Automatic Document Sketching: Generating Drafts from Analogous Texts

Zeqiu Wu[♢] **Michel Galley**[♠] **Chris Brockett**[♠] **Yizhe Zhang**[♠] **Bill Dolan**[♠]
[♢]University of Washington [♠]Microsoft Research
zeqiuwu1@washington.edu
{mgalley,chrisbkt,yizzhang,billdol}@microsoft.com

Abstract

The advent of large pre-trained language models has made it possible to make high-quality predictions on how to add or change a sentence in a document. However, the high branching factor inherent to text generation impedes the ability of even the strongest language models to offer useful editing suggestions at a more global or *document* level. We introduce a new task, DOCUMENT SKETCHING, which involves generating entire draft documents for the writer to review and revise. These drafts are built from sets of documents that overlap in form – sharing large segments of potentially reusable text – while diverging in content. To support this task, we introduce a Wikipedia-based dataset of analogous documents and investigate the application of weakly supervised methods, including use of a transformer-based mixture of experts, together with reinforcement learning. We report experiments using automated and human evaluation methods and discuss relative merits of these models.

1 Introduction

Large pre-trained language models such as T5 and GPT-3 (Raffel et al., 2019; Brown et al., 2020) have enabled impressive progress on a variety of natural language generation tasks by producing fluent, coherent long texts (Rashkin et al., 2020; Zellers et al., 2019). While automated document-level generation seems tantalizingly within reach, a high branching factor presents significant challenges in tailoring generated documents to the specific requirements of users. Topic drift and "hallucination" of information are endemic to these models (Wiseman et al., 2017). These risks have ensured that end-user applications involving text generation (e.g., Smart Compose, Smart Reply, Grammarly) still require a human to remain in control of content and are restricted to individual sentences or even smaller segments of text (Chen et al., 2019; Kannan et al., 2016; Alikaniotis and Raheja, 2019; Prabhumoye et al., 2019; Faltings et al., 2021).

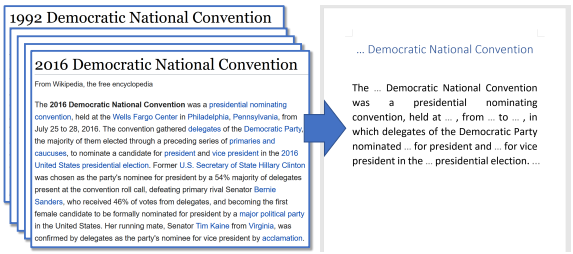

Figure 1: The right side shows a sketch for writing the report of a future democratic national convention, generated from a pile of previous reports.

Can large generative language models be used to assist user writing at the document level while the user still controls the factual content? A possible answer lies in the observation that a substantial portion of day-to-day writing involves some form of reuse. Similar documents (e.g., monthly reports, sales letters, job descriptions) are effectively recycled by changing those segments that need to be modified (Fig. 1).[1] Moreover, documents containing analogous texts are often found collocated in repositories, a common practice for organizations that manage professional documents.[2] The high branching factor that impedes the application of long-form generation might thus be mitigated if models were to exploit conventionally-structured analogous "reusable" texts to produce a "sketch" that captures prototypical document-level text structure. In this view, initial sketches would assist authors by reducing the manual editing effort

[1]Evidenced by the plethora of companies offering reusable business templates for enterprise use, e.g., https://www.businesswire.com/news/home/20201028005573/en/.

[2]For instance, monthly reports of world trade in grains are organized chronologically at https://usda.library.cornell.edu/concern/publications/zs25x844t?locale=en

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2102–2113

(planning, inserting and deleting content, etc.). A good sketch would reflect structural patterns and reusable text, and provide indication, beyond static boilerplate, of locations where user input might be warranted. This could be especially beneficial for novice writers who would otherwise have to read many analogous documents before developing a full picture of what is entailed in writing such documents.[3] A fully-implemented dynamic system might update other portions of the document to reflect modifications introduced by the user.

In this work, we propose a new task, called DOCUMENT SKETCHING, in which initial template-like prototype documents are generated from collections of analogous documents. To support this task, we collected a dataset consisting of approximately 20K Wikipedia documents with similar textual characteristics.[4] For this new task, inspired by previous work in measuring machine translation post-editing productivity (Tatsumi, 2009; Specia and Farzindar, 2010), we define an automatic evaluation metric based on Word Error Rate (Snover et al., 2006; Tomás et al., 2003). We compare against strong baseline models including a mixture of experts model and a reinforcement learning approach designed to handle multi-source inputs and a weak supervision setting. Finally, we provide experimental analysis of these models, using automated and human evaluation studies.

2 Related Work

2.1 Document Generation

Recent work leverages the success of large pre-trained language models to generate long texts such as stories (Rashkin et al., 2020), reviews (Cho et al., 2019a,b) and fake news (Zellers et al., 2019). Most end-user applications for assisting user writing, however, are confined to sentence-level generation (Chen et al., 2019; Kannan et al., 2016; Alikaniotis and Raheja, 2019; Prabhumoye et al., 2019; Faltings et al., 2021). Our work focuses on *document-level* writing assistance in which a document sketch is constructed from a set of similar documents.

[3] Our experiments in Section 6 also show that drafts derived from multiple analogous documents are more effective than those derived from one or two documents.

[4] We release our data and source code for dataset construction and experiments at https://github.com/ellenmellon/document_sketching.

