Planning with Learned Entity Prompts for Abstractive Summarization

Shashi Narayan
Google Research
shashinarayan@google.com

Yao Zhao
Google Brain
yaozhaoyz@google.com

Joshua Maynez
Google Research
joshuahm@google.com

Gonçalo Simões
Google Research
gsimoes@google.com

Vitaly Nikolaev
Google Research
vitalyn@google.com

Ryan McDonald*
ASAPP
ryanmcd@asapp.com

Abstract

We introduce a simple but flexible mechanism to learn an intermediate plan to ground the generation of abstractive summaries. Specifically, we prepend (or *prompt*) target summaries with entity chains—ordered sequences of entities mentioned in the summary. Transformer-based sequence-to-sequence models are then trained to generate the entity chain and then continue generating the summary conditioned on the entity chain and the input. We experimented with both pretraining and finetuning with this content planning objective. When evaluated on CNN/DailyMail, XSum, SAMSum, and BillSum, we demonstrate empirically that the grounded generation with the planning objective improves entity specificity and planning in summaries for all datasets, and achieves state-of-the-art performance on XSum and SAMSum in terms of ROUGE. Moreover, we demonstrate empirically that planning with entity chains provides a mechanism to control hallucinations in abstractive summaries. By prompting the decoder with a modified content plan that drops hallucinated entities, we outperform state-of-the-art approaches for faithfulness when evaluated automatically and by humans.

1 Introduction

Jones (1993) described text summarization—the task of generating accurate and concise summaries from source document(s)—as a three-step process: (i) Building the source representation from the source document(s), (ii) Learning a summary representation from the source representation, and (iii) Synthesizing the output summary text. Common to most traditional methods, an input representation was learned by semantically analyzing the source text, the summary representation was then learned by modifying and refining the input representation, and finally the summary was generated grounded to the intermediate summary representation (Luhn, 1958; McKeown and Radev, 1995; Barzilay and Elhadad, 1997; Mihalcea and Tarau, 2004).

State-of-the-art neural summarizers are powerful representation learners and conditional language models, thanks to sequence-to-sequence architectures (seq2seq) with attention and copy mechanisms (Hochreiter and Schmidhuber, 1997; Bahdanau et al., 2015; See et al., 2017), Transformer architectures with multi-headed self-attention (Vaswani et al., 2017), and large pretrained conditional language models (Dong et al., 2019; Song et al., 2019; Lewis et al., 2020; Rothe et al., 2020; Raffel et al., 2019; Zhang et al., 2020). However, the grounding of summary generation that was inherent to most traditional methods is yet to be achieved in neural summarization. The attention mechanism (Bahdanau et al., 2015), especially in pretrained encoder-decoder models (Lewis et al., 2020; Raffel et al., 2019; Zhang et al., 2020), plays a key role in aligning summary content to the input, yet undesired hallucinations are common in generated summaries (Maynez et al., 2020; Kryscinski et al., 2020; Gabriel et al., 2021).

In this paper, we investigate *Entity Chains*—ordered sequences of entities[1] in the summary—as an intermediate summary representation to better plan and ground the generation of abstractive summaries. During training, we construct an augmented target summary by extracting and prepending its corresponding entity chain (Figure 1, right). At test time, the model must generate

*Work done while Ryan McDonald was at Google.

[1]We use the term ''entity'' broadly and consider named entities, dates, and numbers to form an entity chain.

Transactions of the Association for Computational Linguistics, vol. 9, pp. 1475–1492, 2021. https://doi.org/10.1162/tacl_a_00438
Action Editor: Alexander M. Rush. Submission batch: 6/2021; Revision batch: 8/2021; Published 12/2021.

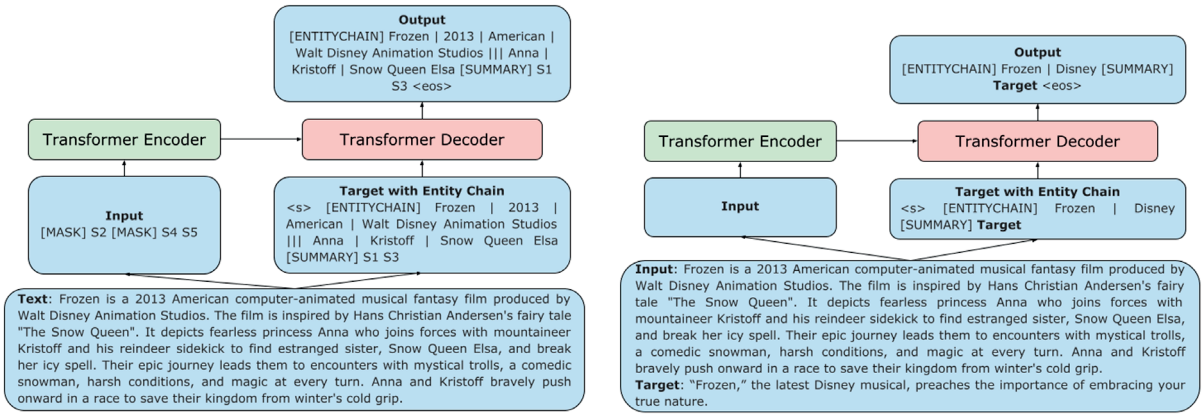

Figure 1: Pretraining and finetuning for abstractive summarization with entity chains.

both the entity chain followed by the summary. Concretely, we use Transformer-based encoder-decoder (Vaswani et al., 2017) models; a transformer encoder first encodes the input and a transformer decoder generates (i) an intermediate summary representation in the form of an entity chain; and (ii) the summary conditioned on the entity chain and the input. We evaluate our approach on four popular summarization datasets: CNN/DailyMail highlight generation (Hermann et al., 2015), XSum extreme summarization (Narayan et al., 2018), SAMSum dialogue summarization (Gliwa et al., 2019), and BillSum (Kornilova and Eidelman, 2019), and show that the state-of-the-art PEGASUS (Zhang et al., 2020) pretrained models finetuned with the planning objective clearly outperform regular finetuning in terms of entity specificity and planning in generated summaries on all datasets. We further demonstrate that this simple planning mechanism can be easily used for pretraining summarization models to do entity-level content planning and summary generation. Similar to PEGASUS pretraining, we mask important sentences from an input document, extract an entity chain from the masked sentences, and generate these gap-sentences prepended with their entity chain from the rest of the document (Figure 1, left). We see further gains with pretraining achieving state of the art performance on XSum in terms of ROUGE. We further demonstrate how the entity-level content planning in summarization can be easily leveraged to mitigate hallucinations in abstractive summaries. In particular, we modify the predicted entity chain to only keep entities that are seen in the document and then generate the summary prompted with the modified entity chain, outperforming state-of-the-art approaches when evaluated automatically and by humans for faithfulness. Our main contributions are as follows:

Planned and Grounded Abstractive Summarization We introduce a novel training objective to neural summarization models for content planning with entity chains, and study integrating it with supervised finetuning and self-supervised pretraining objectives without altering the models themselves. As the entity chains are extracted from the reference summaries during training, our models learn to ground the generation of summaries to the entity chains found in them. Hence, we refer to this objective by FROST for its ability to try to ''FReeze entity-level infOrmation in abstractive SummarizaTion with planning.''

Controlled Abstractive Summarization with Entity Chains FROST provides a very effective knob for entity-level content modification in abstractive summaries. In this paper we empirically demonstrate how FROST is critical for faithfulness by enabling the *drop-prompt* mechanism where we drop out hallucinated entities from the predicted content plan and prompt the decoder with this modified plan to generate faithful summaries. We further qualitatively demonstrate that FROST enables generation of summaries (i) with topical diversity by choosing different sets of entities from the source to plan what we want to discuss in the summary, and (ii) with style diversity by reordering entities in the predicted plan to get an equivalent summary but with a different entity emphasis.

Our codes to process summarization datasets to do FROST-style content planning and generation,

models and predictions are available at https://github.com/google-research/google-research/tree/master/frost.

