear in the summary matters by defaulting to a random ordering (see —Sorted in the table). Removing either Rheme or Coverage has a small negative impact on the summaries but not their blueprints, while removing them both is detrimental to summary quality, while the absence of Sorting mostly affects the quality of the blueprint. It is not surprising that sorting is most important to generating a blueprint with correctly ordered propositions.
6 Human-based Evaluation

In addition to automatic evaluation, we conducted three human-based studies assessing different dimensions of output quality. Wishing to avoid well-documented issues\(^7\) with automated bots on Amazon Mechanical Turk and crowdworkers running through HITs as quickly as possible without paying attention to the tasks, we used a few trained annotators. They were given task-specific instructions and went through several pilots to iron out disagreements on edge cases.\(^8\)

6.1 Summary Quality

Our first study assessed overall summary quality. Specifically, we asked our annotators to select the best among three system summaries taking into account how much they deviated from the reference in terms of informativeness (are the summaries on topic or emphasize irrelevant details?) and overall fluency. We adapted the definition of fluency provided in Howcroft et al. (2020): Does the text ‘flow well’ or is it a sequence of unconnected parts?

We conducted our annotation study on 100 instances, each randomly sampled from AQuaMuse, WikiCatSum, and SummScreen. We collected ratings from three annotators (after two rounds of pilot studies to improve agreement) for the output of seven systems. Overall, we obtained 100 (instances) x 3 (datasets) x 6 (systems) x 3 (annotators) = 5,400 annotations. Annotator agreement was 97.11%. Our results are presented in Table 11.

We report on percentage of times each system was ranked best.

In general, we observe that LONGT5 and blueprint models based on it are perceived as significantly better than previous state-of-the-art models (i.e., SiBERT and r2T-BART). On AQuaMuse, LONGT5 is rated overall best, followed by E2E and MULTITASK (however, differences between them are not statistically significant). On WikiCatSum, E2E is rated best bus is not significantly different compared to the other models. On SummScreen, our ITERATIVE variant is rated best followed by LONGT5. These results mirror the difficulty of the task (see Table 3), the longer the input/output, the better ITERATIVE performs.

7\text{https://stanforddaily.com/2020/06/21/}.
8\text{We release our instructions and annotation templates together with our data and models.}

| Models | AQuaMuse | WikiCatSum | SummScreen |
|--------|----------|------------|------------|
| SiBERT | 0.12     | —          | —          |
| r2T-BART | —        | —          | —          |
| LONGT5 | 0.49     | 0.35\(^†\) | 0.42\(^†\) |
| E2E    | 0.39\(^†\) | 0.40     | 0.26      |
| MULTITASK | 0.39\(^†\) | 0.37\(^†\) | 0.39\(^†\) |
| ITERATIVE | 0.30     | 0.28\(^†\) | 0.44      |
| +drop  | 0.31     | 0.27\(^†\) | 0.39\(^†\) |

Table 11: Proportion of times each system was ranked best for summary quality (on AQuaMuse, WikiCatSum, and SummScreen test sets). Best results for each task are \textbf{boldfaced}. Systems in each column are marked with \(^†\) when they are not significantly different from the best system; unmarked pairwise differences from the best system are significant \((p < 0.01);\) using Friedman’s ANOVA test (with post-hoc Wilcoxon signed-rank test, Bonferroni corrected for multiple comparisons).

6.2 Blueprint Quality

We further evaluated the predicted plans more directly. Participants were shown QA blueprints and asked to assess whether they tell a coherent story (are they all relevant and ordered comprehensively?) using a 3-point scale (where 3 is best and 1 is worst). They were also asked to evaluate whether the plans have redundant QA pairs; a QA pair is redundant if it does not add new information to the plan. We collected judgments for

| Blueprint Models | AQuaMuse | WikiCatSum | SummScreen |
|------------------|----------|------------|------------|
| E2E              | 2.83     | 8.30       | 2.94       | 4.70       | 2.43       | 49.3\(^†\) |
| MULTITASK        | 2.58     | 16.00\(^†\) | 2.84\(^†\) | 6.30\(^†\) | 2.28\(^†\) | 33.0 |
| ITERATIVE        | 2.51     | 40.00       | 2.64       | 31.00       | 2.09       | 81.7 |
| +drop            | 2.56     | 36.70       | 2.66       | 27.30       | 2.28       | 63.7 |
| Gold             | 2.85\(^†\) | 7.00\(^†\) | 2.67       | 14.30\(^†\) | 2.02       | 95.7 |