2.2 Template-Based Generation

Some existing work induces templates as an intermediate step for performing tasks like text summarization or response generation. Most use a retrieval-based method to extract similar references from the training corpus as prototypes (Cao et al., 2018; Yang et al., 2019; Wang et al., 2019; Gao et al., 2019; Peng et al., 2019), and learn to separate salient information and latent template structure. Cai et al. (2019) induce an intermediate template for response generation explicitly, but from a single retrieved relevant response. Similar prototype editing work (Guu et al., 2018; Hashimoto et al., 2018; Fabbri et al., 2020) focuses on short text (e.g., a question or a single sentence) or structured output (e.g., code snippet) editing. Oya et al. (2014); Magooda and Litman (2019); Yang et al. (2020); Li et al. (2018) convert each single input text into a template with blanks using rule-based methods. Other work such as (Wiseman et al., 2018) and (Gangadharaiah and Narayanaswamy, 2020) relies on a knowledge base or a domain/task specific ontology to segment text sequences into templates.

2.3 Multi-Sequence Processing

Multiple Sequence Alignment (MSA), widely used in the biological domain (Sauder et al., 2000) to align multiple biological sequences like proteins, has long been leveraged for text pattern matching (Barzilay and Lee, 2003; Alonso et al., 2004). We adopt this method to align input documents and create heuristic templates as weak supervision.

Other tasks taking multiple text sequences as input include multi-document summarization, which seeks to generate an abstractive text summary of multiple input documents (Liu and Lapata, 2019; Chu and Liu, 2019), and multi-source machine translation (Nishimura et al., 2018; Garmash and Monz, 2016) that encodes input texts in multiple source languages and translates them into a target language. Cho et al. (2021) generate a question from input documents by applying a multi-encoder model with a transformer-based coordinator.

3 Problem Definition

We introduce the task of document sketching, which aims to facilitate the authoring process by generating a template-like document draft, based on a collection of sampled similar documents. Formally, the task can be defined as follow: given a set of n documents $X = \{x_1, x_2, ..., x_n\}$, generate

a text sequence s that can be used as the sketch to reduce the human effort involved in composing a target document y.

Evaluation Metrics As in most text generation tasks, we rely on human evaluation (see details in Section 6.3) to draw final conclusions on system comparison. However, due to the high cost of human evaluation, we use automatic evaluation metrics for system development.

It is difficult (and expensive) to collect human-written sketches as references. However, since a generated sketch s is used to help the user complete writing a target document y (i.e., post-editing), we can instead use the target document as a reference and calculate the extra edits required to transform a sketch into that target document. Inspired by the previous uses of word error rate (WER) in evaluating machine translation (Snover et al., 2006; Tomás et al., 2003) and evidence of reasonable correlation between WER and human post-editing productivity (Tatsumi, 2009; Specia and Farzindar, 2010), we propose to assess the effectiveness of s in the completion of y based on WER and the Levenshtein distance (abbreviated as 'lev') between s and y:

$$\begin{aligned} \text{score}(s, y) &= 1 - \text{WER}(s, y) \\ &= 1 - \frac{\text{lev}(s, y)}{|y|} \end{aligned} \tag{1}$$

The higher $\text{score}(s, y)$ is, the fewer minimum number of word-level insertion and deletion edits a user would need to complete writing y if starting from s. To account for the lexical and phrasal variety of writing a target document, we calculate the average score of multiple reference documents Y:

$$\text{score}(s, Y) = \frac{1}{m} \sum_{i=1}^{m} \left(1 - \frac{\text{lev}(s, y_i)}{|y_i|}\right) \tag{2}$$

where $Y = \{y_1, y_2, ..., y_m\}$ and m denotes the number of reference documents.

4 Data

We collect our dataset from the English Wikipedia dump (June 20, 2020). We first group documents into collections with analogous texts, with the observation that articles with shared reusable structural texts also tend to have similar titles. These collections are then split into training, validation and test sets. This dataset is designed for a weakly supervised setting, as gold sketches are unavailable.

% Percent (Category)	Example Collection Titles
90.0 (reasonable)	__ Academy Awards; Super Bowl __
10.0 (arguable):	
5.8 (partial name)	Bob __ (ice hockey)
2.9 (X of a location)	__ of the United States
1.4 (other)	Origin of the __

Table 1: Quality analysis of over 200 randomly selected document collections.

4.1 Wikipedia Document Collections

We observe that Wikipedia article titles provide a strong indication of whether documents are likely to share structural text (i.e., can be put in the same collection) or not. For example, it is reasonable to consider articles with titles like "Super Bowl I", "Super Bowl II" and so on to comprise a document collection of the annual championship game. Therefore, we group documents whose titles are identical but for one token at a specific position. In the "Super Bowl" example, the document titles are the same up to the third token in each title, and the collection can be named "Super Bowl _". It is worth noting that collecting these articles based on their titles is simply reflective of how similar documents in Wikipedia tend to cluster. Our task setup only requires a set of references as the input (e.g., documents organized in the same directory), without the need to have reference documents to share titles.

Title matching can yield noise in the extracted document collections, so we apply simple yet effective restrictions in the extraction pipeline. We empirically set the minimum number of documents per collection to 15 and the minimum document title length to 3 tokens. We randomly select and inspect over 200 extracted collections, and find 90% of them contain analogous documents for a certain topic (examples in Tab. 1). The remaining 10% are less clear, but can be usefully grouped into 3 classes as in Tab. 1. To further reduce noise in each collection, we apply Eq. (2) to calculate the average similarity score of each document with the other documents in a collection, removing those with an average score lower than an empirically chosen threshold -1.5. Finally, we keep document collections with more than 5 documents and truncate them to a maximum of 50 documents, which are divided into train, validation, and test sets in the ratio of 0.8/0.1/0.1. The collection statistics are summarized in Tab. 2.

