Text Editing by Command

Felix Faltings[♢,*] **Michel Galley**[♠] **Gerold Hintz**[♠] **Chris Brockett**[♠]
Chris Quirk[♠] **Jianfeng Gao**[♠] **Bill Dolan**[♠]
[♢]Department of Computer Science, ETH Zürich [♠]Microsoft Research
fafelix@student.ethz.ch
{mgalley,gehint,chrisbkt,chrisq,jfgao,billdol}@microsoft.com

Abstract

A prevailing paradigm in neural text generation is one-shot generation, where text is produced in a single step. The one-shot setting is inadequate, however, when the constraints the user wishes to impose on the generated text are dynamic, especially when authoring longer documents. We address this limitation with an interactive text generation setting in which the user interacts with the system by issuing commands to edit existing text. To this end, we propose a novel text editing task, and introduce WikiDocEdits, a dataset of single-sentence edits extracted from Wikipedia revision histories. We show that our Interactive Editor, a transformer-based model trained on this dataset, outperforms baselines and obtains positive results in both automatic and human evaluations. We present empirical and qualitative analyses of this model's performance.[1]

1 Introduction

A long-standing goal of natural language processing research has been to generate long-form text (Lebowitz, 1985; Fan et al., 2018; Rashkin et al., 2020). Recent large generative language models such as GPT-2 (Radford et al., 2019), and GPT-3 (Brown et al., 2020), demonstrate an impressive ability to generate fluent text, but their outputs are difficult to control beyond a prompt, and they manifest a tendency to hallucinate facts (Wiseman et al., 2017). Much recent work has thus focused on making such models more controllable (Keskar et al., 2019; Hu et al., 2017; Zhang et al., 2020; Dathathri et al., 2019), and factually grounded (Guu et al., 2020; Liu et al., 2018b).

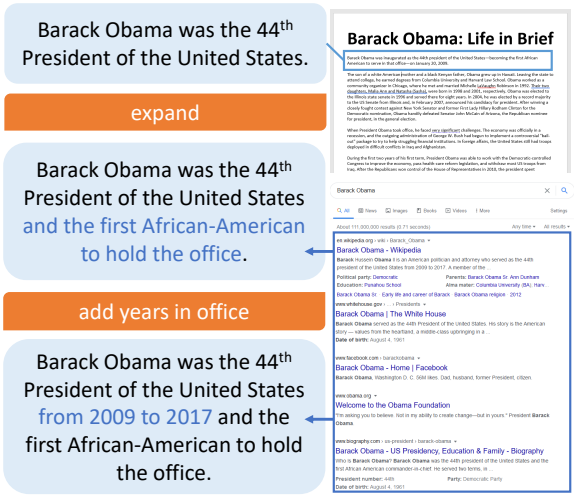

Figure 1: An illustration of our interactive text generation setting. This is an example generated by our model. The blue panels represent the text being edited, taken from the document shown on the right. The orange panels represent user edit commands. The model grounds edits in query results from a commercial search engine.

Most such work only considers a *one-shot* generation setting. Given a set of inputs, which may be a prompt, a control code (Keskar et al., 2019), or a table of data (Liu et al., 2018b) for example, the system generates text in a single step. Humans, though, often produce text through an evolutionary process involving multiple draft-edit cycles. This is not simply because they make mistakes when writing, but because they may require multiple iterations to help them shape and even make sense of what they want to express (Pirolli and Card, 2005). For example, consider a user writing an article about Barack Obama. They might start with a simple sentence such as "Barack Obama was the 44th President of the United States". Next, they may wish to expand on that sentence, adding information, or rephrasing it to integrate it better with

* Work done at Microsoft Research.

[1]All our code (including code to recreate our data) and pre-trained models will be made available at: http://microsoft.com/research/project/interactive-document-generation

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5259–5274
June 6–11, 2021.

the text. Replicating this process in software will mean allowing users to adjust their requirements in response to model outputs. Even an error-free system that meets all of a user's initial requirements does not obviate the need for iteration, since those constraints are themselves dynamic. While this work focuses on text, we also note that these arguments extend to other settings where a system must generate a complex, structured object for a user, such as image or code generation.

The purpose of this paper is to bring into view the task of controllable text editing, as a step beyond *one-shot* generation towards *interactive* document generation. A full interactive document generation system will likely comprise multiple components, possibly including one-shot generation to create a first draft. Editing is crucial to interactivity because it allows users to change previously generated text to fit their dynamic constraints. This is a stateful operation, where the state is the current version of the document, as opposed to stateless recasting of text from scratch using a one-shot model. While services like Grammarly or MS Word already offer rewriting suggestions, they mainly focus on syntactic or stylistic edits such as paraphrases (Gupta et al., 2018). In this work, we are interested in a broader range of edits, particularly those that add or remove content, or change the meaning of text. Figure 1 illustrates this editing setting with an example from our trained model, where a user produces a sentence about Barack Obama over multiple edits.

In sum, we make the following contributions: We introduce a challenging new text editing task, wherein a model must learn to edit text in response to a user command, while drawing on grounding to avoid problems of hallucination (Wiseman et al., 2017). To accompany this task, we release an open-source dataset of sentence-level edits extracted from Wikipedia, including editor comments, which we leverage as natural language commands, together with pre-retrieved grounding documents. We show that a transformer-based editing model trained on our data outperforms "parrot" and GPT-2 baselines, and obtains competitive results compared to gold-standard edits in human evaluations. We then perform an empirical analysis of our model's performance, showing the importance of the command and grounding, and the varying difficulty of edits in our dataset.

2 Text Editing Task

We now formalize our text editing task. Let D be a document, q a user command[2], and $\mathcal{G}$ some appropriate form of grounding. Moreover, let D' be an edited version of D. Then our task is, given a dataset of edits $\mathcal{D} = \{(D_0, q_0, \mathcal{G}_0, D'_0), ..., (D_N, q_N, \mathcal{G}_N, D'_N)\}$, learn to produce document D', given D, q, and $\mathcal{G}$.

Note that while previous work on text editing usually only considers D as input, we include both a form of control q and grounding $\mathcal{G}$. The command is needed because otherwise the type of edit to be made is undefined, while the grounding provides external knowledge needed to make an edit.