2 Related Work

Content Planning for Summarization. Traditional methods argue on the granularity of linguistic, domain, and communicative information included in the source representation needed in order to plan and build better summary representations. Some argued to use a deep semantic analysis of the source text, such as Rhetorical Structure Theory (RST; Mann and Thompson, 1988) or MUC-style representations (McKeown and Radev, 1995) to interpret the source texts, while others used a shallow semantic analysis using only word frequency (Luhn, 1958) or lexical chains (Barzilay and Elhadad, 1997).

Recent encoder-decoder models for text generation (Bahdanau et al., 2015; Sutskever et al., 2014; Vaswani et al., 2017; Rothe et al., 2020; Lewis et al., 2020; Raffel et al., 2019; Zhang et al., 2020) tend to perform text generation in an end-to-end setting, which means that most approaches do not explicitly model content planning. Wiseman et al. (2018) and Hua and Wang (2020) start the generation process by building templates which are then used for realization. In data-to-text generation, Puduppully et al. (2019) generate a content plan highlighting which information should be mentioned in the table and in which order. In story generation, there has been some work on exploring events (Martin et al., 2018) or sequences of words (Yao et al., 2019) to plan ahead when creating a consistent story. We are not aware of any similar work on content planning for summarization using encoder-decoder models.

Pretraining for Summarization and Planning. Pretrained transformer-based models have dramatically changed the text generation space, and summarization is no exception to this. Most models focus on task-agnostic pretraining using the left-to-right language modeling objective (Radford et al., 2018; Khandelwal et al., 2019; Dong et al., 2019) or reconstructing the corrupted input text using a sequence-to-sequence framework (Song et al., 2019; Lewis et al., 2020; Raffel et al., 2019). There have been few attempts towards task-specific pretraining for summarization to teach models to do better content selection. Zhang et al. (2020) proposed to select important sentences from an input document as a proxy for human-authored summary and then to generate them from the rest of the document. Narayan et al. (2020) proposed question generation pretraining to better align with summarization. To the best of our knowledge we are the first to propose a solution that incorporates content planning directly into pretraining.

Controlled Abstractive Summarization. There is a growing interest in enabling users to specify high-level characteristics such as length, keywords, and topic in order to generate summaries that better suit their needs. In most cases, these features are first manually provided or estimated using third-party content selectors, and then either (i) encoded along with the input (Fan et al., 2018; He et al., 2020; Dou et al., 2021) or (ii) used to filter beams for lexically constrained decoding (Mao et al., 2020), to control the summary generation. On the contrary, our approach is more generic as we do not rely on external systems or data to augment the input; users can prompt the decoder with a desired content plan in the form of an entity chain to control the summary.

3 Content Planning with Entity Chains

We introduce a new training objective for encoder-decoder generative models to do content planning while summarizing.

Model Formulation. Let d be an input document, we aim to teach our model to first generate a content plan c for the summary s as $p(c|d)$, and then generate the summary s as $p(s|c, d)$. We define the ordered chain of entities observed in the summary s as its content plan. Instead of modeling $p(c|d)$ and $p(s|c, d)$ separately, we take a simpler approach, we train an encoder-decoder model to encode the document d and generate the concatenated content plan and summary sequences $c; s$, essentially the decoder first predicts the entity chain c and then continues predicting the summary s using both c and d. We prefix c and s with special markers ''[ENTITYCHAIN]'' and ''[SUMMARY]'', respectively, as shown in Figure 1. If s consists of multiple sentences, we use sentence markers ''|||'' to mark them in c. The model is trained with the standard maximum-likelihood objective generating the augmented target $c; s$.

Pretraining Content Plans. We modified PEGASUS (Zhang et al., 2020) to pretrain our models for entity-level content planning and summary generation.[2] In particular, we select a maximum of n most important sentences using self-ROUGE from an input document, the selected sentences work as a proxy for a human-authored abstractive summaries for the rest of the document. We construct a target by prepending the selected sentences dynamically with their entity chain. Our model is then trained to generate this target from the rest of the document.[3]

Modeling Entity-Level Lexical Cohesion and Coherence. As entities in the summary contribute to the continuity of lexical meaning of the summary, we hypothesize that by learning to predict the entity chain c in advance, we enforce our model to learn entity-level lexical cohesion (Barzilay and Elhadad, 1997) and coherence (Halliday and Hasan, 1976; Azzam et al., 1999) in the summary. We hope that by doing so our model will be better at predicting pertinent entities (*entity specificity*) in their right order (*entity planning*) in generated summaries; the prediction of entities in the entity chain c (as $p(c|d)$ in FROST) will be less susceptible to local correlations compared to when predicting them directly in the summary s (as $p(s|d)$). Furthermore, as c is predicted in advance and the generation of s is grounded to c,[4] our model will be better equipped to predict correct events relating to different entities in c with full access to c and not just entities to the left, already decoded.

Controlled Generation with Entity Prompts. An advantage of training to generate the summary s following the generation of the plan c using the same decoder is that now during the inference time the decoder can be easily prompted with any desired content plan c' to control the content in the output summary s' with a probability of $p(s'|c', d)$. In Section 5, we prompt our decoder with modified content plans to mitigate hallucinations and to generate diverse summaries.

[2]We experimented with PEGASUS, but our technique can be used with any pretraining objectives that require sentence-level input corruptions.

[3]The PEGASUS objective uses a summary-document length ratio to select n. This could lead to an undesirably long summary when the input document is very long. Modeling such summaries prepended with long entity chains effectively is beyond the limit of our decoders (256 sentencepieces). Hence, we set $n = 5$.

[4]Here, s is not strictly constrained to the entity chain c. We hope that this will happen given c is extracted from s during the training time. Future work will focus on constraining s to c, e.g., using a checklist model (Kiddon et al., 2016) or entity-chain constrained decoding (Mao et al., 2020).

4 Experimental Setup

4.1 Base and Large Models

We experiment with both base and large transformer architectures (Vaswani et al., 2017). The base architecture has $L = 12$, $H = 768$, $F = 3072$, $A = 12$ (223M parameters) and the large architecture has $L = 16$, $H = 1024$, $F = 4096$, $A = 16$ (568M parameters), where L denotes the number of layers for encoder and decoder Transformer blocks, H for the hidden size, F for the feed-forward layer size, and A for the number of self-attention heads. All pretrainings are done with a batch size of 1024, whereas all finetuning experiments are done with a smaller batch size of 256. For optimization, we use Adafactor (Shazeer and Stern, 2018) with square root learning rate decay and dropout rate of 0.01 during pretraining and 0.0001 during finetuning. All finetuned models were decoded with a beam size of 8 and a length-penalty of 0.8.

4.2 Datasets and Entity Annotations

Pretraining Datasets. Following Zhang et al. (2020), our model pretraining also relied on two large Web corpora which were processed to look like plain text: (i) **C4** (Raffel et al., 2019) is composed of 350M Web pages that were obtained from Common Crawl, and (ii) **HugeNews** (Zhang et al., 2020) is composed of 1.5B news and news-like articles from 2013–2019. This dataset includes articles from multiple allowlisted sources including news publishers, high-school newspapers, and blogs.

Abstractive Summarization Datasets. We evaluate our models on four summarization datasets: CNN/DailyMail highlight generation (Hermann et al., 2015), XSum extreme summarization (Narayan et al., 2018), SAMSum dialogue summarization (Gliwa et al., 2019), and BillSum summarizing US Congressional bills (Kornilova and Eidelman, 2019). We use the publicly available versions through the TFDS Summarization Datasets.[5] We use the original

[5]`https://www.tensorflow.org/datasets/catalog`.