Table 12: Blueprint quality human evaluation on AQuaMuse, WikiCatSum, and SummScreen-FD test sets. Mean scores for coherence (Coh; higher is better) and proportion of QA pairs deemed redundant (Red; lower is better). Best results for each task are \textbf{boldfaced}. Systems in each column are marked with \(^†\) when they are not statistically significant from the best system; unmarked pairwise differences from the best system are significant \((p < 0.01);\) using a Friedman’s ANOVA test with post-hoc Wilcoxon signed-rank test, Bonferroni corrected for multiple comparisons).
Table 13: Human evaluation results for blueprint grounded generation on AQuaMuse, WikiCatSum, and SummScreen-FD test sets. Proportion of QA pairs not mentioned in the summary (Absent; lower is better); proportion of QA pairs with information contradictory to the summary (Contra; lower is better), and mean scores for new information present in the summary (NewInfo; lower is better). The best results for each task are boldfaced. Systems in each column are marked with † when they are not statistically significant from the best system; unmarked pairwise differences from the best system are significant ($p < 0.01$; using a Friedman’s ANOVA test with post-hoc Wilcoxon signed-rant test, Bonferroni corrected for multiple comparisons).

| Models       | Absent↓ | Contra↓ | NewInfo↓ | Absent↓ | Contra↓ | NewInfo↓ | Absent↓ | Contra↓ | NewInfo↓ |
|--------------|---------|---------|----------|---------|---------|----------|---------|---------|----------|
| E2E          | 6.30↑   | 4.10↑   | 1.79↑    | 2.00↓   | 0.40↑   | 1.74↓    | 26.60   | 5.30↑   | 1.76↑    |
| MULTITASK    | 13.20↑  | 13.60↑  | 2.06     | 5.80↑   | 3.70↑   | 1.91↑    | 63.00   | 8.70↑   | 2.60     |
| ITERATIVE    | 5.00↑   | 2.50↑   | 1.77     | 2.80↑   | 2.90↑   | 1.90↑    | 12.00   | 2.80↑   | 1.53     |
| +drop        | 4.90↑   | 2.30↑   | 1.86↑    | 3.60↑   | 2.10↑   | 1.74     | 14.20   | 3.60↑   | 1.89     |
| Gold         | 3.60↑   | 2.40↑   | 1.88↑    | 1.60↑   | 1.80    | 2.02     | 4.50    | 3.90↑   | 1.39     |

The results of our grounding experiments are summarized in Table 13. Across datasets, we observe that ITERATIVE summaries are most grounded. ITERATIVE blueprints have the least number of questions that are absent from or contradict their generated texts. ITERATIVE summaries also display the least amount of new information in relation to their blueprints. ITERATIVE+drop is slightly less grounded compared to ITERATIVE, however, this is not entirely surprising since we prompt the ITERATIVE model with externally modified blueprints (see ITERATIVE+drop in Table 13). Note that ITERATIVE+drop summaries are deemed more faithful than ITERATIVE summaries in automatic evaluation.

The entailment scores improve for all three datasets (see Table 4).
7 Conclusion

In this work we proposed a novel plan-based approach to conditional generation. We conceptualized text plans as a sequence of QA pairs operating as a proxy for what to say and in what order. We developed Transformer-based models that generate by conditioning on a global QA blueprint plan (E2E, MULTITASK) or iteratively by planning and generating one sentence at a time (ITERATIVE). Experimental results across three challenging datasets demonstrate that blueprint models are inherently more informative than vanilla sequence-to-sequence approaches without a planning component. Among the three presented here (E2E, MULTITASK, ITERATIVE), we find that ITERATIVE is the best choice for grounded generation and suggests a promising direction for long-form generation.