In our specific instance of this task, we will only consider sentence-level edits. More formally, we consider edits $D \rightarrow D'$, where D and D' differ only on a single sentence $s \in D$, respectively $s' \in D'$. While, in general, edits can vary in complexity from document-level to character-level changes, sentences are a natural way to break down text into relatively independent units of meaning, so it makes sense to edit text one sentence at a time. More complex, document-level edits can be seen as a composition of multiple sentence-level edits.

Additionally, we will consider user commands q written in natural language, e.g., "add years in office". The command could also take other forms, such as a categorical variable, but natural language allows for the greatest flexibility in specifying what the edit should accomplish. Moreover, natural language commands are a good fit for our model, which we will initialize with pre-trained language model weights. For similar reasons, we will also consider corpora of text snippets as our grounding $\mathcal{G}$. Alternatively, the grounding could also consist of structured data such as tables or graphs. In a real user scenario, this grounding might be supplied by the user, or retrieved on the fly. For our dataset, we pre-retrieve groundings by querying a commercial search engine.

3 Data

To accompany our text editing task we present a novel dataset of nearly 12 million sentence-level edits, WikiDocEdits. These edits were extracted from the revision histories in the February 1, 2020

[2]This notation reflects that the edit command is analogous to a query in a retrieval or QA setting in that it expresses a form of user intent.

dump of English Wikipedia.[3]

For a given Wikipedia page, a revision consists of a source and target text, corresponding to the old and new versions of the page. Each revision is also accompanied by an editor comment, which we will use as a proxy for the user command. For a given revision, we split the source and target texts into sentences and then attempt to match the sentences between source and target. For efficiency, we only look at a k-sentence neighborhood. Unmatched sentences are candidates for edits. A source sentence s and target sentence t form an edit pair $s \rightarrow t$ if $f(s,t) > \epsilon$, where f is sentence-level BLEU[4] without smoothing and $\epsilon = 0.1$ in our case. If an unmatched source sentence does not form an edit pair with any target sentence, we consider it to be a sentence deletion. This can also be thought of as matching to an empty sentence. We identify sentence insertions in an analogous manner. Importantly, we only consider revisions that contain a single sentence-level edit. Otherwise, the editor comment that accompanies each revision may only describe one of the possibly many sentence-level edits. See appendix A for a detailed description of our processing pipeline.

3.1 Grounding

We retrieve grounding snippets for the edits in our dataset by querying a commercial search engine. In order to formulate a query for a given edit, we combine the relevant page and section titles with keywords[5] from the target sentence. While the target sentence is not available at test time, we make the assumption that in a real user scenario the relevant grounding would be provided by the user.

We retrieve the top 200 returned web page results and only keep the preview snippets returned by the search engine as the grounding corpus.[6]

Because Wikipedia, as well as several clones, often appear in search engine results, we check for 4-gram overlap between the target sentence and each grounding snippet, removing any snippet with more than 50% overlap. Finally, we rerank[7] the retrieved snippets using an information extraction score, and merge the ranked snippets to take the first $N = 512$ tokens.

[3]Downloadable from https://dumps.wikimedia.org/.

[4]We use BLEU-4 in all experiments of this paper.

[5]See appendix B for how we identify keywords.

[6]We also experimented with retrieving and parsing the HTML pages from the search but this did not lead to better end-to-end performance than just using the snippets.

[7]See appendix C for details on reranking.

Statistic	Percentiles			Mean
	25%	50%	75%	
Sentence length	16	23	31	25.25
Diff length	2	3	9	7.27
Comment length	2	3	7	5.20

Table 1: Summary statistics of WikiDocEdits. All statistics were computed on a 1% subsample of the data. Lengths reported in number of words. The diff length corresponds to the number of words, inserted or deleted, affected by a given edit.

3.2 Data Analysis

We now provide an overview of our dataset. From 667 dump files in the February 1st 2020 dump of Wikipedia, we extract 11,850,786 edits, and take a 1% sample of 118,818 edits to run our analyses. Table 1 presents summary statistics for our data, and in the following, we break down the edits by edit type, and present some examples. See also appendix D for an analysis of the quality of the retrieved grounding.

Fluency and Content Edits We are interested in the distribution of different edit types within our dataset. In particular, we want to distinguish between fluency edits, which only affect the grammar or structure of a sentence, and content edits, which change the meaning of a sentence. We can lean on previous work to categorize edits on Wikipedia. Yang et al. (2017) create 13 edit intention categories, and train a classifier to label revisions according to the categories. We apply their classifier to our data, and group their 13 categories into "fluency", "content", or "other" edits, as reported in table 2. With the caveat that the edits were labelled automatically using a trained classifier, we see that, while fluency edits make up the majority of the edits in our data, a large proportion are content edits.

Examples Table 3 presents some examples from our data. These were chosen to illustrate a variety of edits. The first example shows an elaboration edit, appending new information to the end of a sentence. The second example is a simple typo fix, while the third is changing a fact. Finally, the last example is a more complex edit to reword a sentence. We can see that there is a large variety of edits in our dataset. See table 11 in the appendix for more examples.

Group	Labels	%
Fluency	Refactoring, Copy-editing, Wikification, Point-of-view	57.00
Content	Fact-update, Simplification, Elaboration, Verifiability, Clarification	24.77
Other	Unlabeled, Counter-vandalism, Vandalism, Process, Disambiguation	26.65

Table 2: Breakdown of edits by grouped intention labels. See Table 10 in the appendix for a breakdown by intention label instead of group. The percentages do not total 100 because edits can have multiple labels.

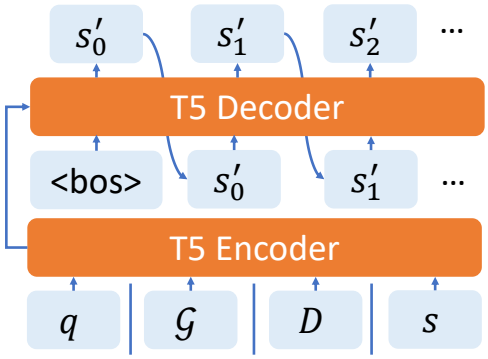

Figure 2: An illustration of our model. The inputs to the encoder are sequences of tokens separated by ⟨ sep⟩ tokens, represented by the vertical bars in the figure.

4 Model

We formalize our model, which we refer to as Interactive Editor, as a standard auto-regressive sequence to sequence model. Because our data only contains single-sentence edits, we assume that the sentence to be edited in the source document is given as an input to the model.