			Target Summaries (Validation)						
	Size		avg.	avg.	avg.	% target	total		
Dataset	train/dev/test	Case	sent.	ent.	uniq. ent.	(no ent.)	named	date	number
BillSum	18.9k/–/3.3k	cased	4.38	14.78	11.10	0.18	28931	2578	16646
CNN/DailyMail	287k/13.4k/11.5k	cased	4.11	7.55	6.92	0.10	74292	3569	23094
SAMSum	14.7k/818/819	cased	2.03	3.59	3.01	0.37	2594	33	309
XSum	204k/11.3k/11.3k	cased	1.00	2.81	2.80	5.97	24682	777	6287

Table 1: Abstractive summarization datasets studied in this work. We report on their train/validation/test sizes and how they were processed (cased/uncased). To better understand the effect of summary planning with entity chains, we report on average number of sentences (avg. sent.), average number of entities (avg. ent.) and average number of unique entities (avg. uniq. ent.), per target in validation sets. We also report on total number of named entities, date, and number in target summaries.

train/validation/test splits for them. For BillSum, where the validation split was not provided, we split 10% of the training set to serve as validation. Inputs and outputs were truncated to 512 and 128 for XSum and SAMSum, and 1024 and 256 for CNN/DailyMail and BillSum. Table 1 provides more insights into these datasets to better understand the effect of summary planning with entity chains.

Entity Chain Annotation. We experimented with entity chains consisting of named entities, dates, and numbers. We annotate the whole document in the pretraining datasets to allow dynamic construction of summaries and their entity chains during pretraining. We only annotate the reference summaries for the finetuning summarization datasets. We use a BERT-based tagger trained on CoNLL-2003 data (Tjong Kim Sang and De Meulder, 2003) to identify named entities, and regular expressions to identify dates and numbers (Guu et al., 2020). See Table 1 for the number of named entities, dates and numbers found in different datasets.

Tables 6 and 7 in Appendix A present hyperparameters used for pretraining and finetuning PEGASUS and FROST base and large-sized models. We used Cloud TPU v3 accelerators for training.

4.3 Evaluation Measures

We evaluate our FROST models on ROUGE, *entity specificity*, *entity planning*, *faithfulness* using automatic and human evaluations, and *overall quality* by humans. Our models predict a summary plan in the form of an entity chain, followed by a summary. All evaluations are done on the summary, the predicted entity chains are stripped out.

Summary-level ROUGE. We report ROUGE F1 scores (Lin and Hovy, 2003) to assess generated summaries.[6]

Entity Planning. We evaluate the quality of content plans learned by our models by assessing the entities and the order in which they appear in the summary. We annotate both predicted and reference summaries with named entities, dates, and numbers, and report on ROUGE F1 scores for entity chains found in the predicted summaries against corresponding reference entity chains.

Entity Specificity. We compare entities in predicted and reference summaries, and report average entity F1-scores (ENTF1) in predicted summaries. We lower-case and remove duplicate entities; we report on the exact match with reference entities.

Faithfulness. For entity-level faithfulness, we report on ENTPREC, measuring the precision of entities in predicted summaries against their input documents. ENTF1, in comparison, evaluates predicted summaries for entity specificity against reference summaries and is not a measure for faithfulness. For summary-level faithfulness, we follow Maynez et al. (2020) and report on textual entailment (Pasunuru and Bansal, 2018; Falke et al., 2019; Kryscinski et al., 2020). In particular, we report the probability of a summary

[6]We lowercased candidate and reference summaries and used `pyrouge` with parameters ''-a -c 95 -m -n 4 -w 1.2.''

entailing (*Entail.*) its input document using an entailment classifier trained by fine-tuning an uncased BERT-Large pretrained model (Devlin et al., 2019) on the Multi-NLI dataset (Williams et al., 2018).

We further assess summary faithfulness by humans. Our annotators, proficient in English, were tasked to read the document carefully and then grade its summary on a scale of 1–5 (*fully unfaithful*, *somewhat unfaithful*, *50-50*, *somewhat faithful*, and *fully faithful*); a summary is ''fully faithful'' if all of its content is fully supported or can be inferred from the document.

Overall Summary Quality. Finally, we assess the overall quality of summaries by humans. We ask our annotators to read the document carefully and then grade its summary on a scale of 1–5 (*poor summary*, *better than poor*, *okay summary*, *better than okay*, and *great summary*). To improve the annotator agreement, we clearly define 4 features that a ''great summary'' should have: *Relevant* (summary should have most important information from the document), *Accurate* (summary should be accurate with respect to the document or the expert knowledge),[7] *Concise* (summary should not have redundant or less-important content), and *Fluent* (summary should be grammatically correct and coherent). A *great summary* should pass on all 4 features, a *better than okay* should have 3 out of 4 features, and so on.

For both human assessments, we collected 3 ratings for each (document, summary) pair and report the average of all assigned labels (1–5) to a system. We conducted 3 pilot studies for each setup to train our annotators with examples to improve their understanding of the task. Additionally, extra measures were taken to improve agreements among our annotators. For example, for the faithfulness assessment, when one of *somewhat unfaithful*, *50-50*, and *somewhat faithful* were selected, annotators were asked to also specify what was faithful or unfaithful in the summary. Similarly for the overall quality assessment, when one of *better than poor*, *okay summary*, and *better than okay* were selected, they were asked to list all features on which the candidate summary fails.

[7]With *accurate* we mean *factual* to the background knowledge and not just *faithful* to the document; as it is natural to construct summaries that integrate with the author's background knowledge (Maynez et al., 2020).

Sentence-level Entity Chain with Target: [EntityChain] Frozen | Disney [Summary] "Frozen," the latest Disney musical, preaches the importance of embracing your true nature. [EntityChain] Princess Anna | Kristoff | Snow Queen Elsa [Summary] It depicts fearless Princess Anna who joins forces with mountaineer Kristoff and his reindeer sidekick to find estranged sister, Snow Queen Elsa, and break her icy spell.

Summary-level Entity Chain with Target: [EntityChain] Frozen | Disney ||| Princess Anna | Kristoff | Snow Queen Elsa [Summary] "Frozen," the latest Disney musical, preaches the importance of embracing your true nature. It depicts fearless Princess Anna who joins forces with mountaineer Kristoff and his reindeer sidekick to find estranged sister, Snow Queen Elsa, and break her icy spell.

Original Target: "Frozen," the latest Disney musical, preaches the importance of embracing your true nature. It depicts fearless Princess Anna who joins forces with mountaineer Kristoff and his reindeer sidekick to find estranged sister, Snow Queen Elsa, and break her icy spell.

Figure 2: An example of sentence-level and summary-level entity chains along with the reference summary.

Figures 9 and 10 in Appendix B show detailed instructions for human evaluations for faithfulness and overall quality of summaries, respectively.

5 Results

5.1 FROST Ablations

Sentence-Level vs Summary-Level Planning. Let the summary s consists of m sentences $s_1 \dots s_m$ and c_i be the entity chain for the sentence s_i, we consider generating the summary s in two ways. *Sentence-level* approach trains a model to generate s by consecutively generating the sentence-level content plan c_i followed by its summary sentence s_i with a probability $p(c_i s_i | c_1 s_1 \dots c_{i-1} s_{i-1}; d)$; d is the input document. *Summary-level* approach first generates a summary-level content plan $c = c_1 ||| \dots ||| c_m$ and then continue generating the summary s with a probability $p(s|c; d)$; $|||$ are sentence markers. The summary-level planning is arguably better suited for summarization than the sentence-level planning. By planning the whole summary beforehand, the summary-level planner would be (i) less susceptible to local correlations than the sentence-level planning while generating the entities, and (ii) a better microplanner in deciding sentence boundaries and avoiding verbose summaries (Reiter and Dale, 1997). See examples for sentence-level and summary-level entity chains in Figure 2.

We finetuned our large models initialized with the PEGASUS checkpoint on the CNN/DailyMail

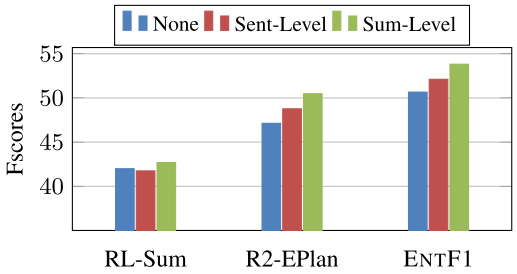

Figure 3: Sentence-level vs summary-level entity chains. We report summary-level ROUGE-L (RL-Sum), entity chain-level ROUGE-2 (R2-EPlan), and EntF1 on the CNN/DailyMail validation set. Similar observations were made for other measures.