	Train	Valid	Test	Overall
# Collections	15.7k	2.0k	2.0k	19.7k
# Docs / Collection	21.1	21.3	21.0	21.1
# Tokens / Doc	80.6	79.3	80.3	80.5

Table 2: Document Collection Statistics.

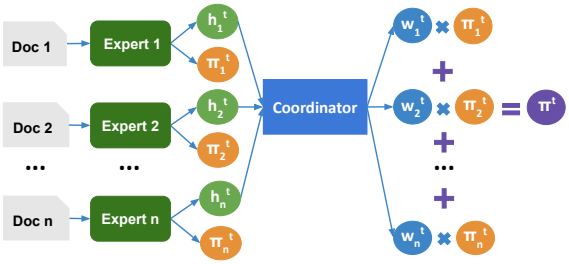

Figure 2: Mixture of Experts (MoE) Framework.

4.2 Task Data Points

We divide each document collection into multiple smaller collections of analogous documents. For standard supervised training, each data point consists of 1 document d to construct a heuristic sketch as weak supervision (see details in Section 5) and up to 9 input documents X. For evaluation, additional 4 documents are used as references Y for calculating the score in Eq. (2). Since document collections whose title tokens differ by a number usually contain documents about events or entities of different times, we order the documents in ascending numerical order to imitate practical scenarios where humans sketch a document by looking at collections of previously written documents. We divide each collection into data points with up to 14 (=1+9+4) documents for all dataset splits, yielding 24k, 3.1k and 3.0k data points (smaller collections) for train, validation, and test sets, respectively.

5 Approach

Since gold document sketches are not available for supervised training, we first perform weakly supervised training by constructing heuristic sketches as targets. Then we apply reinforcement learning strategies for text generation.

5.1 Weakly-Supervised Learning

Heuristic Labels As described in Section 4, each data point has one document d for creating the weak-supervised sketch. We conduct pair-wise sequence alignment (using the Ratcliff-Obershelp algorithm (Ratcliff and Metzener, 1988)) for d and each input document $x_i \in X$, and count the relative alignment ratio (the number of tokens being aligned divided by $|X| = n$) of each token in d. We retain tokens with this ratio above a threshold and replace other tokens with an ellipsis token to form the heuristic target sketch s given X. Consecutive ellipses are merged. The threshold is empirically set at 0.6, which yields the highest average $\text{score}(s, Y)$ on the validation set.

Mixture of Experts (MoE) To generate an initial document sketch, the model needs to take multiple input documents. The most obvious way to do this is to concatenate all input documents into a single long sequence and feed that into an encoder-decoder model like T5 (Raffel et al., 2019). However, processing long sequences in such models is memory-consuming and lack of structure makes it difficult for the model to perform document-level coordination. We therefore use a mixture of experts (MoE) framework, in which a coordinator decodes a token at one timestamp by taking the hidden state and the output vocabulary distribution from each expert that processes a single document. Fig. 2 shows the overall framework of MoE.

The experts have the same encoder-decoder structure and aim to generate s by each encoding a separate single input document only. All experts share the same model parameters. At each decoding timestamp t, the i^{th} expert encodes x_i as well as previously decoded tokens from a coordinator $\tilde{s}_1 ... \tilde{s}_{t-1}$ (or $s_1 ... s_{t-1}$ during training), and outputs a probability distribution π_i^t over all vocabulary words. The coordinator is a transformer-based encoder that takes the hidden state at the current timestamp h_i^t of each i^{th} expert, and outputs a weight w_i^t with a final linear layer. The output weights are used to calculate a weighted sum of the probability distributions from their corresponding experts $\pi_t = \sum_{i=1}^{n} w_i^t \, \pi_i^t$, where π_t is the final distribution for generation at timestamp t.

5.2 Reinforcement Learning

We leverage reinforcement learning (RL) to further improve generation quality. For each training example with input X, we generate a sequence $\hat{s}$, which is sampled from the probability distribution at each time step, $p(\hat{s}_t | \hat{s}_1 ... \hat{s}_{t-1}, X)$. We observe that directly optimizing the evaluation function proposed in Eq. (2) at the sequence level, using a vanilla policy gradient (PG) or a self-critical sequence training (TD-SCST) algorithm (Rennie et al., 2017;

Paulus et al., 2018; Pasunuru and Bansal, 2018), can lead to instability during training as the reward cannot be calculated until the end of generation (Celikyilmaz et al., 2018). Therefore, we instead use a token-level incremental reward that is based on the change to the original reward function $r(\hat{s}, Y) = \text{score}(\hat{s}, Y)$ from each sampled token $\hat{s}_t$, given references Y:

$$r_t(\hat{s}_t, Y) = r(\hat{s}_{1...t}, Y) - r(\hat{s}_{1...t-1}, Y). \quad (3)$$

The training objective can be written as:

$$L_{\text{RL}} = \sum_{t=1}^{T} -r_t(\hat{s}_t, Y) p(\hat{s}_t | \hat{s}_1 ... \hat{s}_{t-1}, X) \quad (4)$$

where $T = |\hat{s}|$. Since optimizing RL loss alone runs the risk of compromising the language model (Paulus et al., 2018; Pasunuru and Bansal, 2017), we use a mixed loss as follows:

$$L_{\text{MIX}} = \lambda L_{\text{RL}} + (1 - \lambda) L_{\text{MLE}} \quad (5)$$

where λ is a hyperparameter to be tuned.