Given a source sentence $s \in D$, the context around s, which we will refer to as D by abuse of notation, a user command q, a grounding corpus $\mathcal{G}$, and a candidate target sentence s', the model, f, computes

$$f(s, s', D, q, \mathcal{G}) = P(s'|s, D, q, \mathcal{G}) = \prod_i P(s'_i|s'_{<i}, s, D, q, \mathcal{G}),$$

where $s'_{<i} = \{s'_0, ..., s'_{i-1}\}$ are the tokens preceding s'_i in s'.

We use the same encoder-decoder architecture as T5 (Raffel et al., 2020) and initialize our model with pretrained language model weights. The encoder-decoder architecture allows us to perform full attention over the inputs s, D, q, and $\mathcal{G}$, while the decoder allows us to auto-regressively generate s'. Meanwhile, initializing with pretrained weights has been shown to achieve state-of-the-art results on many NLP tasks (Raffel et al., 2020).

In order to adapt T5 for our task, we represent all our inputs as sequences of tokens. We then concatenate these sequences together using separator tokens, truncating and padding them to fixed lengths. This is straightforward since all our inputs are text. See fig. 2 for reference. We also use the standard cross-entropy loss to train.

5 Experiments

We train our model on a subset of ∼1,020K edits from WikiDocEdits. We use a training/validation/test split of 1,000K/10K/10K edits, and train for 3 epochs with a fixed learning rate of 0.0001, and a batch size of 128. We use the T5-base implementation from Huggingface (Wolf et al., 2020), and finetune all weights in the model. We validate every 200 steps and select the model with the lowest validation loss.

5.1 Evaluation

For inference we use beam search with a beam width of 5, and keep the 5 highest ranked candidates, excluding any generation that parrots the source as this corresponds to making no edits.

Metrics We consider several metrics to evaluate our model. One natural metric to consider is BLEU ((Papineni et al., 2002)). BLEU shows high correlation with human judgement on machine translation (Papineni et al., 2002; Doddington, 2002). While this should not a priori transfer to evaluating different tasks, our task in fact bears a high similarity to machine translation because of how the output is constrained by the inputs. If, for example, the source sentence in an English to German translation task is "Sally met Lucy", the German translation must in some way mention Sally and Lucy. Similarly, in our task, if the source sentence is "Barack Obama was the 44th President of the United States", and the command is "add birth date", the edit must somehow mention a birth date somewhere. Thus, in our setting, BLEU makes sense as a metric since in principle a good model output should not deviate too far from the reference. We use macro-averaged

Comment	`added class of '13`
Source	`Krishna attended Dartmouth College where she was a double major in government and French.`
Target	`Krishna attended Dartmouth College where she was a double major in government and French` **`and graduated in the class of '13.`**
Comment	`sp`
Source	`Mountain State is currently` **`seeing`** `alternative accreditation by the Commission on Collegiate Nursing Education.`
Target	`Mountain State is currently` **`seeking`** `alternative accreditation by the Commission on Collegiate Nursing Education.`
Comment	`correct year of marriage (did not fit NSW records)`
Source	`He married Margaret Frances Prowse Shaw in Sydney in` **`1874.`**
Target	`He married Margaret Frances Prowse Shaw in Sydney in` **`1871.`**
Comment	`Rephrasing`
Source	`Entitled "It Feels Like Home (Re Invented) Tour 2011", it` **`contained his songs and`** `remakes of Alliage hits.`
Target	`Entitled "It Feels Like Home (Re Invented) Tour 2011", it` **`included many`** `remakes of Alliage hits` **`as well as some of his newer songs.`**

Table 3: Example edits from WikiDocEdits. The edited portions are highlighted in bold.

sentence-level BLEU with epsilon smoothing and equally weighted n-grams, with n up to 4.

One issue with BLEU is that the source and target sentences in our task are already very similar, so a model that simply parrots back the source sentence could achieve an unduly high score. Therefore, we also evaluate model outputs by comparing the word-level edits made by the model against the reference, where a word-level edit is a tuple of an operation, either insertion or deletion, a position, and a word. For example, in the edit "Barack Obama was the 44th President of the United States" $\rightarrow$ "Barack Obama, born August 4th 1961, was the 44th President of the United States", the set of word edits would look like $\{(\text{insert}, 2, \text{“,”}), (\text{insert}, 3, \text{“born”}), ...\}$. Now, denote the set of word edits between two sentences a and b as $\text{WE}(a, b)$. Then, with s the source sentence, s' the reference target sentence and h the target sentence generated by the model, we compute the precision

$$P_{\text{WE}}(s', h, s) = \frac{|\text{WE}(s', s) \cap \text{WE}(h, s)|}{|\text{WE}(h, s)|},$$

recall,

$$R_{\text{WE}}(s', h, s) = \frac{|\text{WE}(s', s) \cap \text{WE}(h, s)|}{|\text{WE}(s', s)|},$$

and F1 score,

$$F_{1,\text{WE}}(s', h, s) = 2 \cdot \frac{P_{\text{WE}} \cdot R_{\text{WE}}}{P_{\text{WE}} + R_{\text{WE}}}.$$

Finally, we compute sentence-level accuracy, which reports the proportion of edits for which the model output exactly matched the reference.

Baselines We use two baselines to compare our model to. First, we consider the parrot baseline that simply outputs the source sentence as is. The second baseline attempts to delete the source sentence and replace it with a new sentence. We use a pretrained GPT-2 model (Radford et al., 2019) that generates a sentence given the left context.

5.2 Results

Table 5 presents our main results. Notice that the parrot baseline is able to achieve a considerably high BLEU score, as expected, while the GPT-2 baseline surprisingly achieves a high word edit recall score. Our interactive neural editor model is able to beat both baselines across all metrics, as would be expected. Even on a harsh metric like accuracy our model achieves a nontrivial score, although we suspect most of the edits that the model gets exactly right are fluency edits. See table 6 for

Comment	Added more marriage info.
Reference edit	Johnson married Group 1 Crew member Manwell Reyes **in 2011.** Johnson married Group 1 Crew member Manwell Reyes **on June 11, 2011 in Half Moon Bay, California.**
Model edit	Johnson married Group 1 Crew member Manwell Reyes in 2011. Johnson married Group 1 Crew member Manwell Reyes in 2011 **in a ceremony at Half Moon Bay, California.**
Comment	another minor addition
Reference edit	They are more frequent than primary brain tumors. They are more frequent than primary brain tumors**, and are mainly a problem in adults, though children may also have secondary tumors.**
Model edit	**They** are more frequent than primary brain tumors. **Secondary brain tumors** are more frequent than primary brain tumors.