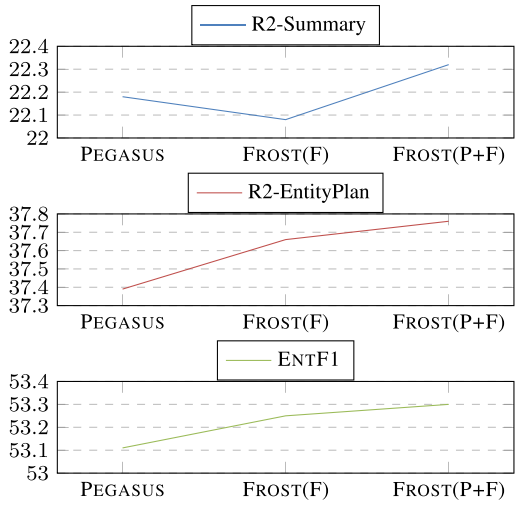

Figure 4: Finetuning results on the XSum validation set using one of the base-sized pretrained models: Pegasus, Frost(F), and Frost(P+F). All pretrained models were trained for 1.5m steps. See text for more details. We only report on a subset of measures, similar observations were made for other measures.

dataset for sentence-level vs summary-level entity chain ablations. We did not do this study on the XSum dataset as the XSum summaries are single sentences, which means that sentence-level and summary-level entity chains are the same. In contrast, the CNN/DailyMail summaries consists of multi-sentence highlights. Results are presented in Figure 3.

We found that the summary-level planning is not only superior to the sentence-level planning, it helps with generating better quality summaries in terms of summary-level ROUGE (RL-Sum), entity planning (R2-EPlan), and entity specificity (EntF1). For the rest of the pretraining or finetuning experiments, we focused on summary-level planning unless specified otherwise.

Pretraining for Planning from Scratch. We pretrained three base-sized models from scratch: Pegasus-base pretrained with the original gap-sentence objective (Zhang et al., 2020) for 1.5m steps, Frost(F)-base pretrained with the gap-sentences prepended with their entity chain for 1.5m steps, and Frost(P+F)-base first pretrained with the Pegasus objective for 1m steps and then with the Frost objective for another 500k steps. Maximum input and output lengths were set to 512 and 256 sentencepieces during pretraining, respectively. We finetuned these three models on the XSum dataset. Results are shown in Figure 4.

First of all, our results confirm that the pretraining for planning is beneficial for summarization; both Frost(F) and Frost(P+F) learned better content plans (in terms of entity chain-level ROUGE and EntF1) than Pegasus without sacrificing the summary quality (in terms of summary-level ROUGE) significantly. Interestingly, the Frost(P+F) fine-tuned model outperformed Frost(F) on all measures confirming that the pretraining for planning and summarization is more effective when started with summarization pretrained models such as Pegasus than when trained jointly from scratch. Hence, for our future pretraining experiments we initialize our model with existing Pegasus checkpoint and continue pretraining for content planning and refer to them as Frost for simplicity.

Effect of Pretraining Longer for Planning. Based on our findings, we pretrain a large Frost model for planning for summarization starting with an existing Pegasus-Large checkpoint. Figure 5 presents results for finetuning these models on the XSum and CNN/DailyMail datasets at various steps during pretraining.

Similar to our findings in Figure 4, our results with large models further confirm the advantages of the pretraining for planning; improvement over the Pegasus baseline were larger for the XSum dataset than for the CNN/DailyMail dataset. We achieved the best performance on both datasets at the 1m pretraining step. We use the Frost model at 1m pretraining step for our finetuning experiments.

5.2 Abstractive Summarization Results

Table 2 presents our final results from finetuning our large models on summarization datasets. For

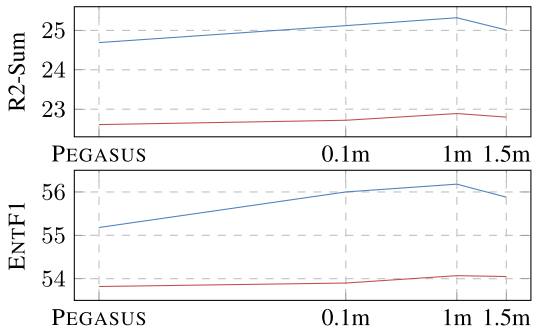

Figure 5: Finetuning results on the XSum (in blue) and CNN/DailyMail (in red) validation sets at various steps during pretraining FROST-Large. Instead of pretraining from scratch, we start with a PEGASUS-Large checkpoint, and continue pretraining for additional 1.5m steps with the planning objective. We report finetuning results for the PEGASUS finetuned baseline and our models at 0.1m, 1m, and 1.5m steps.

each dataset, we first report results from earlier work directly taken from corresponding papers; our results are in the bottom blocks for each dataset. We finetune our own PEGASUS using the standard approach (document to summary). We report results from FROST (ECP; Entity Chain Planning), that is, PEGASUS finetuned with the FROST objective (document to content plan and summary). Finally, we report on FROST (ECPP; Entity Chain Planning with Pretraining), our models both pretrained and finetuned with the FROST objective.

We find that even simply finetuning an existing PEGASUS pretrained model to do content planning and summarization (as in FROST (ECP)) leads to improvements in entity specificity (ENTF1) and the quality of entity plans in summaries (Entity Planning ROUGE), across all datasets. In fact, in some cases, better content plans lead to large improvements in summary-level ROUGE as well, for example, FROST (ECP) improve on PEGASUS from 44.05/21.69/40.98 to 44.85/21.83/41.80 on ROUGE scores for CNN/DailyMail summaries.

The pretraining for content planning and generation in FROST (ECPP) further improves the entity chain quality for CNN/DailyMail, XSum and BillSum, and the summary-level ROUGE for CNN/DailyMail and XSum. Our FROST models establish new state-of-the-art ROUGE results on XSum. For CNN/DailyMail, we perform inferior to CTRLsum (He et al., 2020) and GSum (Dou et al., 2021) on ROUGE scores. However, we outperform CTRLsum on entity planning and specificity. Further discussion on comparing our methods to controlled summarization systems such as CTRLsum can be found in Section 5.5.

Model	Summary R1/R2/RL	Entity Planning R1/R2/RL	Specificity ENTF1
CNN/DailyMail			
LEAD-3	40.31/17.83/36.45	57.41/42.32/47.16	46.22
EXT-ORACLE*	57.03/34.38/53.12	68.79/55.77/59.02	59.91
RoBERTaShare	39.25/18.09/36.45	–	–
MASS	42.12/19.50/39.01	–	–
UniLM	43.33/20.21/40.51	–	–
T5	43.52/21.55/40.69	–	–
BART	44.16/21.28/40.90	–	–
ProphetNet	44.20/21.17/41.30	–	–
PEGASUS	44.16/21.56/41.30	–	–
GSum	**45.94**/22.32/42.48	–	–
CTRLsum	45.65/**22.35**/**42.50**	62.19/48.35/52.53	51.69
PEGASUS (ours)	44.05/21.69/40.98	61.12/46.46/52.40	49.93
FROST (ECP)	44.85/21.83/41.80	**64.34**/49.85/**54.98**	53.17
FROST (ECPP)	45.11/22.11/42.01	64.28/**49.86**/54.96	**53.22**
XSum			
LEAD-1	16.66/1.85/12.26	21.45/4.36/19.48	5.36
EXT-ORACLE*	28.81/8.61/21.97	32.64/13.34/28.85	18.78
RoBERTaShare	38.52/16.12/31.13	–	–
MASS	39.75/17.24/31.95	–	–
BART	45.14/22.27/37.25	–	–
GSum	45.40/21.89/36.67	–	–
PEGASUS	47.60/24.83/39.64	–	–
PEGASUS (ours)	47.56/24.87/39.40	62.15/39.76/56.36	53.48
FROST (ECP)	47.44/24.54/39.24	63.57/40.45/57.65	54.62
FROST (ECPP)	**47.80**/**25.06**/**39.76**	**64.09**/**41.07**/**58.18**	**55.49**
SAMSum			
(Gliwa et al., 2019)	40.99/17.72/38.30	–	–
PEGASUS (ours)	52.27/**28.34**/**47.83**	72.42/51.07/64.85	81.17
FROST (ECP)	**52.39**/27.70/47.82	74.42/**55.32**/66.35	**83.60**
FROST (ECPP)	51.86/27.67/47.52	**75.02**/55.19/**66.80**	**83.60**
BillSum			
PEGASUS	59.67/41.58/47.59	–	–
PEGASUS (ours)	59.33/**41.60**/54.80	69.99/62.79/66.17	61.91
FROST (ECP)	58.76/40.34/54.03	70.84/63.59/66.86	62.38
FROST (ECPP)	**59.50**/41.17/**54.85**	**71.67**/**64.56**/**67.79**	**63.38**