Blueprint models offer several advantages compared to blackbox generation. Model predictions can be examined, and errors can be traced back to the blueprint, which in turn can reveal whether the output is informative and faithful to its input. The formulation of the blueprint plan as question-answer pairs makes it intuitive and user-friendly. We have discussed how blueprint models might be used in a human-in-the-loop setting, where users interact with and influence model predictions directly, e.g., by editing the blueprint length and content (as different blueprints lead to different outputs). In the future, we would like to use blueprints more directly to advance methods for training language models using reward learning (Sutton and Barto, 2018), e.g., based on whether the output answers the blueprint questions. Rather than eliciting expensive human feedback (Stiennon et al., 2020), blueprints could provide a cheaper automatic alternative. Finally, although we focused primarily on the generation problem in this work, we believe blueprints might also be useful as a general-purpose approach to retrieving and organizing important content, especially when faced with many and very long inputs.

Acknowledgments

We thank the action editor and our reviewers for their valuable feedback. The human rating process was managed by Muqhtar Mohammad, Kiranmai Chennuru, Ashwin Kakarla and their team; without them this work would not have been possible. Thanks for invaluable support from Sheila de Guia and Suneet Dhingra.

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 6168–6173, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1620

Kristjan Arumae and Fei Liu. 2018. Reinforced extractive summarization with question-focused rewards. In Proceedings of ACL 2018, Student Research Workshop, pages 105–111, Melbourne, Australia. Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-3015

Kristjan Arumae and Fei Liu. 2019. Guiding extractive summarization with question-answering rewards. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 2566–2577, Minneapolis, Minnesota. Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1264

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. Computational Linguistics, 34(1):1–34. https://doi.org/10.1162/coli.2008.34.1.1

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. ArXiv, abs/2004.05150.

Daniela Brook Weiss, Paul Roit, Ayal Klein, Ori Ernst, and Ido Dagan. 2021. QA-align: Representing cross-text content overlap by aligning question-answer propositions. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 9879–9894, Online and Punta Cana,
Shuyang Cao and Lu Wang. 2022. HIBRIDS: Attention with hierarchical biases for structure-aware long document summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 786–807, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.58

L. Carlson. 1983. *Dialogue Games: An Approach to Discourse Analysis*. Riedel, Dordrecht. https://doi.org/10.1007/978-94-015-3963-0

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1662–1675, New Orleans, Louisiana. Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1150

Mingda Chen, Zewei Chu, Sam Wiseman, and Kevin Gimpel. 2022. SummScreen: A dataset for abstractive screenplay summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8602–8615, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.589

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *ArXiv*, abs/1904.10509. https://doi.org/10.48550/arXiv.1904.10509

Kordula De Kuthy, Madeeswaran Kannan, Haemanth Santhi PonnuSamy, and Detmar Meurers. 2020. Towards automatically generating questions under discussion to link information and discourse structure. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5786–5798, Barcelona, Spain (Online). International Committee on Computational Linguistics. https://doi.org/10.18653/v1/2020.coling-main.509

Kordula De Kuthy, Nils Reiter, and Arndt Riester. 2018. QUD-based annotation of discourse structure and information structure: Tool and evaluation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Daniel Deutsch and Dan Roth. 2021a. Question-based salient span selection for more controllable text summarization. *ArXiv*, abs/2111.07935. https://doi.org/10.48550/arXiv.2111.07935

Daniel Deutsch and Dan Roth. 2021b. Understanding the extent to which content quality metrics measure the information quality of summaries. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 300–309, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.conll-1.24

Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. 2021. GS Sum: A general framework for guided neural abstractive summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4830–4842, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.384

Esin Durmus, He He, and Mona Diab. 2020. FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 5055–5070, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.454

Ondřej Dušek and Zdeněk Kasner. 2020. Evaluating semantic accuracy of data-to-text generation with natural language inference. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 131–137, Dublin, Ireland. Association for Computational Linguistics.
Ori Ernst, Avi Caciularu, Ori Shapira, Ramakanth Pasunuru, Mohit Bansal, Jacob Goldberger, and Ido Dagan. 2022. Proposition-level clustering for multi-document summarization. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1765–1779, Seattle, United States. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.naacl-main.128

Matan Eyal, Tal Baumel, and Michael Elhadad. 2019. Question answering as an automatic evaluation metric for news article summarization. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 3938–3948, Minneapolis, Minnesota. Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1395

Alexander R. Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. Summ-Eval: Re-evaluating summarization evaluation. Transactions of the Association for Computational Linguistics, 9:391–409. https://doi.org/10.1162/tacl_a_00373

Tobias Falke, Leonardo F. R. Ribeiro, Prasetya Utama, Ido Dagan, and Iryna Gurevych. 2019. Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2214–2220, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1213

Tim Fischer, Steffen Remus, and Chris Biemann. 2022. Measuring faithfulness of abstractive summaries. In Proceedings of the 18th Conference on Natural Language Processing (KONVENS 2022), pages 63–73, Potsdam, Germany.