Table 4: Example outputs from Interactive Editor for two edits from the test data. The edit shown is the top-ranked generation from beam search, excluding the parrot generation if it occurs. The grounding and context are omitted here for brevity.

Model	Acc.	Word Edit			BLEU
		R	P	F1	
Baselines:					
Parrot baseline	0	0	0	0	0.67
GPT-2	0	0.38	0.05	0.07	0.00
Ablations:					
Only source	0.17	0.24	0.26	0.23	0.62
No command	0.20	0.28	0.31	0.28	0.65
No grounding	0.18	0.24	0.28	0.24	0.64
Our system:					
Interactive	**0.30**	**0.41**	**0.44**	**0.41**	**0.70**

Table 5: Evaluation of our model (Interactive Editor) against baselines and ablations.

a breakdown by edit type, and table 4 for example model outputs.

Ablations The middle rows of Table 5 show the results for three ablations of our model. The first ablation removes everything but the source sentence s. This is similar to the paraphrase setting (Gupta et al., 2018), and the editing setting in Faruqui et al. (2018) and Yin et al. (2018). We can see that including the context, grounding, and command as additional inputs yields significant improvements over only using the source sentence. We can also see from the second ablation that the commands are a crucial element in the model's performance. This is not surprising since without a command the model must guess what type of edit to make. Similarly, the model without grounding performs considerably worse than the full model, showing that the grounding is equally important as the command. Surprisingly, the last two ablations perform only marginally better than the first, meaning that removing the grounding in addition to the commands, or vice-versa, does not lead to a large drop in performance. This seems to suggest a synergistic effect between the command and the grounding, which makes sense since the model would not know what to do with the grounding without a command, and likewise, the model would not have access to the right information without the grounding, even if it knew what to edit from the command.

Breakdown by edit type The results of our full model are broken down by edit intention labels in Table 6. The columns report the same metrics as in our main table of results, with the exception of S-BLEU, which reports the BLEU score between the source sentence and target, and the last column, which reports the number of test edits that were classified into each category. With the caveat that intention labels come from an automatic classifier and not human annotation, we can observe that our model has varying performance across different types of edits. The model performs very well on fluency edits, but worse on content edits. This comes at no surprise given that fluency ed-

Intention Category	Acc.	Word Edit			BLEU	S-BLEU	#Edits
		P	R	F1			
Fluency	**0.36**	**0.49**	**0.47**	**0.46**	**0.76**	**0.73**	6244
Content	0.10	0.24	0.19	0.20	0.42	0.38	2792
Other	0.29	0.45	0.41	0.41	0.74	0.72	3027

Table 6: Breakdown of results by intention category for our full model. The categories are the same as in table 2.

Task	Preference (%)		
	Reference	Neutral	Interactive
Command	**41.00**	31.71	27.29
Grounding	29.14	34.86	**36.00**

Table 7: Human Evaluation: judging preferences for our system (Interactive Editor) vs. human references.

System A		System B	
Full +	**3.45**	2.55	Ablated +
Full -	**3.33**	3.12	Ablated -
Full +	3.45	3.33	Full -
Ablated -	**3.12**	2.55	Ablated +

Table 8: Human Evaluation: comparisons between absolute evaluations of different settings. Raters were asked whether edits were satisfactory. 0 corresponds to strong disagreement, and 5 to strong agreement. Systems are given by model (full or with the comment ablated), and whether the command was shown to the raters (+ or -). Bolded numbers indicate significant difference with $p < 0.0125$.

its should be easier as they usually correct minor mistakes, which a language model should be able to detect from pretraining. Content edits, on the other hand, require pulling the correct information from the grounding and incorporating it in the correct manner into the sentence. The S-BLEU scores confirm this since the source sentences in the fluency examples are much more similar to the target sentences than for the content edits. In fact, when looking at the absolute improvement of the BLEU over the S-BLEU scores, the model performs equally well on both types of edits.

5.3 Human Evaluations

We conducted two rounds of human evaluations, each time across 200 examples from our test set. Annotators were crowd sourced, and each example was rated by seven judges for a total of 1400 judgements.[8]

Command and Grounding In our first round of human evaluations we compared our model's top output from beam search to the reference edit. There were two tasks. In the first task, we asked judges to choose which system better accomplished the command q. In the second, we asked which system was more faithful to the grounding $\mathcal{G}$. Table 7 presents the results. Although there is a clear preference for the Reference edits in the command-related task, 59% of judgments suggest that Interactive Editor may be equal to or better than the reference.[9] In the grounding task, Interactive Editor demonstrates good correspondence with the background material.[10] Judges were further asked whether the retrieved grounding was relevant to the context D: 92.86% of judgments recorded the grounding as either "Somewhat relevant" or "Very relevant".

Absolute Scoring We also evaluated the overall quality of model outputs. We considered our full model, and our ablated model that only takes the source sentence as input. We also considered showing and hiding the edit commands, for a total of 4 settings. For a given setting, raters were asked whether they found each of the top 3 model outputs satisfactory. Table 8 presents the results for the top model outputs, with bootstrapped p-values for pairwise comparisons. We use a Bonferroni corrected $\alpha = 0.0125$ to determine significance. Note that our full model outperforms our ablated model in the first two comparisons. Inter-

[8]The annotators were remunerated at a rate above the prevailing Seattle minimum wage at the time.

[9]The high percentage of Neutral judgments here may be partially attributable to other factors. Majority Neutral judgments are observed for approximately 65% of those examples that received at least one Neutral judgment. This suggests many commands may not be readily interpretable to judges.

[10]Appendix E presents some additional automatic metrics to measure the faithfulness of the model to the grounding.

estingly, the difference is smaller when the raters are not shown the commands. Additionally, only the ablated model is rated differently depending on whether the commands are shown. This is to be expected since the ablated model is not likely to be faithful to the commands. In addition to reporting the mean scores from the raters, we can also look at the number of examples where at least one of the top model outputs was found satisfactory by human judges (i.e. scored higher than 3). We find that, when showing the edit commands, at least one of the outputs from our full model was satisfactory in 85.83% of cases versus 60.17% for the ablated model.