Table 2: Final results on abstractive summarization datasets compared with the previous SOTA. We report results from RoBERTaShare (Rothe et al., 2020), MASS (Song et al., 2019), UniLM (Dong et al., 2019), T5 (Raffel et al., 2019), BART (Lewis et al., 2020), ProphetNet (Qi et al., 2020), (Gliwa et al., 2019), CTRLsum (He et al., 2020), GSum (Dou et al., 2021), and PEGASUS (Zhang et al., 2020). We also include commonly reported LEAD-n baselines (selecting top n sentences from the document) and extractive oracle (EXT-ORACLE; selecting best set of sentences from the document with the most overlapping content with its reference summary), for the CNN/DailyMail and XSum datasets. EXT-ORACLES are marked with $*$ and are not directly comparable. All results are reported on the publicly available test sets.

5.3 Controlling Hallucinations

We demonstrate in two ways that planning with entity chains can be useful in mitigating hallucinations in summarization: *data filtering*

and *drop-prompt mechanism* with entity chains. We focused on the XSum dataset for these experiments.[8]

Data Filtering using Entity Chains. During training FROST prepends reference summaries with their entity chains to better plan them. We leverage this to filter the dataset to only keep examples where summaries have fully extractive entity chains; an entity chain is fully extractive if all entities in it can be found in the input document. It's notable that this filtered dataset will not contain novel entities in the summary targets. FROST models trained on this data will ground the summary generation to extracted entity chains while allowing abstraction for non-entity related generations. The resulting XSum dataset has 62.7k/3.5k/3.5k train/validation/test instances. We finetune our models from Table 2 on this filtered dataset and report their performance on the filtered test set (3.5k). We also evaluate them on the rest of the test set (7.8k) where reference entity chains are not fully extractive to their input documents.[9] See Figure 6 for examples.

Drop-Prompt Mechanism. FROST decoders are trained to generate the summary s following the generation of its plan c. To improve faithfulness, we take the predictions from FROST (ECPP)[10] and modify the generated plan c to c_{drop} by dropping entities (or parts of them) that are not found in the input document. We then prompt our decoder with c_{drop} to generate a new summary. We conduct this with both models, one trained on the full dataset and another on the filtered subset. We also report results for the oracle entity chain prompts c_{oracle} for a comparison.

Effect on Summary-level ROUGE, Entity Planning, and Specificity. Results are presented in Table 3. First of all, similar to our results in Table 2 on the full XSum test set, our results with the models trained on the original dataset further validate that the pretraining and finetuning with the content planning objective is equally useful for both test sets: one where entities are simply copied from the input documents and another where novel entities were inferred, to generate corresponding targets.

The data filtering using entity chains are particularly useful for test cases with extractive reference entity chains; models trained on the filtered data lead to much higher ENTF1 (e.g., 65.97 vs 63.03, for FROST (ECPP)) and higher quality entity plans (entitly-level ROUGE of 66.43/30.00/62.48 vs 63.73/30.08/60.10 for FROST (ECPP)), without sacrificing the summary quality substantially (summary-level ROUGE of 44.87/22.05/36.79 vs 45.08/22.01/36.80 for FROST (ECPP)). Unsurprisingly, these models don't do well on the test set with non-extractive reference entity chains, such examples were not seen during training.

The drop-prompt mechanism works particularly well for models trained on the full dataset and evaluated on the test set with extractive reference entity chains. For example, the entity planning ROUGE scores and ENTF1 for

GOLD: Walsall have signed defender Luke Leahy on a two-year contract from Scottish Championship side Falkirk.
Trained on the original XSum dataset
PEGASUS: Walsall have signed defender Paddy Leahy from Scottish Championship side Falkirk on a three-year deal.
FROST (ECP): [ENTITYCHAIN] Walsall | Falkirk | Paddy Leahy | two [SUMMARY] Walsall have signed Falkirk defender Paddy Leahy on a two-year deal.
FROST (ECPP): [ENTITYCHAIN] Walsall | Falkirk | Liam Leahy | two [SUMMARY] Walsall have signed Falkirk defender Liam Leahy on a two-year deal.
FROST (ECPP), c_{drop}: [ENTITYCHAIN] Walsall | Falkirk | Leahy [SUMMARY] Walsall have signed Falkirk defender Leahy on a free transfer.
Trained on the filtered XSum dataset
PEGASUS: Walsall have signed Falkirk defender Declan Leahy for an undisclosed fee.
FROST (ECP): [ENTITYCHAIN] Walsall | Falkirk | Paddy Leahy [SUMMARY] Walsall have signed Falkirk defender Paddy Leahy for an undisclosed fee.
FROST (ECPP): [ENTITYCHAIN] Walsall | Falkirk | Conor Leahy [SUMMARY] Walsall have signed Falkirk defender Conor Leahy for an undisclosed fee.
FROST (ECPP), c_{drop}: [ENTITYCHAIN] Walsall | Falkirk | Leahy [SUMMARY] Walsall have signed Falkirk defender Leahy for an undisclosed fee.

Figure 6: Example XSum predictions for models presented in Tables 3 and 4. We highlight entities in orange that are not faithful to the input document. Entities in green are faithful to the input document.

[8] The CNN/DailyMail summaries are mostly extractive in nature, these studies are less interesting as they don't lead to significant differences when assessing faithfulness (He et al., 2020).

[9] Such data divergence is not unique to XSum, around 30% of CNN/DailyMail summaries also have reference entity chains that are not fully extractive to their input documents. Writing these summaries requires either document-level inference or the background knowledge of the input documents to generate novel entities or numbers.

[10] The drop-prompt mechanism can be used with FROST (ECP) also, we have simply used the best among FROST (ECP) and FROST (ECPP) on the XSum set in terms of summary-level ROUGE, entity planning and ENTF1.