Saadia Gabriel, Asli Celikyilmaz, Rahul Jha, Yejin Choi, and Jianfeng Gao. 2021. GO FIGURE: A meta evaluation of factuality in summarization. In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 478–487, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.findings-acl.42

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4098–4109. Association for Computational Linguistics.

Jonathan Ginzburg. 1994. An update semantics for dialogue. In Proceedings of the 1st Tilburg International Workshop on Computational Semantics. Tilburg, The Netherlands.

Markus Guhe. 2007. Incremental Conceptualization for Language Production. Mahwah, NJ: Lawrence Erlbaum Associates Publishers.

Mandy Guo, Joshua Ainslie, David C. Uthus, Santiago Ontaño, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2021. LongT5: Efficient text-to-text transformer for long sequences. ArXiv, abs/2112.07916. https://doi.org/10.18653/v1/2022.findings-naacl.55

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 643–653, Lisbon, Portugal. Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1076

Or Honovich, Roee Aharoni, Jonathan Herzig, Hagai Taitelbaum, Doron Kukliansy, Vered Cohen, Thomas Scialom, Idan Szpektor, Avinatan Hassidim, and Yossi Mattias. 2022. TRUE: Re-evaluating factual consistency evaluation. In Proceedings of the Second DialDoc Workshop on Document-grounded Dialogue and Conversational Question Answering, pages 161–175, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.dialdoc-1.19

Or Honovich, Leshem Choshen, Roee Aharoni, Ella Neeman, Idan Szpektor, and Omri Abend. 2021. Q2: Evaluating factual consistency in knowledge-grounded dialogues via question generation and question answering. In Proceedings of the 2021 Conference on Empirical
Methods in Natural Language Processing, pages 7856–7870. https://doi.org/10.18653/v1/2021.emnlp-main.619

David M. Howcroft, Anya Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A. Hasan, Saad Mahamood, Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser. 2020. Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In Proceedings of the 13th International Conference on Natural Language Generation, pages 169–182, Dublin, Ireland. Association for Computational Linguistics.

Hayate Iso, Yui Uehara, Tatsuya Ishigaki, Hiroshi Noji, Eiji Aramaki, Ichiro Kobayashi, Yusuke Miyao, Naoki Okazaki, and Hiroya Takamura. 2019. Learning to select, track, and generate for data-to-text. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2102–2113, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1202

Nikiforos Karamanis. 2004. Entity Coherence for Descriptive Text Structuring. Ph.D. thesis, School of Informatics, University of Edinburgh.

Rodger Kibble and Richard Power. 2004. Optimizing referential coherence in text generation. Computational Linguistics, 30(4):401–416. https://doi.org/10.1162/0891201042544893

Wei-Jen Ko, Te-yuan Chen, Yiyan Huang, Greg Durrett, and Junyi Jessy Li. 2020. Inquisitive question generation for high level text comprehension. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6544–6555, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.530

Wei-Jen Ko, Cutter Dalton, Mark P. Simmons, Eliza Fisher, Greg Durrett, and Junyi Jessy Li. 2021. Discourse comprehension: A question answering framework to represent sentence connections. ArXiv, abs/2111.00701. https://doi.org/10.48550/arXiv.2111.00701

Wei-Jen Ko and Junyi Jessy Li. 2020. Assessing discourse relations in language generation from GPT-2. In Proceedings of the 13th International Conference on Natural Language Generation, pages 52–59, Dublin, Ireland. Association for Computational Linguistics.