6 Discussion

Text	`Geoff Hinton is an English tennis player.`
Command	**`fix profession`**
Text	`Geoffrey Hinton is a computer science professor at the University of Toronto.`
Command	**`add nationality`**
Text	`Geoffrey Hinton is an English-Canadian computer science professor at the University of Toronto.`
Command	**`add birthdate`**
Text	`Geoffrey Hinton (born 1946) is an English-Canadian computer science professor at the University of Toronto.`
Command	**`add most famous work`**
Text	`Geoffrey Hinton (born 1946) is an English-Canadian computer science professor at the University of Toronto. Geoffrey Hinton is most famous for his work on artificial neural networks.`

Table 9: An example of a multi-turn interaction with our model. At each turn, the edit was chosen among the top 3 outputs returned by beam-search. See table 12 in the appendix for the grounding used in this example.

This paper focuses on the task of editing individual sentences, which we believe to be a challenging task for NLP, as it involves making nuanced changes to text according to natural language commands. We also believe this task has useful applications, particularly in speech-to-text scenarios, where it may be more convenient to speak out a command rather than edit the text directly. However, we also wish to emphasize that this task is a step towards a larger goal of interactive document generation, and that there are many interesting future directions to explore in this space. While this paper has focused on single interactions (i.e. making isolated edits to text), it would be worth modeling multiple interactions between the user and model. One can imagine that there may be a natural order in which to make edits, such as adding information at the start, and fine-tuning the language at the end. It is an open question whether or not a model could learn this. For illustration, table 9 gives an example of using our model to make several edits in order to create a sentence. Ultimately, this may look more like a dialogue than a sequence of commands coming from the user. Additionally, it would also be interesting to look at other settings where a model must generate a complex, structured object for a user, such as code, or images. We hope that our text editing task, as a first step, can demonstrate the potential for interactive generation systems, and that it will encourage the community to pursue more ideas in this space.

7 Related Work

Grounded Generation Large language models can generate fluent text (Radford et al., 2019; Brown et al., 2020; Raffel et al., 2020), but they have a tendency to hallucinate facts (Wiseman et al., 2017). Thus, several works have explored using various forms of grounding to enable models to generate factually consistent texts (Koncel-Kedziorski et al., 2019; Liu et al., 2018b; Prabhumoye et al., 2019; Liu et al., 2018a; Guu et al., 2020). Our work uses grounding to ensure that edits are factually correct, although our task differs from previous work because of the user command, which requires specific information to be retrieved from the grounding during generation.

Controllable Generation While grounding can be seen as a way to implicitly control the contents of generated text, other works have explored more explicit forms of control. Hokamp and Liu (2017) and Zhang et al. (2020) use lexical constraints, while Keskar et al. (2019) and Dathathri et al. (2019) control higher level attributes of text, such as style, tone, or topic. Our task instead

uses natural language commands, which can flexibly express different types of constraints, ranging from low-level lexical ones, to high-level topical ones. In this sense, we can also draw the parallel to dialog response generation (Ghazvininejad et al., 2018; Dinan et al., 2018), task-oriented dialog (Gao et al., 2018), or open domain question answering (Min et al., 2019; Chen et al., 2017), that also involve user responses or queries, although these tasks are not concerned with text generation in the context of document creation.

Story Generation The task of Document Generation considered in our work bears similarity with work on generating long-form narratives (Jain et al., 2017). While earlier work in Story Generation focused more on plan-based architectures (Lebowitz, 1985), more recent work moved towards end-to-end approaches (Fan et al., 2018) allowing generation to be unconstrained and creative. As narratives are often aimed at particular goals expressed in terms of outlines and plans, much of the literature in Story Generation is framed as a form of controllable generation, using storylines (Peng et al., 2018), events (Martin et al., 2017; Harrison et al., 2017), plot words or word skeletons (Xu et al., 2018; Ippolito et al., 2019), plans (Yao et al., 2019), story ending (Tambwekar et al., 2019), and outlines (Rashkin et al., 2020) as various forms of constraints. Our work takes a significantly different approach, as we treat document or story generation as an iterative process that allows a human to generate a full document from scratch, but also allows constraints to be more dynamic (e.g., add nationality in Table 9 only if the system missed that the first time).

Text Editing Several previous works have focused on text editing. Guu et al. (2018) generate sentences by editing prototypes taken from their training corpus, although they use editing only as a means for language modeling. Wu et al. (2019) expand upon Guu et al. (2018)'s setting, but for dialog. More related to our own setting, Faruqui et al. (2018) propose WikiAtomicEdits, a dataset of edits crawled from Wikipedia. However, they consider a much narrower definition of edits than our data does. Yin et al. (2018) use WikiAtomicEdits and propose the task of learning to represent edits, which Marrese-Taylor et al. (2020) expand using a variational approach. In contrast, we are more interested in generating edits rather than representing them. Related to Wikipedia data, Pryzant et al. (2020) also used Wikipedia revision histories to learn to debias text, whereas we considered general edits. Iso et al. (2020) propose a fact-based text editing task, but they do not consider control or other types of edits. Another related task to text editing is text paraphrasing (Gupta et al., 2018), however paraphrasing usually conserves the meaning of a sentence. While the edits we consider include meaning-preserving edits, we are mostly interested in edits that affect meaning.

8 Conclusion

In this work we argued that text generation should be interactive, and, as a means towards that end, we proposed a general text editing task, where a system must edit a document in response to a user command. In our specific instance of the task we considered single-sentence edits, and we crawled a dataset of several million edits from Wikipedia that included commands, in the form of editor comments, as well as grounding documents. We then showed that training a transformer-based model on our data, while initializing with pre-trained language model weights, yields encouraging results on both automatic and human evaluations. Additionally, our ablation studies showed the crucial role played by the user command and grounding. Breaking down our results by types of edits, we saw that our model not only performs well on easier fluency edits, but also on much harder content edits. Finally, we discussed future research directions for interactive document generation, as well as possible extensions to other domains such as images or code.

Acknowledgments

The authors would like to thank Thomas Hofmann, as well as Sudha Rao, Matt Richardson, Zhang Li, Kosh Narayanan, and Chandra Chikkareddy for their helpful suggestions.