6 Experiments

6.1 Setup

To leverage the recent success in such transformer-based generation models, neural generation models in our experiments are initialized with the base version of T5 (Raffel et al., 2019; Wolf et al., 2020), an encoder-decoder architecture pre-trained on a variety of text-to-text tasks.[5] All hyperparameters are tuned on the validation set. For MoE, we first fine-tune T5-base to obtain a "single expert" model to initialize each individual component model of the MoE. The "single expert" is trained by leveraging a sequence-to-sequence objective as in T5, with each of the n training (input, output) pairs: (x_1, s), (x_2, s), ..., (x_n, s) from a data instance. This is followed by training a complete MoE model as described in Section 5. We use greedy beam search as the decoding strategy for all generative models with a beam size of 4 (the default value in T5-base). We observe that consecutive ellipses and uninformative tokens hurt readability. To improve the readability of output sketches, we apply minor post-processing to all models in our experiments: we merge consecutive ellipses if all tokens between them are punctuation or among the 30 most frequent tokens.[6] Sentence-terminating periods are excepted.

Training and Parameters T5 has about 220 million parameters and MoE has about 310 million parameters in total. The average training time is about 20 hours for T5 and about 30 hours for MoE on a single Tesla V100 node (32GB) with 3 epochs. It takes an additional 10 hours to train a single expert in MoE. For MoE+RL, we use 4 V100 nodes and it takes about 2 days for training. During training, we tune batch size, learning rate and warm-up steps for T5. For MoE and MoE+RL, there is a separate pair of learning rate and warm-up steps for the coordinator. Each batch size is tuned within a range of [1, 32] (with no more than 3 values for each model); each warm-up step is tuned at $\{4000, 8000, 16000\}$; each learning rate is tuned in $\{1e^{-5}, 3e^{-5}, 1e^{-4}\}$. For RL training, we observe performance gain only when λ is relatively large, so we tune $\lambda \in \{0.95, 0.97, 0.99\}$ (similar to Celikyilmaz et al. (2018)). In order to avoid the situation where the models learn to not generate ellipses that lead to unreadability, we set the cost for deleting an ellipsis to be much smaller (0.1) in the reward function. In addition, we randomly select three seed values for each model. Each model is trained with the above parameter value combinations. We apply grid parameter search for T5 and MoE, and random search for MoE+RL. Hyperparameters are tuned based on automatic evaluation score (1-WER) on the validation set.

The best hyperparameter configurations for best-performing models were: **T5**: batch size = 2, warm-up step = 4000, learning rate = $3e^{-5}$; **MoE**: batch size = 15, warm-up step = 8000, learning rate = $3e^{-5}$, warm-up step (coordinator) = 4000, learning rate (coordinator) = $1e^{-4}$; **MoE+RL**: batch size = 4, warm-up step = 4000, learning rate = $3e^{-5}$, warm-up step (coordinator) = 4000, learning rate (coordinator) = $3e^{-5}$, $\lambda = 0.99$.

6.2 Systems

Non-neural systems include the following approaches: **Last-pair**: The output is the aligned boilerplate text between the last pair of ordered input documents (as described in Section 4, in cases where the differing title token is time-related, the last pair refers to the most recent two docu-

[5] Note that T5 can be easily replaced by Other pre-trained generative models like BART (Lewis et al., 2020) in our model framework.

[6] https://github.com/first20hours/google-10000-english

ments); **Last-retrieval**: The last document is retrieved as the initial draft; **MSA**: We apply the multi-sequence alignment approach (Barzilay and Lee, 2003), to align tokens in input documents and replace tokens that have too few alignments (threshold tuned on valid set) with ellipses; **Consensus-MSA**: This resembles how we generate heuristic drafts as weak supervision in Section 5.1. However, instead of using the held-out document d to create the skeleton, we use the input document x_i that gives the highest value of $\sum_{j=1}^{n} \text{score}(x_i, x_j)$.

We evaluate the following neural approaches: **T5 (zero-shot)**: We directly apply T5 without fine-tuning. We use T5 to encode each input document individually and at each decoding timestamp, we combine probability distributions from all T5 decoders with average pooling. As T5 rarely generates ellipses, we insert an ellipsis token if the probability of the top candidate token w from the combined distribution is below a threshold α that is tuned on the validation set (i.e., $p(w) < \alpha$).[7] We used the "summarize :" prefix in the zero-shot setting, this being is the most similar T5 pre-trained tasks to ours; **T5 (doc-finetune)**: We finetune T5 with all input documents concatenated into a single string input (with a special separator token) and each target document (instead of the heuristic sketch) as the generation target; **T5**: Similar to the *T5 (doc-finetune)* setting, but with the heuristic sketch defined in Section 5 as the generation target; **MoE**: This is the mixture of experts model described in Section 5, also trained with the heuristic sketches as the weak supervision; **MoE + RL**: This is the RL model described in Section 5, with the trained MoE as the warm-start.