Model	Test set with Extractive Entity Chains Only (3.5k)			Test set with Non-Extractive Entity Chains Only (7.8k)		
	Summary R1/R2/RL	**Entity Planning** R1/R2/RL	**Specificity** ENTF1	**Summary** R1/R2/RL	**Entity Planning** R1/R2/RL	**Specificity** ENTF1
Models trained on the original XSum dataset (204k/11.3k/11.3k)						
PEGASUS	44.67/21.75/36.37	61.54/28.84/58.02	60.88	48.85/26.26/40.75	62.42/44.64/55.62	50.17
FROST (ECP)	44.53/21.38/36.28	63.60/29.20/60.01	62.04	48.74/25.95/40.56	63.56/45.48/56.60	51.31
FROST (ECPP)	**45.08/22.01/36.80**	63.73/30.08/60.10	63.03	**49.02/26.42/41.08**	**64.25/45.98/57.32**	**52.12**
$(d \rightarrow c_{\text{drop}}; s)$	44.97/21.97/36.77	**67.10/30.19/63.58**	64.41	44.93/21.41/37.39	48.89/30.58/43.76	33.68
$(d \rightarrow c_{\text{oracle}}; s)^*$	50.26/27.33/43.12	98.55/53.94/98.53	98.36	61.80/40.55/56.20	99.10/92.10/99.08	98.69
Models trained on the filtered set with extractive entity chains only (62.7k/3.5k/3.5k)						
PEGASUS	44.10/21.24/35.87	65.66/28.69/61.86	63.47	44.72/21.80/36.94	52.87/35.39/46.99	40.77
FROST (ECP)	44.29/21.26/36.04	67.05/29.11/63.02	64.69	44.02/20.91/36.21	49.59/31.16/43.55	38.97
FROST (ECPP)	44.87/22.05/36.79	66.43/30.00/62.48	**65.97**	44.28/21.15/36.59	52.17/34.40/46.25	39.10
$(d \rightarrow c_{\text{drop}}; s)$	44.56/21.80/36.53	65.94/29.51/62.04	65.08	42.93/19.41/35.40	47.29/28.27/41.95	31.17
$(d \rightarrow c_{\text{oracle}}; s)^*$	49.60/26.64/42.41	97.53/53.35/97.51	97.41	59.80/38.00/54.10	97.95/90.57/97.84	97.11

Table 3: Performance on XSum summaries when models are trained on the dataset with extractive entity chains only (data filtering, the bottom block) and when novel entities are dropped from the predicted entity chains using the drop-prompt mechanism (c_{drop}). Results with * are with oracle entity chain prompts. We report results on the filtered test set with extractive entity chains only (3.5k) and the rest of the test set with novel entities in the targets (7.8k). Best results are bold faced.

Model	Test set with Extractive Entity Chains Only (3.5k)						Test set with Non-Extractive Entity Chains Only (7.8k)					
	Faithfulness				**Overall**		**Faithfulness**				**Overall**	
	Entail.	**ENTPREC**	**Human**	**Agree.**	**Human**	**Agree.**	**Entail.**	**ENTPREC**	**Human**	**Agree.**	**Human**	**Agree.**
Models trained on the original XSum dataset (204k/11.3k/11.3k)												
PEGASUS	0.613	0.800	4.20	0.74	4.09	0.69	0.402	0.361	3.15	0.71	2.93	**0.80**
FROST (ECP)	0.606	0.770	4.22	0.75	**4.23**	0.70	0.379	0.317	3.11	0.73	2.85	0.77
FROST (ECPP)	0.589	0.751	4.13	0.72	4.11	0.66	0.371	0.357	3.31	**0.79**	2.81	0.77
$(d \rightarrow c_{\text{drop}}; s)$	0.650	**0.943**	4.09	0.73	4.09	0.64	0.441	0.746	3.53	0.75	3.13	0.79
Models trained on the filtered set with extractive entity chains only (62.7k/3.5k/3.5k)												
PEGASUS	**0.667**	0.887	4.39	**0.80**	4.14	0.69	0.389	0.501	3.37	**0.79**	3.09	0.76
FROST (ECP)	0.581	0.858	4.27	0.76	4.16	0.66	0.442	0.548	3.43	0.72	3.01	0.73
FROST (ECPP)	0.502	0.806	4.19	0.77	4.19	0.66	**0.465**	0.491	3.55	0.76	3.16	0.76
$(d \rightarrow c_{\text{drop}}; s)$	0.533	**0.943**	**4.41**	0.75	4.18	**0.72**	0.453	**0.826**	**3.85**	0.74	**3.46**	0.77

Table 4: Faithfulness assessment using automatic (Entailment and ENTPREC) and human evaluations, and overall quality assessment by humans for models presented in Table 3. Following Durmus et al. (2020), agreement (Agree.) is computed by taking the percentage of the annotators that annotate the majority class for the given (document, summary) pair. Best results are bold faced.

FROST models improve from 63.73/30.08/60.10 to 67.10/30.19/63.58, and, 63.03 to 64.41, respectively. Interestingly, we do not observe a significant drop in the summary-level ROUGE scores (45.08/22.01/36.80 vs 44.97/21.97/36.77). Basically, our results demonstrate that when models are trained on the noisy data (with data divergence issues, common in summarization datasets (Dhingra et al., 2019; Maynez et al., 2020)), the drop-prompt mechanism is very effective in generating high quality summaries when entities need to be simply copied from the input documents.

The drop-prompt mechanism doesn't help models when they are already trained on the filtered training set. Also this mechanism is counter intuitive to use for the test set where novel entities need to be inferred to generate targets, we observe drops for models irrespective of how they were trained, either using the full training set or the filtered set.

Effect on Faithfulness. The planning objective in FROST itself does not ensure faithfulness; planning is done with the entities in the target, if the target is noisy, its plan may also be noisy (see Figure 6 for predicted entity chains with hallucinated entities). But the planning with the entity chains facilitates the data filtering and

the drop-prompt mechanism, using entity chains. Table 4 presents our results assessing them for faithfulness. We randomly selected 50 documents for each of test sets (with extractive or non-extractive reference entity chains) and assessed their summaries from all 8 systems (except with c_{oracle}) from Table 3.

The data filtering using entity chains is extremely useful for improving faithfulness in summaries. We see improvements for all models trained on the filtered set (Table 4, bottom) compared to their counterparts trained on the full training set (Table 4, top), when evaluated using ENTPREC and by humans for faithfulness. We don't observe similar improvements in entailment. Contrary to findings in Maynez et al. (2020), our results suggest that the entailment scores are not a reliable indicator of faithfulness for documents with extractive reference entity chains.

The drop-prompt mechanism is also very powerful in improving faithfulness. Irrespective of which training data were used to train the models and which test sets they were evaluated on, we see improvements in entailment scores, ENTPREC, and the human assessment of faithfulness across the board, except for a single case where the entailment score slightly drops from 0.465 to 0.453. Figure 6 demonstrates how we drop hallucinated entities '*Liam*', '*two*', and '*Conor*' from entity chains to enforce models to generate faithful summaries grounded to the modified entity chains.

We achieve the best performance in terms of ENTPREC and the human assessment of faithfulness when both the data filtering and the drop-prompt mechanism were used. We achieve 0.943 for ENTPREC and 4.41 for faithfulness for the test set with extractive entity chains only, and 0.826 for ENTPREC and 3.85 for faithfulness for the test set with non-extractive entity chains. Humans also found that the predictions from these models were the best in terms of overall summary quality.[11]

Finally, we carried out pairwise comparisons for human assessments for faithfulness and overall summary quality for all models (using a one-way ANOVA with post-hoc Tukey HSD tests; $p < 0.01$). Interestingly, differences among all model pairs for both faithfulness and overall summary quality are insignificant when evaluated on the test set with extractive reference entity chains only. On the more challenging test set with non-extractive entity chains, FROST (ECPP) trained on the filtered training set and with the drop-prompt mechanism is significantly better than all other models except for (i) FROST (ECPP) trained on the filtered training set but without the drop-prompt mechanism and (ii) FROST (ECPP) trained on the original training set and with the drop-prompt mechanism, for both faithfulness and overall summary quality. All other pairwise differences are insignificant.

[11] The extractive systems (LEAD and EXT-ORACLE, reported in Table 2) will have perfect ENTPREC scores and will always be faithful to the document. Yet, it is not worth evaluating them by humans on XSum. These extractive systems are highly inferior to abstractive systems in terms of ROUGE; even the best extractive system (EXT-ORACLE) gets ROUGE scores of 28.81/8.61/21.97, far below than 47.80/25.06/39.76 for FROST (ECPP), on the full test set.

GOLD: [ENTITYCHAIN] BP | Lord Browne | North Sea [Summary] Former BP chief executive Lord Browne has warned North Sea oil operators' costs must fall in order for them to compete globally.