References

Giuseppe Attardi. 2015. WikiExtractor. https://github.com/attardi/wikiextractor.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

T. Brown, B. Mann, Nick Ryder, Melanie Subbiah, J. Kaplan, P. Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, G. Krüger, Tom Henighan, R. Child, Aditya Ramesh, D. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, E. Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, J. Clark, Christopher Berner, Sam McCandlish, A. Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proc. of ACL*.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and play language models: A simple approach to controlled text generation. In *Proc. of ICLR*.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2018. Wizard of wikipedia: Knowledge-powered conversational agents. In *Proc. of ICLR*.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proc. of ACL*, pages 889–898.

Manaal Faruqui, Ellie Pavlick, Ian Tenney, and Dipanjan Das. 2018. Wikiatomicedits: A multilingual corpus of wikipedia edits for modeling language and discourse. In *Proc. of EMNLP*, pages 305–315.

Jianfeng Gao, Michel Galley, and Lihong Li. 2018. Neural approaches to conversational ai. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1371–1374.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, W. Dolan, Jianfeng Gao, Wen tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *Proc. of AAAI*.

A. Gupta, A. Agarwal, Prawaan Singh, and P. Rai. 2018. A deep generative framework for paraphrase generation. In *Proc. of AAAI*.

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.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. *ArXiv*, abs/2002.08909.

B. Harrison, Christopher Purdy, and Mark O. Riedl. 2017. Toward automated story generation with markov chain monte carlo methods and deep neural networks. In *AIIDE Workshops*.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proc. of ACL*, pages 1535–1546.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Proc. of ICML*, pages 1587–1596.

Daphne Ippolito, David Grangier, Chris Callison-Burch, and Douglas Eck. 2019. Unsupervised hierarchical story infilling. In *Proceedings of the First Workshop on Narrative Understanding*.

Hayate Iso, Chao Qiao, and Hang Li. 2020. Fact-based text editing. In *Proc. of ACL*, pages 171–182.

Parag Jain, Priyanka Agrawal, Abhijit Mishra, Mohak Sukhwani, Anirban Laha, and Karthik Sankaranarayanan. 2017. Story generation from sequence of independent short descriptions. *arXiv preprint arXiv:1707.05501*.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. *ArXiv*, abs/1909.05858.

T. Kiss and J. Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32:485–525.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers. In *Proc. of NAACL-HLT*.

M. Lebowitz. 1985. Story-telling as planning and learning. *Poetics*, 14:483–502.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018a. Generating wikipedia by summarizing long sequences. In *Proc. of ICLR*.

T. Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Z. Sui. 2018b. Table-to-text generation by structure-aware seq2seq learning. In *Proc. of AAAI*.

Edison Marrese-Taylor, Machel Reid, and Y. Matsuo. 2020. Variational inference for learning representations of natural language edits. *ArXiv*, abs/2004.09143.

Lara J Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark O Riedl. 2017. Event representations for automated story generation with deep neural nets. *arXiv preprint arXiv:1706.01331*.

Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Knowledge guided text retrieval and reading for open domain question answering. *ArXiv*, abs/1911.03868.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

Nanyun Peng, Marjan Ghazvininejad, Jonathan May, and Kevin Knight. 2018. Towards controllable story generation. In *Proceedings of the First Workshop on Storytelling*, pages 43–49.

Peter Pirolli and Stuart Card. 2005. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of international conference on intelligence analysis*.

Shrimai Prabhumoye, Chris Quirk, and Michel Galley. 2019. Towards content transfer through grounded text generation. In *Proc. of NAACL-HLT*, pages 2622–2632.

Reid Pryzant, Richard Diehl Martinez, Nathan Dass, Sadao Kurohashi, Dan Jurafsky, and Diyi Yang. 2020. Automatically neutralizing subjective bias in text. In *Proc. of AAAI*, volume 34, pages 480–489.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report, Open AI.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. 2020. PlotMachines: Outline-conditioned generation with dynamic plot state tracking. In *Proc. of EMNLP*.

Pradyumna Tambwekar, Murtaza Dhuliawala, Lara J Martin, Animesh Mehta, Brent Harrison, and Mark O Riedl. 2019. Controllable neural story plot generation via reward shaping. In *Proc. of IJCAI*, pages 5982–5988. AAAI Press.

Sam Joshua Wiseman, Stuart Merrill Shieber, and Alexander Sasha Matthew Rush. 2017. Challenges in data-to-document generation. In *Proc. of EMNLP*. Association for Computational Linguistics.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proc. of EMNLP*, pages 38–45.

Yu Wu, Furu Wei, Shaohan Huang, Yunli Wang, Zhoujun Li, and Ming Zhou. 2019. Response generation by context-aware prototype editing. In *Proc. of AAAI*, volume 33, pages 7281–7288.

Jingjing Xu, Xuancheng Ren, Yi Zhang, Qi Zeng, Xiaoyan Cai, and Xu Sun. 2018. A skeleton-based model for promoting coherence among sentences in narrative story generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4306–4315, Brussels, Belgium. Association for Computational Linguistics.

Diyi Yang, Aaron Halfaker, Robert Kraut, and Eduard Hovy. 2017. Identifying semantic edit intentions from revisions in wikipedia. In *Proc. of EMNLP*, pages 2000–2010.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proc. of AAAI*, volume 33, pages 7378–7385.

Pengcheng Yin, Graham Neubig, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L Gaunt. 2018. Learning to represent edits. In *Proc. of ICLR*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019a. Bertscore: Evaluating text generation with bert. In *Proc. of ICLR*.

Xizhe Zhang, Dheeraj Rajagopal, Michael Gamon, Sujay Kumar Jauhar, and Chang-Tien Lu. 2019b. Modeling the relationship between user comments and edits in document revision. In *Proc. of EMNLP-IJCNLP*.

Yizhe Zhang, G. Wang, C. Li, Zhe Gan, Chris Brockett, and B. Dolan. 2020. Pointer: Constrained text generation via insertion-based generative pre-training. *ArXiv*, abs/2005.00558.

A Data Processing pipeline

This section describes our pipeline to obtain atomic edits from Wikipedia revisions in more detail. We start by filtering the revisions in the data. In particular, following (Zhang et al., 2019b), we only keep revisions that affect a single section, and we exclude revisions that do not contain an editor comment. We also exclude certain page types like talk or user pages.