6.3 Results

Automatic Evaluation We use the automatic evaluation of Section 3 for system development while relying on human evaluation to draw final conclusions on system comparison. However, we observe reasonable consistency between our automatic and human evaluation results. We report automatic evaluation results in Tab. 3. The first numerical column includes overall results of the test set, and the remaining columns categorize the results into three roughly equi-sized levels of similarity among documents within the input document set, estimated on the basis of average pair-wise edit distance normalized by length.

[7] Since the original T5 was not trained with ellipsis as a special token, each T5 decoder uses its own greedily decoded token if an ellipsis is inserted ($p(w) < \alpha$).

System	All	Input document similarity High	Med.	Low
last-retrieval	−0.242	0.338	−0.284	−0.733
last-pair	0.147	0.410	0.063	−0.021
MSA	0.149	0.374	0.072	0.012
consensus-MSA	0.202	**0.438**	0.132	0.047
T5 (zero-shot)	0.000	0.000	0.000	0.000
T5 (doc-finetune)	0.050	0.382	−0.014	−0.197
T5	0.182	0.389	0.124	0.044
MoE	0.205	0.411	0.147	0.069
MoE+RL	**0.213**	0.414	**0.158**	**0.077**

Table 3: Automatic evaluation scores (1−WER) of overall test examples and scores categorized by input document similarity levels.

System A	System B	Prefer A	Prefer B
MoE	consensus-MSA	**50.70%**	39.28%
MoE	T5	**46.15%**	42.88%
MoE	MoE+RL	**49.93%**	39.65%

Table 4: Human evaluation scores. A number in bold indicates that the system is significantly better at $p < 0.03$, computed using 10k bootstrap replications.

Among non-neural models, we observe that multi-sequence alignment approaches outperform last-retrieval and last-pair where only one or two input documents are leveraged to produce the initial draft. We also notice that consensus-MSA with a "central" document as the skeleton is more effective than the standard MSA that treats all inputs equally. When directly applying zero-shot T5, we notice that when the threshold α described in Section 6.2 equals 1.0 (i.e., the output is always an empty string), the model can achieve the best score 0. This is partly because that none of the T5 pre-trained tasks is the same as ours. In addition, simply exploiting the generation probability at each decoding timestamp can be problematic. For example, when decoding a frequently-appearing entity with multiple tokens, the first token might have low probability given the preceding context while subsequent tokens can be more probable simply because they frequently appear with the first token. T5 with supervision from the target document (doc-finetune) yields a much lower score than T5 finetuned with the heuristic draft. This is mostly because the complete target document contains much more information than can be inferred from the input; the model learns to hallucinate facts, necessitating heavy deletion by users.

MoE outperforms T5, which indicates that hav-

Collection Title Target Document	__ Waterford Senior Hurling Championship The '1960 WSHC' was the 60th staging of the WSHC since its establishment by the Waterford County Board in 1897. On 9 October 1960, Mount Sion won the championship after a 5-09 to 2-05 defeat of Erin's Own in the final. This was their 16th championship title overall and their eighth title in succession.
consensus-MSA	The '... WSHC' was the ... staging of the WSHC since its establishment by the Waterford County Board in 1897. On ..., Mount Sion won the championship after a ... defeat of ... in the final. This was their ... championship title overall and their ... title in succession.
T5	The '... WSHC' was the 58th staging of the WSHC since its establishment by the Waterford County Board in 1897. On ... September ... won the championship after a ...-10 to 1-... defeat of ... in the final. This was their ... championship title overall and their ... title in succession.
MoE	The '1957 WSHC' was the 57th staging of the WSHC since its establishment by the Waterford County Board in 1897. On 15 September 1957, Mount Sion won the championship after a 2-10 to 1-08 defeat of ... in the final. This was their 13th championship title overall and their fifth title in succession.
MoE+RL	The '... WSHC' was the ... staging of the the WSHC since its establishment by the ... County Board in 1897. On ... 1957, ... won the championship after a ...-10 to ... defeat of ... in the final. This was their ... championship title overall and their ... title in succession.
Collection Title Target Document	Botanischer Garten der Universität __ The 'Botanical Garden in Potsdam' (or Botanischer Garten der Universität Potsdam), is a botanical garden and arboretum maintained by the University of Potsdam. It has a total area of 8.5 hectares, of which 5 hectares are open to the public, and is located immediately southwest of the Orangery Palace at Maulbeerallee 2, Potsdam, in the German state of Brandenburg. It is open daily; an admission fee is charged for the glasshouses only (2017). The garden was established in 1950 on two adjacent plots of land: part of the Sanssouci Park, and the Paradise Garden (about 2.5 hectares).
consensus-MSA	... a botanical garden ... maintained by the University of ... It is located ... open ... The garden was ... contains ...
T5	The 'Botanischer Garten der Universität ... is a botanical garden maintained by the University of ... It is located at ... Baden - Württemberg, Germany, and open ... The garden was established in ... and contains about ... species ...
MoE & MoE+RL	The 'Botanischer Garten der Universität ... is a botanical garden maintained by the University of ... It is located at ... The garden was established in ...

Table 5: Sample outputs: in the first set of outputs there is high similarity among documents in the input document set while in the second inter-document similarity is low. Generated tokens are in blue if they are consistent with the target document, and in red if inconsistent.

ing a coordinator to communicate between each individual encoder and decoder effectively improves model performance. We also experiment with coordinators using a simple linear layer and find that transformer-based ones are much more effective. Applying RL on top of a trained MoE model helps further improve the model performance in automatic evaluation. We denote the RL approach as MoE+RL in Tab. 3. Warm-starting RL with a MoE model close to fully-converged gives slightly better results than with an fully-converged MoE model.