Ivana Kruijff-Korbayová and Mark Steedman. 2003. Discourse and information structure. Journal of logic, language and information, 12(3):249–259. https://doi.org/10.1023/A:1024160025821

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 9332–9346, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.750

Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2021. Booksum: A collection of datasets for long-form narrative summarization. ArXiv, abs/2105.08209. https://doi.org/10.48550/arXiv.2105.08209

Sayali Kulkarni, Sheide Chammas, Wan Zhu, Fei Sha, and Eugene Ie. 2020. Aquamuse: Automatically generating datasets for query-based multi-document summarization. ArXiv, abs/2010.12694. https://doi.org/10.48550/arXiv.2010.12694

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. ArXiv, abs/2010.12694 https://doi.org/10.1007/978-3-030-86331-9_6

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. Transactions of the Association for Computational Linguistics, 7:452–466. https://doi.org/10.1162/tacl_a_00276
Staffan Larson. 2002. *Issue-based Dialogue Management*. Ph.D. thesis, Göteborg University, Sweden.

Logan Lebanoff, John Muchovej, Franck Dernoncourt, Doo Soon Kim, Seokhwan Kim, Walter Chang, and Fei Liu. 2019. Analyzing sentence fusion in abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 104–110, Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-5413

Willem J. M. Levelt. 1993. *Speaking: From Intention to Articulation*. The MIT Press. https://doi.org/10.7551/mitpress/6393.001.0001

Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. 2018. Improving neural abstractive document summarization with explicit information selection modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1205

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 150–157.

Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1500

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *ArXiv*, abs/1907.11692. https://doi.org/10.48550/arXiv.1907.11692

Chao-Yi Lu and Sin-En Lu. 2021. A survey of approaches to automatic question generation: From 2019 to early 2021. In *Proceedings of the 33rd Conference on Computational Linguistics and Speech Processing (ROCLING 2021)*, pages 151–162, Taoyuan, Taiwan. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP).

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.173

Kathleen McKeown. 1985. *Text Generation: Using Discourse Strategies and Focus Constraints to generate Natural Language Text*. Studies in Natural language Processing. Cambridge University Press.

Chris Mellish, Alistair Knott, Jon Oberlander, and Mick O’Donnell. 1998. Experiments using stochastic search for text planning. In *Natural Language Generation*, Niagara-on-the-Lake, Ontario, Canada. Association for Computational Linguistics.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank project: An interim report. In *Proceedings of the Workshop Frontiers in Corpus Annotation at HLT-NAACL 2004*, pages 24–31, Boston, Massachusetts, USA. Association for Computational Linguistics.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019a. Improving quality and efficiency in plan-based neural data-to-text generation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 377–382, Tokyo, Japan. Association for Computational Linguistics. https://doi.org/10.18653/v1/W19-8645

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019b. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2267–2277, Minneapolis, Minnesota. Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1236
Bowen Tan, Zichao Yang, Maruan Al-Shedivat, Eric Xing, and Zhiting Hu. 2021. Progressive generation of long text with pretrained language models. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4313–4324, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.341

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017a. Abstractive document summarization with a graph-based attentional neural model. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1171–1181, Vancouver, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1108

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017b. Abstractive document summarization with a graph-based attentional neural model. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1171–1181. Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1108

Ran Tian, Shashi Narayan, Thibault Sellam, and Ankur P. Parikh. 2019. Sticking to the facts: Confident decoding for faithful data-to-text generation. ArXiv, abs/1910.08684. https://doi.org/10.48550/arXiv.1910.08684

Enric Vallduví and Maria Vilkuna. 1998. On rheme and kontrast. The Limits of Syntax, pages 79–108. Brill.

Jan Van Kuppevelt. 1995. Discourse structure, topicality and questioning. Journal of Linguistics, 31(1):109–147. https://doi.org/10.1017/S002222670000058X

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 5998–6008. Curran Associates, Inc.

Alex Wang, Kyunghyun Cho, and Mike Lewis. 2020. Asking and answering questions to evaluate the factual consistency of summaries. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5008–5020, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.450

Matthijs Westera, Laia Mayol, and Hannah Rohde. 2020. TED-Q: TED talks and the questions they evoke. In Proceedings of the 12th Language Resources and Evaluation Conference, pages 1118–1127, Marseille, France. European Language Resources Association.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics. https://doi.org/10.18653/v1/D17-1239

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning neural templates for text generation. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3174–3187, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1356

Xinnuo Xu, Ondřej Dušek, Verena Rieser, and Ioannis Konstas. 2021. AggGen: Ordering and aggregating while generating. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1419–1434, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.113
Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir Radev. 2021. QMSum: A new benchmark for query-based multi-domain meeting summarization. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5905–5921, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.472