We then strip the Wikipedia markup in the retrieved text, using the WikiExtractor script (Attardi, 2015). This removes most markup and Wikimedia templates from the text. Because the markup language used on Wikipedia is not completely formalized[11], and because malformed markup often appears in intermediate versions of Wikipedia pages, there is no guarantee that we can remove all the markup from the text.

We then split each section into sentences using the Punkt sentence tokenizer (Kiss and Strunk, 2006) provided in the NLTK python package (Bird et al., 2009).

After splitting into sentences, we attempt to match the sentences from the pre-edit (source) document to the sentences in the post-edit (target) document. Unmatched sentences will be candidates for edits. Similarly to (Faruqui et al., 2018), for each sentence s_i in the source document, we only look at the target sentences $\{t_{i-k}, ..., t_i, ..., t_{i+k}\}$, with $k = 20$. This avoids the quadratic complexity of looking at all matches.

We then filter out revisions that contain more than one sentence-level edit to ensure that the comment is relevant. If there is a single unmatched source, respectively target, sentence, we consider it a sentence deletion, respectively insertion. Because we do not look at all matches between source and target sentences, a sentence may remain unmatched if, in the target document, it was moved more than k sentences away compared to the source document. Thus we only keep a sentence insertion or deletion if the total number of source and target sentences differ by one. If there are both an unmatched source sentence s and target sentence t, we consider them to form an edit $s \rightarrow t$ if $f(s,t) > \epsilon$, where f is the BLEU score and $\epsilon = 0.1$.

As a final step, we filter out edits that involve sentences with markup punctuation. We have found that this helps remedy the shortfalls of the markup removal step, since it often leaves behind markup symbols. While there may be valid sentences that use markup punctuation, we do not expect them to make up a significant part of the data, nor do we expect them to be significantly different from regular sentences, except for their use of unusual punctuation.

B Grounding Search Query Construction

For a given edit, we combine the relevant page and section titles with keywords from the target sentence to construct a query that we use to retrieve grounding from a commercial search engine. In order to identify keywords we look at document frequency

$$\text{df}(w) = \frac{|\{D \in \mathcal{D} \,|\, w \in D\}|}{|\mathcal{D}|},$$

where $\mathcal{D}$ is a sample of $500,000$ Wikipedia articles taken from the Tensorflow Wikipedia dataset.[12] We consider words w with $\text{df}(w) < 0.01$ to be keywords.

C Grounding Document Reranking

Because the combined length of the grounding snippets we retrieve far exceeds the capacity of our model, we rerank the retrieved snippets using an information extraction score. We then merge the ranked snippets and take only the first $N = 512$ tokens. Following (Liu et al., 2018a) we use tf-idf scores to rerank. For a given edit $s \rightarrow s'$, with retrieved grounding documents $\mathcal{G}$, the information extraction score of snippet $G \in \mathcal{G}$ is

$$\text{score}(G) = \sum_{w \in s'} \text{tf-idf}(w, G),$$

where the tf-idf score of word w is

$$\text{tf-idf}(w, G) = N_w(G) \cdot \log\left(\frac{N_g}{N_{gw}}\right),$$

where $N_w(G)$ is the number of occurrences of w in G, N_{gw} is the number of documents in $\mathcal{G}$ that contain w, and N_g is the number of documents in $\mathcal{G}$.

[11] See https://www.mediawiki.org/wiki/Markup_spec for a discussion.

[12] https://www.tensorflow.org/datasets/catalog/wikipedia

Label	Description	%Edits	%Orig.
Counter-Vandalism	Revert or otherwise; remove vandalism	0.05	1.90
Fact-update	Update numbers, dates, scores, episodes, status, etc. based on newly available information	1.57	5.50
Copy-editing	Rephrase; improve grammar, spelling, tone, or punctuation	29.22	11.80
Wikification	Format text to meet style guidelines, e.g. add links or remove them where necessary	21.12	33.10
Vandalism	Deliberately attempt to damage the article	1.01	2.50
Simplification	Reduce the complexity or breadth of discussion; may remove information	3.13	1.60
Elaboration	Extend/add substantive new content; insert a fact or new meaningful assertion	9.50	12
Verifiability	Add/modify references/citations; remove unverified text	7.63	5.40
Process	Start/continue a wiki process workflow such as tagging an article with cleanup, merge or deletion notices	0.62	4.40
Clarification	Specify or explain an existing fact or meaning by example or discussion without adding new information	3.54	0.70
Disambiguation	Relink from a disambiguation page to a specific page	0.70	0.30
Point-of-view	Rewrite using encyclopedic, neutral tone; remove bias; apply due weight	0	0.30
Unlabeled	No label	21.39	1.20

Table 10: Breakdown of the edits in our data by intention label. The descriptions are taken from Yang et al. (2017). %Edits gives the prevalence of each label in our data, while %Orig. gives the prevalence in the hand-labelled dataset presented in Yang et al. (2017). The percentages do not total 100 because edits can have multiple labels.

Comment	Reword
Source	ByteDance responded by adding a kids-only mode to TikTok which **allows music** videos **to be recorded, but not posted** and by removing some accounts and content **from those determined to be underage.**
Target	ByteDance responded by adding a kids-only mode to TikTok which **blocks the upload of** videos, **the building of user profiles, direct messaging,** and **commenting on other's videos, while still allowing the viewing** and **recording of** content.
Comment	corrected tense for decedent
Source	While Bob Steward **has** not **been** an active producer since 1992, he **serves** as a Creative Consultant in his son's new production company, Steward Television, and **is** listed on the official website as Steward Television's founder.
Target	While Bob Steward **was** not an active producer since 1992, he **served** as a Creative Consultant in his son's new production company, Steward Television, and **was** listed on the official website as Steward Television's founder.
Comment	fixed spelling for Walter Yetnikoff
Source	Mottola was hired by Sony Music (then known as CBS Records) by its controversial President Walter **Yentlkoff** to run its U.S. operations.
Target	Mottola was hired by Sony Music (then known as CBS Records) by its controversial President Walter **Yetnikoff** to run its U.S. operations.

Table 11: More example edits from WikiDocEdits. The edited portions are highlighted in bold.