The rightmost three columns in Tab. 3 divide test examples into 3 roughly equal-sized levels of similarity among documents within the input document set ('high', 'medium', 'low'). Both last-retrieval and T5 (doc-finetune) models drop dramatically as the input similarity decreases, mostly because they contain a lot of hallucinated content, an issue that is even more severe when there is little overlapping structure or factual content among the input documents. MoE based models are much more robust to low input similarity compared to the baselines. For the group with the most similar input documents, consensus-MSA gives the best score, while MoE based models yield much better performance in the other two groups.

Human Evaluation In order to avoid bias in human judgments, we control possible confounding factors including sequence length and the number of ellipses in a sequence during decoding (Nakov et al., 2012; Guzmán et al., 2015). We apply a tuneable penalty for each confounding factor (at inference time only) for each neural model (Murray and Chiang, 2018) to generate examples for human evaluation, such that the average output length and number of ellipses in each sequence from all neural systems compared in Tab. 4 are almost the

same.[8] We notice that such normalization does not affect the automatic evaluation score of each system much and the system ranks do not change.

Human evaluation was conducted using crowd-sourced workers. Judges were presented with paired randomized outputs and target documents, and were instructed to choose their preference for a starting point for writing the target document in order to save editing time.[9] Judgments were based on a five-point Likert scale, and ties were permitted. Four judges evaluated each pair, and metrics were imposed to block poorly performing judges. Sample sizes are 1000 for all system pair comparisons.

Results in Tab. 4 confirm that MoE outperforms consensus-MSA and T5. Although RL helps improve automatic evaluation scores on top of MoE, human judges mostly prefer MoE over MoE+RL. The fact that applying RL can hurt readability (Paulus et al., 2018; Pasunuru and Bansal, 2018) may explain why MoE+RL achieves higher automatic scores yet worse human scores than MoE. We observe that MoE+RL has occasional difficulty predicting where ellipsis tokens should be, which hurts sketch readability. Moreover, since our task is a subjective one (e.g., different preferences for more/less verbose sketches), a customized RL reward (e.g., different cost for deletion and insertion for WER calculation) should be applied in real applications to reflect different user preferences.

6.4 Analysis

We investigated the differences in model performance when the input documents have different similarity scores. When the similarity is lower, models tend to produce drafts that are harder to interpret because they generate a higher percentage of generic or ellipsis tokens, although MoE and MoE+RL are less vulnerable to this. Tab. 7 shows the average percentage of generated ellipsis or punctuation tokens for each system when the input document similarity is high or low. We can see that when the similarity score is higher, the two statistical numbers do not differ greatly between systems, except that MSA has a much higher percentage of ellipses. When similarity is low, MoE and MoE+RL have much lower percentages of ellipses or punctuation, compared with other models.

Tab. 5 shows two examples of system outputs. These examples are from the clusters of high and low input similarity respectively. When the input similarity is high, content tokens from consensus-MSA are included in the target document while the neural models tend to generate more hallucinated tokens. This explains the better score consensus-MSA achieves in the higher input similarity cluster. We also notice that when the similarity is high, consensus-MSA generally has better coverage of overlapped content between input documents, and its generated drafts are on average more than 10% longer than other systems. When the similarity is low, the second example shows that all systems generate shorter outputs due to less shared structure between input documents. In this case, consensus-MSA starts to include more content that does not necessarily appear in the target and MoE-based models generate more reasonable sketches and contain fewer ellipses and uninformative tokens, which can hurt the sketch readability.

7 Discussion

Since gold document sketches are difficult to annotate, there remain challenges of how to leverage or create better weak supervision. We may also want models to adjust to users' personalized preferences over document sketches (e.g., some may prefer more deletions than insertions or vice versa) in practical application deployment, where RL can play a major role by having models directly optimized by customized rewards.

We show that MoE-based models tend to generate more readable sketches by outputting fewer ellipses and functional tokens, though in some cases it can be difficult to guess what should fill a given ellipsis. An interesting future study could examine how the placement and number of ellipses in a sketch affects readability. We would also like to explore whether replacing ellipsis tokens with some more contentful label (e.g., entity type) or a retrieved sample content for a gap (Xu et al., 2021) could help make sketches more interpretable.

An initial document sketch provides the reusable text structure that allows a user to fill in detailed content. Interactive document generation can be seen as the next step of our document sketching task that dynamically fills in more local contexts based on users' inputs. We suggest that RL could again

[8] Since MoE has a more balanced average sequence length and the number of ellipses, we keep MoE unchanged and adjust T5 and MoE+RL to have the same values as MoE.

[9] Ideally, it would be preferable to directly measure editing productivity through usability testing, but this is impractical in a crowd worker setting, and dependent on confounding factors such as design of the user interface.

Collection Title	1960 United States Presidential Election in __
Target Doc Title	1960 United States Presidential Election in Colorado
w/o user input	**The '1960 United States presidential election in ...' took place on November 8, 1960, as part of the 1960 ... election.** ... chose ... representatives, or electors to the Electoral College, who voted for president and vice president.
w/ user input	**The '1960 United States presidential election in Colorado' took place on November 8, 1960, as part of the 1960 United States presidential election.** Voters chose three representatives, or electors to the Electoral College, who voted for president and vice president. Colorado was won by incumbent Vice President Richard Nixon (R-California), with 50.9% of the popular vote, against Senator John F. Kennedy (D-Massachusetts) with 49.1%.