FROST (ECPP): [ENTITYCHAIN] Browne | BP [Summary] Lord Browne, the former boss of BP, has warned that some oil companies could go out of business if costs continue to rise.

FROST (ECPP), c_{mod}, Style diversity
[ENTITYCHAIN] BP | Browne [Summary] The former boss of BP, Lord Browne, has warned that some oil companies could go out of business if costs continue to rise.
[ENTITYCHAIN] BBC | BP | Browne [Summary] In an interview with the BBC, former BP chief executive Lord Browne has warned the oil and gas industry is in danger of "going to the wall".

FROST (ECPP), c_{mod}, Topical diversity
[ENTITYCHAIN] Brent Crude [Summary] The former boss of the world's biggest oil company has warned that the price of Brent Crude will not recover until next year.
[ENTITYCHAIN] Brent Crude | Unions [Summary] Oil companies will have to cut costs dramatically if they are to survive the fall in the price of Brent Crude, according to the former chief executive of the Unions.

Figure 7: An example of generating summaries with topical and style diversity using modified entity prompts c_{mod} on XSum.

5.4 Generating Diverse Summaries

FROST provides a handle to easily control or manipulate content plan for predicted summaries. In Figure 6, we saw how we can use this to control entity-level hallucinations, but the strength of FROST models go beyond this. Figure 7 shows how FROST models can be effectively used in generating summaries with different entity focus by simply modifying the entity prompt. Future work will focus on how to leverage FROST for synthetic data generation for training improvements.

Model	XSum Summary R1/R2/RL	XSum Entity Planning R1/R2/RL	XSum Specificity EntF1	XSum Avg. Length	CNN/DailyMail Summary R1/R2/RL	CNN/DailyMail Entity Planning R1/R2/RL	CNN/DailyMail Specificity EntF1	CNN/DailyMail Avg. Length
Pegasus ($d \rightarrow s$)	47.56/24.87/39.40	62.15/39.76/56.36	53.48	22.27	44.05/21.69/40.98	61.12/46.46/52.40	49.93	68.65
CTRLsum ($k; d \rightarrow s$)	–	–	–	–	45.65/22.35/42.50	62.19/48.35/52.53	51.69	74.46
Frost ($d \rightarrow c; s$)	47.80/25.06/39.76	64.09/41.07/58.18	55.49	21.09	45.11/22.11/42.01	64.28/49.86/54.96	53.22	65.75
CTRLsum ($k_{\text{oracle}}; d \rightarrow s$)	–	–	–	–	64.72/40.56/61.02	78.18/67.39/66.17	71.35	71.76
Pegasus ($d; c_{\text{oracle}} \rightarrow s$)	56.58/34.64/49.87	94.48/75.17/93.32	93.59	22.12	56.46/33.62/53.51	86.35/80.54/83.76	82.41	64.84
Pegasus ($c_{\text{oracle}}; d \rightarrow s$)	57.60/35.62/51.11	98.13/79.27/97.58	97.90	22.13	61.66/38.43/58.75	97.97/96.11/97.72	97.43	64.24
Frost ($d \rightarrow c_{\text{oracle}}; s$)	58.24/36.47/52.16	98.93/80.32/98.91	98.59	21.80	61.85/38.95/59.00	98.02/96.19/97.96	97.31	63.86

Table 5: Comparison of Frost with encoder-guided control summarization. The results in the top block are repeated from Table 2 for comparison. The bottom block presents oracle results with access to oracle keywords (k_{oracle}) in CTRLSum or oracle entity promts (c_{oracle}) in Frost and Pegasus. d and s stand for the input document and the output summary, respectively. All the results are reported on the full test sets.

5.5 Comparison with Encoder-Guided Control Summarization

We compare the decoding strategy in Frost to encoder-guided control summarization systems (He et al., 2020; Dou et al., 2021). Table 5 presents our results. In particular, we report on CTRLsum (He et al., 2020); first, a keyword extraction system (BERT-based sequence tagger) is used to extract keywords (k) from the input document d, and then, the extracted keywords are encoded along with the input document (as $k; d$) to generate the summary. We also finetune Pegasus where the entity chain (c) is encoded along with the input document d (as $d; c$ or $c; d$). We only report the oracle results (with c_{oracle}) for these models; like CTRLsum, generating an entity chain c during inference will require training an additional entity chain generator, which is out of scope of this paper.

There are several advantages of Frost-style decoding. Unlike encoder-guided control summarization systems, Frost models can be used in a usual way to generate summaries for input documents without relying on external systems to augment them during inference. Additionally, users can modify the generated entity prompts or provide their desired entity prompts to control the generated summaries (see Section 5.3 and 5.4). To the best of our knowledge, Frost is the first decoder-prompted control summarization model.

Our results in Table 5 show that the prompting the decoder with entity prompts are more effective in generating high-quality and grounded summaries compared to when entity prompts are encoded with the input, especially for abstractive summaries; both Pegasus versions ($d; c_{\text{oracle}}$ and $c_{\text{oracle}}; d$) perform inferior to Frost ($d \rightarrow c_{\text{oracle}}; s$) in terms of summary-level rouge, entity-level rouge and EntF1 on the XSum dataset. CTRLsum ($k_{\text{oracle}}; d \rightarrow s$) achieves better rouge scores than Frost ($d \rightarrow c_{\text{oracle}}; s$) on CNN/DailyMail (64.72/40.56/61.02 vs 61.85/38.95/59.00). This is not surprising as the oracle keywords k_{oracle} in CTRLSum retain most words from the summary found in the source document; as such due to the extractive nature of CNN/DailyMail summaries, k_{oracle} tend to be closer to the surface forms of the reference summaries (rouge scores between k_{oracle} and reference summaries are 52.70/17.37/44.37). The difference in rouge scores narrows down between CTRLsum and Frost to 45.65/22.35/42.50

Gold: Theia, a bully breed mix, was apparently hit by a car, whacked with a hammer and buried in a field. "She's a true miracle dog and she deserves a good life," says Sara Mellado, who is looking for a home for Theia.

CTRLSum: (Oracle keywords) hit car apparently whacked hammer buried field | dog bully breed mix Theia | true miracle deserves good life **(Summary)** A bully breed mix named Theia was hit by a car, apparently whacked with a hammer and buried in a field. Four days after her apparent death, the dog managed to stagger to a nearby farm. "She's a true miracle dog and she deserves a good life," the dog's foster mother says.
Frost: (Oracle Entity Prompt) Theia ||| Sara Mellado | Theia **(Summary)** The dog, now named Theia, was apparently hit by a car and buried. Sara Mellado is raising money to help pay for Theia's medical care.

CTRLSum: (Predicted keywords) stray pooch Washington State hit car head hammer mercy killing buried field survive | dog Theia | found **(Summary)** A stray pooch in Washington State was hit by a car and buried in a field. The dog was apparently hit on the head with a hammer in a mercy killing. She managed to survive and was found by a worker who took her to a vet. Theia is now being cared for at a veterinary hospital.
Frost: (Predicted Entity Prompt) Washington State ||| Theia ||| **(Summary)** A stray dog in Washington State was apparently hit by a car and buried. The dog, now named Theia, survived but still needs surgery. A fundraising page has been set up to help pay for her care.

Figure 8: CNN/DailyMail example predictions from Frost and CTRLSum along with their entity prompts and keywords, respectively.

vs 45.11/22.11/42.01 with automatically extracted keywords. FROST outperforms CTRLsum on entity planning and specificity. It is not clear how well CTRLsum's keyword extraction system will generalize to more abstractive datasets where words in reference summaries are often not found in the source document.

Finally, our FROST models work as a better microplanner and produce concise summaries than those generated by systems without doing content planning or when its done on the input side. For example, the average lengths of CNN/DailyMail test summaries are 68.65, 74.46, 65.75 and 60.70 tokens for PEGASUS, CTRLSum, FROST, and humans, respectively. See example predictions comparing CTRLSum and FROST in Figure 8.