Geoffrey Everest Hinton CC FRS FRSC (born 6 December 1947) is an English Canadian cognitive psychologist and computer scientist, most noted for his work on artificial neural networks.Since 2013 he divides his time working for Google (Google Brain) and the University of Toronto.In 2017, he cofounded and became the Chief Scientific Advisor of the Vector Institute in Toronto. Geoffrey Hinton : index. Department of Computer Science : email: [REDACTED] : University of Toronto : voice: send email: 6 King's College Rd. We would like to show you a description here but the site won't allow us. Geoffrey's great grandfather, the mathematician [REDACTED] Charles Hinton, coined the word \tesseract" and popularized the idea of higher dimensions, while his father, Howard Everest Hinton, was a distinguished entomologist. Geoffrey Hinton is a fellow of the Royal Society, the Royal Society of Canada, and the Association for the Advancement of Artificial Intelligence. He is an honorary foreign member of the American Academy of Arts and Sciences and the National Academy of Engineering, and a former president of the Cognitive Science Society. Geoffrey Hinton. Emeritus Prof. Comp Sci, U.Toronto & Engineering Fellow, Google. Verified email at cs.toronto.edu - Homepage. machine learning psychology artificial intelligence cognitive science computer science. Articles Cited by Co-authors. Title. Sort. Sort by citations Sort by year Sort by title. Geoff Hinton was born in Wimbledon in 1947 to Howard Hinton, an entomologist, and a schoolteacher mother, Margaret Clark. The childhood Hinton describes is a mash-up of Lemony Snicket, ... As the first of this interview series, I am delighted to present to you an interview with Geoffrey Hinton. Welcome Geoff, and thank you for doing this interview with deeplearning.ai. ⟩⟩ Thank you for inviting me. ⟩⟩ I think that at this point you more than anyone else on this planet has invented so many of the ideas behind deep learning. Talks by Geoffrey Hinton. The next generation of neural networks A 45min version of this talk which was given at the 10 year celebration of the Microsoft Cambridge Research Laboratory. the original powerpoint file version for most browsers.ps version with 4 slides per page. Very gentle after-dinner version of IJCAI-2005 Research Excellence ...

Table 12: Grounding used for the example in table 9. Parts indicated by [REDACTED] were removed for containing sensitive material.

Coverage corpus	Percentiles			Mean
	25%	50%	75%	
All Inputs	0.66	0.75	0.83	0.74
Grounding	0.50	0.62	0.73	0.61
Comment	0.22	0.31	0.46	0.36

Table 13: R_{BERT} statistics of inserted words for edits in WikiDocEdits. All statistics were computed on a 1% subsample of the data. The BERT embeddings used to compute R_{BERT} were produced using a pretrained BERT base model. The idf weights were computed from a sample of 500,000 Wikipedia pages. Each row represents a different recall when considering a different coverage corpus $\mathcal{C}$.

D Grounding Coverage Analysis

We are also interested in knowing how well edits in the data are covered by the inputs (i.e. D, s, q, or $\mathcal{G}$), where an edit is well covered if the information necessary to produce the edit appears somewhere in the inputs. To measure coverage we use word recall: how many words that were inserted in an edit also appear in the grounding? However, because simple recall fails to account for synonyms, or the context in which words appear, we use the BERTScore (Zhang et al., 2019a) recall. This allows for fuzzy matching between BERT embeddings instead of requiring exact word matches. We also use idf scores to weigh words, since we are mostly interested in covering rare words, which are more likely to be meaning-carrying. We can define the BERT recall, R_{BERT}, for a sentence edit

Coverage corpus	Percentiles			Mean
	25%	50%	75%	
Full Model	0.54	0.66	0.75	0.64
Ablated Model	0.46	0.57	0.67	0.57

Table 14: R_{BERT} statistics of inserted words across test edits in WikiDocEdits. The BERT embeddings used to compute R_{BERT} were produced using a pretrained BERT base model. The idf weights were computed from a sample of 500,000 Wikipedia pages. In all rows, the considered corpus $\mathcal{C}$ corresponds to the grounding.

$s \rightarrow s'$, with respect to some text corpus $\mathcal{C}$ as

$$\frac{\sum_{w \in s' \setminus s} \text{idf}(w) \cdot \max_{w' \in \mathcal{C}} \text{BERT}(w)^T \text{BERT}(w')}{\sum_{w \in s' \setminus s} \text{idf}(w)},$$

where $s' \setminus s = \{w \in s' | w \notin s\}$, and idf$(w)$ are the inverse document frequency scores computed on a random sample of $500K$ Wikipedia pages.

Table 13 reports the coverage statistics for our subsample of the data. We used an uncased BERT base model to compute the embeddings. The first row reports the coverage of the target by all of the inputs, namely the command, grounding, context, and source sentence. The second row shows the coverage by the grounding alone. Note that, even with just the grounding, coverage is already fairly high. Finally, the last row presents the coverage by the command alone, which shows that it also provides grounding.

E Additional Factuality Results

In addition to human evaluations, we also used automatic metrics to evaluate how faithful our model is to the grounding.

BERT Recall Similarly to the coverage analysis in appendix D, we can use R_{BERT}, with the grounding as $\mathcal{C}$, to assess how well each word inserted by the model is supported by the grounding. The only difference is that the model output now replaces the reference target s' in the formula for R_{BERT}. Table 14 gives the summary statistics for R_{BERT} across our test set, computed on the outputs of our full model, and the ablated model without grounding. Note that we only consider edits where the model makes at least one insertion. The ablated model serves as a baseline to compare the grounded model to. This baseline achieves a high R_{BERT} score, likely because of spurious matches with the grounding. Nevertheless, our grounded model is still more faithful to the grounding, as expected.

Grounding Usage While R_{BERT} attempts to measure how faithful the model is to the grounding (i.e. is the information inserted by the model found in the grounding?), we can also attempt to measure how much the grounding is used (i.e. how much of the information inserted by the model is only found in the grounding?). One simple approach is to look at how many words inserted by the model are found in the grounding but not in the rest of the inputs. While this isn't obvious to compute similarities between BERT embeddings, we can use exact word matches instead. For the model without grounding we find that in 30.48% of edits in the test set (with at least one insertion), at least one of the words inserted by the model is found in the grounding but not in the rest of the inputs. For the full model, this number increases to 48.66% as expected. The ablated model appears to insert words exclusive to the grounding in a high proportion of edits. However, this could be due to fluency edits, where the model might insert a functional word that happens to only appear in the grounding. If we restrict our attention to content edits, as defined in section 3.2, the ablated model inserts grounding-exclusive words in only 36.85% of edits, and 65.40% for the full model.