Table 6: Example of interactive writing with user input. The first sentence (**bold**) is generated in the "w/o user input" setting, while it is the gold first sentence of the target document in "w/ user input" setting. Generated tokens are in blue if consistent with the target document and red if inconsistent.

System	Ellipses		Punctuation	
	High	Low	High	Low
MSA	15.9%	42.0%	10.9%	19.0%
consensus-MSA	10.6%	29.8%	10.8%	19.2%
T5	10.0%	23.2%	10.9%	14.6%
MoE	8.6%	16.6%	10.2%	13.0%
MoE+RL	7.3%	15.4%	10.2%	14.0%

Table 7: Percentage of ellipses (in all generated tokens) and punctuation (in generated tokens excluding ellipses) tokens. 'High' and 'low' refer to the average pair-wise input document similarity level.

play a major role towards building such writing assistants, with rewards from a real user or a user simulator. We notice, for example, in Tab. 6 that given user input in the first sentence, the subsequent generated sentences become much more specific and relevant to the new input.

8 Conclusions

We have presented a new task, DOCUMENT SKETCHING, designed to generate draft documents that users can edit. To support this task, we introduced a new dataset and a weakly supervised learning setting. Experimental results show that deep learning models outperform established multi-sequence alignment approaches.

Acknowledgments

We thank members of Microsoft Research and University of Washington's NLP groups who provided feedback and insights to this work. In particular, we thank Chenyan Xiong for providing insights during early project discussions. We also thank Michael Lee, Sara Ng, Sitong Zhou, and Trang Tran for their assistance and feedback on the human evaluation setup. We also thank Zhang Li, Kosh Narayanan, Chandra Chikkareddy, Si-Qing Chen, and Weixin Cai of Microsoft's Natural Language Experience team for their insights into authoring experience.

Ethical Considerations

We believe this paper to be part of body of work that can mitigate some of the ethical pitfalls (Bender et al., 2021) of large pre-trained models such as GPT-3 (Raffel et al., 2019). While the present work does not entirely prevent misuse, e.g., by a malicious user trying to promote misinformation, Automatic Document Sketching can be more broadly framed as a form of *controllable* generation, which places greater control into the hands of the users. As such it is complementary to ongoing research on fake news detection (Zellers et al., 2019), toxicity detection (Han and Tsvetkov, 2020; Pavlopoulos et al., 2020), and consistency detection in, e.g., summarization (Maynez et al., 2020; Wang et al., 2020). Human subjects (crowdworkers) were paid at a rate higher the legal minimum wage in our locale (Washington State, U.S.A.).

References

Dimitris Alikaniotis and Vipul Raheja. 2019. The unreasonable effectiveness of transformer language models in grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 127–133, Florence, Italy. Association for Computational Linguistics.

Laura Alonso, Irene Castellón, Jordi Escribano, Xavier Messeguer, and Lluís Padró. 2004. Multiple sequence alignment for characterizing the lineal structure of revision. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using

multiple-sequence alignment. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 16–23.

Emily Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proc. of ACM FAccT*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, and Prafulla Dhariwal et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Deng Cai, Yan Wang, Wei Bi, Zhaopeng Tu, Xiaojiang Liu, and Shuming Shi. 2019. Retrieval-guided dialogue response generation via a matching-to-generation framework. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1866–1875, Hong Kong, China. Association for Computational Linguistics.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–161, Melbourne, Australia. Association for Computational Linguistics.

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.

Mia Xu Chen, Benjamin N Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M. Dai, Zhifeng Chen, Timothy Sohn, and Yonghui Wu. 2019. Gmail smart compose: Real-time assisted writing. In *KDD*.

Woon Sang Cho, Pengchuan Zhang, Yizhe Zhang, Xiujun Li, Michel Galley, Chris Brockett, Mengdi Wang, and Jianfeng Gao. 2019a. Towards coherent and cohesive long-form text generation. In *Proceedings of the First Workshop on Narrative Understanding*, pages 1–11, Minneapolis, Minnesota. Association for Computational Linguistics.

Woon Sang Cho, Pengchuan Zhang, Yizhe Zhang, Xiujun Li, Michel Galley, Chris Brockett, Mengdi Wang, and Jianfeng Gao. 2019b. Towards coherent and cohesive long-form text generation. In *Proceedings of the First Workshop on Narrative Understanding*, pages 1–11, Minneapolis, Minnesota. Association for Computational Linguistics.

Woon Sang Cho, Yizhe Zhang, Sudha Rao, Asli Celikyilmaz, Chenyan Xiong, Jianfeng Gao, Mengdi Wang, and Bill Dolan. 2021. Contrastive multi-document question generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 12–30, Online. Association for Computational Linguistics.

Eric Chu and Peter Liu. 2019. MeanSum: A neural model for unsupervised multi-document abstractive summarization. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1223–1232. PMLR.

Alexander Fabbri, Patrick Ng, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2020. Template-based question generation from retrieved sentences for improved unsupervised question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4508–4513, Online. Association for Computational Linguistics.

Felix Faltings, Michel Galley, Gerold Hintz, Chris Brockett, Chris Quirk, Jianfeng Gao, and Bill Dolan. 2021. Text editing by command. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5259–5274, Online. Association for Computational Linguistics.

Rashmi Gangadharaiah and Balakrishnan Narayanaswamy. 2020. Recursive template-based frame generation for task oriented dialog. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2059–2064, Online. Association for Computational Linguistics.