6 Conclusion

In this work, we proposed the use of entity chains, both during pretraining and finetuning, to better plan and ground the generation of abstractive summaries. Our approach achieves state-of-the-art ROUGE results in XSum and SAMSum, and competitive results in CNN/Dailymail. Compared to other guided summarization models, CTRLSum and GSum, which perform slightly better in CNN/Dailymail, our approach is drastically simpler to implement, trains by augmenting the targets only, utilizes no additional parameters than the baseline PEGASUS model, and does not rely on any external systems to augment the inputs during inference. We further demonstrate that by modifying the entity chain prompts, we can easily control hallucinations in summaries.

Acknowledgments

We thank the reviewers and the action editor for their valuable feedback. We would like to thank Rahul Aralikatte, Ankur Parikh, and Slav Petrov for their insightful comments. Many thanks also to Ashwin Kakarla and his team for help with the human evaluations.

References

Saliha Azzam, Kevin Humphreys, and Robert Gaizauskas. 1999. Using coreference chains for text summarization. In *Coreference and Its Applications*. https://doi.org/10.3115/1608810.1608825

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Regina Barzilay and Michael Elhadad. 1997. Using lexical chains for text summarization. In *Intelligent Scalable Text Summarization*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1483

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019, Unified language model pre-training for natural language understanding and generation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 13042–13054. Curran Associates, Inc.

Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. 2021. GSum: 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*, pages 5055–5070, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.454

Tobias Falke, Leonardo F. R. Ribeiro, Prasetya Ajie 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

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54. Melbourne, Australia. Association for Computational Linguistics.

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

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79, Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-5409

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR.

M. A. K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman. London.

Junxian He, Wojciech Kryściński, Bryan McCann, Nazneen Fatema Rajani, and Caiming Xiong. 2020. CTRLsum: Towards generic controllable text summarization. *CoRR*, abs/2012.04281.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735, Pubmed: 9377276

Xinyu Hua and Lu Wang. 2020. PAIR: Planning and iterative refinement in pre-trained transformers for long text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 781–793, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.57

Karen Sparck Jones. 1993. What might be in a summary? *Information Retrieval*, pages 9–26.

Urvashi Khandelwal, Kevin Clark, Dan Jurafsky, and Lukasz Kaiser. 2019. Sample efficient text summarization using a single pre-trained transformer. *CoRR*, abs/1905.08836.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339, Austin, Texas. Association for Computational Linguistics. https://doi.org/10.18653/v1/D16-1032

Anastassia Kornilova and Vladimir Eidelman. 2019. BillSum: A corpus for automatic summarization of US legislation. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 48–56, Hong Kong, China. Association for Computational Linguistics.

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

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. https://doi.org/10.18653/v1/2020.acl-main.703

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.

Hans Peter Luhn, USA. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165. https://doi.org/10.1147/rd.22.0159

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Yuning Mao, Xiang Ren, Heng Ji, and Jiawei Han. 2020. Constrained abstractive summarization: Preserving factual consistency with constrained generation. *CoRR*, abs/2010.12723.

Lara J. Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark O. Riedl. 2018. Event representations for automated story generation with deep neural nets. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2–7, 2018*, pages 868–875. AAAI Press.

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 and Dragomir R. Radev. 1995. Generating summaries of multiple news articles. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'95, pages 74–82, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/215206.215334

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.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1206

Shashi Narayan, Gonçalo Simões, Ji Ma, Hannah Craighead, and Ryan T. McDonald. 2020. QURIOUS: Question generation pretraining for text generation. *CoRR*, abs/2004.11026.

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.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(1):6908–6915. https://doi.org/10.1609/aaai.v33i01.33016908

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future n-gram for sequence-to-Sequence-Pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, Online. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *Technical report, OpenAI*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1901.07291.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280. https://doi.org/10.1017/S1351324997001502

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: Masked sequence to sequence pre-training for language generation. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 5926–5936. PMLR.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147. https://doi.org/10.3115/1119176.1119195

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1101

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

Lili Yao, Nanyun Peng, Ralph M. Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019,*

The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 – February 1, 2019, pages 7378–7385. AAAI Press. https://doi.org/10.1609/aaai.v33i01.33017378

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR.

A Hyperparameters for Pretraining and Finetuning

Tables 6 and 7 present hyperparameters used for pretraining and finetuning PEGASUS and FROST base and large sized models.

B Human Evaluation Instructions

Figures 9 and 10 show the exact instructions and the template used by our annotators for faithfulness and overall summary quality, respectively.

Model, Size	Learning rate	Label smoothing	Init.	Steps	Batch size	Corpus	Lengths (token)	
							input	target
PEGASUS, base	0.01	0.1	random	1.5m	1024	C4/Hugenews	512	256
FROST(F), base	0.01	0.1	random	1.5m	1024	C4/Hugenews	512	256
FROST(P+F), base	0.01	0.1	PEGASUS (1m)	0.5m	1024	C4/Hugenews	512	256
FROST(P+F), large	0.01	0.1	PEGASUS (1.5m)	1.5m	1024	C4/Hugenews	512	256

Table 6: Hyperparameters used for pretraining PEGASUS and FROST base (223M parameters) and large (568M parameters) models. See text in Section 5.1 for more details and how these models were ablated.

Dataset	Learning rate	Label smoothing	Steps	Batch size	Beam size	Beam alpha	Lengths (token)	
							input	target
BillSum	1e-4	0.1	200k	256	8	0.8	1024	256
CNN/DailyMail	1e-4	0.1	175k	256	8	0.8	1024	256
Samsum	1e-4	0.1	20k	256	8	0.8	512	128
XSum	1e-4	0.1	100k	256	8	0.8	512	128

Table 7: Hyperparameters used for finetuning PEGASUS and FROST models on summarization datasets.

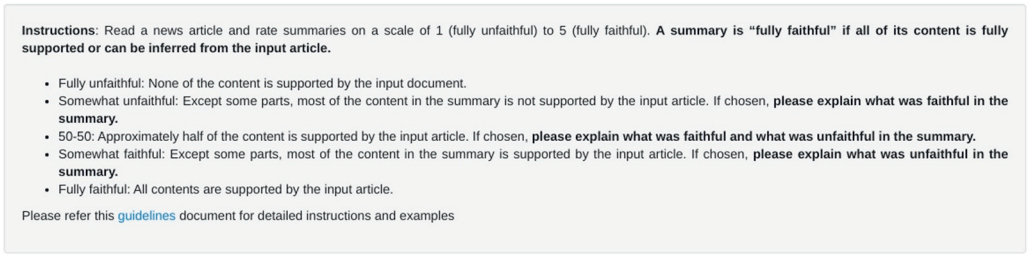

Figure 9: Instructions for human evaluations for faithfulness.

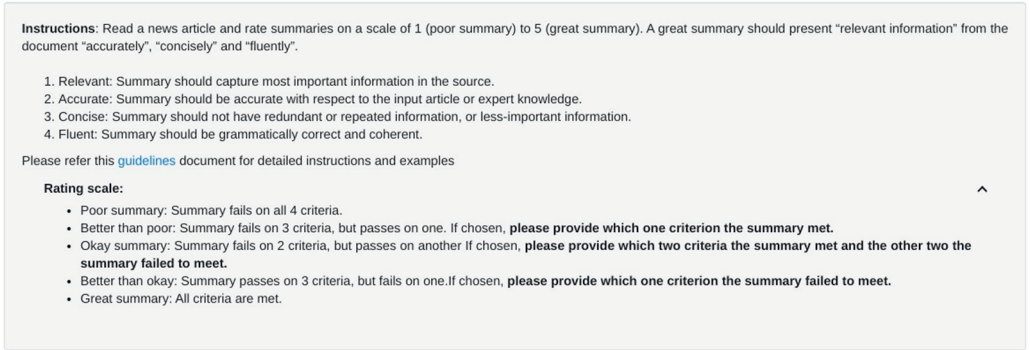

Figure 10: Instructions for human evaluations for overall quality of summaries.