Shen Gao, Xiuying Chen, Piji Li, Zhangming Chan, Dongyan Zhao, and Rui Yan. 2019. How to write summaries with patterns? learning towards abstractive summarization through prototype editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3741–3751, Hong Kong, China. Association for Computational Linguistics.

Ekaterina Garmash and Christof Monz. 2016. Ensemble learning for multi-source neural machine translation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1409–1418, Osaka, Japan. The COLING 2016 Organizing Committee.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

Francisco Guzmán, Preslav Nakov, and Stephan Vogel. 2015. Analyzing optimization for statistical

machine translation: MERT learns verbosity, PRO learns length. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 62–72, Beijing, China. Association for Computational Linguistics.

Xiaochuang Han and Yulia Tsvetkov. 2020. Fortifying toxic speech detectors against veiled toxicity. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7732–7739, Online. Association for Computational Linguistics.

Tatsunori B. Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S. Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *NeurIPS*, pages 10073–10083.

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, and Vivek Ramavajjala. 2016. Smart Reply: Automated response suggestion for email. In *KDD*.

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.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. 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 1865–1874, New Orleans, Louisiana. Association for Computational Linguistics.

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.

Ahmed Magooda and Diane Litman. 2019. Abstractive summarization for low resource data using domain transfer and data synthesis. In *AAAI*.

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.

Kenton Murray and David Chiang. 2018. Correcting length bias in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 212–223, Brussels, Belgium. Association for Computational Linguistics.

Preslav Nakov, Francisco Guzman, and Stephan Vogel. 2012. Optimizing for sentence-level BLEU+1 yields short translations. In *Proceedings of COLING 2012*, pages 1979–1994, Mumbai, India. The COLING 2012 Organizing Committee.

Yuta Nishimura, Katsuhito Sudoh, Graham Neubig, and Satoshi Nakamura. 2018. Multi-source neural machine translation with missing data. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 92–99, Melbourne, Australia. Association for Computational Linguistics.

Tatsuro Oya, Yashar Mehdad, Giuseppe Carenini, and Raymond Ng. 2014. A template-based abstractive meeting summarization: Leveraging summary and source text relationships. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pages 45–53, Philadelphia, Pennsylvania, U.S.A. Association for Computational Linguistics.

Ramakanth Pasunuru and Mohit Bansal. 2017. Reinforced video captioning with entailment rewards. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 979–985, Copenhagen, Denmark. Association for Computational Linguistics.

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.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

John Pavlopoulos, Jeffrey Sorensen, Lucas Dixon, Nithum Thain, and Ion Androutsopoulos. 2020. Toxicity detection: Does context really matter? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4296–4305, Online. Association for Computational Linguistics.

Hao Peng, Ankur Parikh, Manaal Faruqui, Bhuwan Dhingra, and Dipanjan Das. 2019. Text generation with exemplar-based adaptive decoding. 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 2555–2565, Minneapolis, Minnesota. Association for Computational Linguistics.

Shrimai Prabhumoye, Chris Quirk, and Michel Galley. 2019. Towards content transfer through grounded 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 2622–2632, Minneapolis, Minnesota. Association for Computational Linguistics.

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. *arXiv*, abs/1910.10683.

Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. 2020. PlotMachines: Outline-conditioned generation with dynamic plot state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4274–4295, Online. Association for Computational Linguistics.

John W. Ratcliff and David Metzener. 1988. Pattern matching: The gestalt approach. *Dr. Dobb's Journal*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1179–1195.

J. Michael Sauder, Jonathan W. Arthur, and Roland L. Dunbrack Jr. 2000. Large-scale comparison of protein sequence alignment algorithms with structure alignments. *Proteins: Structure, Function and Genetics*, 40(1):6–22.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *AMTA*.

Lucia Specia and Atefeh Farzindar. 2010. Estimating machine translation post-editing effort with hter. In *Proceedings of the Second Joint EM+/CNGL Workshop Bringing MT to the User: Research on Integrating MT in the Translation Industry (JEC 10)*, pages 33–41.

Midori Tatsumi. 2009. Correlation between automatic evaluation metric scores, post-editing speed, and some other factors. In *MT Summit XII: proceedings of the twelfth Machine Translation Summit*.

Jesús Tomás, Josep Àngel Mas, and Francisco Casacuberta. 2003. A quantitative method for machine translation evaluation. In *Proceedings of the EACL 2003 Workshop on Evaluation Initiatives in Natural Language Processing: are evaluation methods, metrics and resources reusable?*, pages 27–34, Columbus, Ohio. Association for Computational Linguistics.

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.

Kai Wang, Xiaojun Quan, and Rui Wang. 2019. BiSET: Bi-directional selective encoding with template for abstractive summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2153–2162, Florence, Italy. Association for Computational Linguistics.

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.

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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, and Anthony et al. Moi. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Frank F Xu, Bogdan Vasilescu, and Graham Neubig. 2021. In-IDE code generation from natural language: Promise and challenges. *arXiv preprint arXiv:2101.11149*.

Jian Yang, Shuming Ma, Dongdong Zhang, Zhoujun Li, and Ming Zhou. 2020. Improving neural machine translation with soft template prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5979–5989, Online. Association for Computational Linguistics.

Ze Yang, Wei Wu, Jian Yang, Can Xu, and Zhoujun Li. 2019. Low-resource response generation with template prior. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1886–1897, Hong Kong, China. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.