Graph2step: A System for Knowledge Driven Procedural Step Generation

Pedro Colon-Hernandez
MIT Media Lab
Cambridge, MA 02139
pe25171@media.mit.edu

Abstract

Procedural step generation (i.e. instruction generation) is an important task, particularly because of its applicability on fields such as usability of technology and education. An effective procedural step generation system should be capable of generating steps to guide a person to accomplish a procedure, and in the case of a mishap in the procedure, generate steps to correct the mishap and continue and complete the original procedure. Procedural step generation is complicated for modern natural language systems, particularly because of knowledge that may be implicit in the steps, and essential for its completion. We present a work-in-progress of a multi-part system that is capable of, given a goal and facts related to a goal, to generate procedural steps to accomplish a procedure. The system is also capable of reusing past knowledge through a neural memory to handle goal changes (i.e., mishaps) while performing procedures. To accomplish this, we leverage a contextual commonsense inference model which can generate contextually relevant facts (i.e., assertions) about a procedure, and train a model which selects facts that are necessary to accomplish a goal and translates these into a procedural step.

1 Introduction

How-to questions (or questions in which a user seeks help on a procedure) account for a significant portion of the web queries (1). Although many people seek answers to these procedural questions, most people find reading instructions to be cognitively demanding and prefer to be guided through procedures by others (2). This is not a scalable strategy, as it requires another person to be present and informed about the procedure at all times. Conversational agents offer a possibility to answer such queries for procedures on a large scale, particularly because of their widespread adoption.

However, popular modern assistants (e.g., Siri, Google, and Alexa) handle just a small fraction of these procedures, such as cooking, or those that would fall under a brief web answer that might be read out loud. Additionally, the procedures that the agents can manage are almost if not directly scraped from the web and do not account for exceptions (i.e., human error), such as miscommunications and problems encountered while performing the procedure. To make matters worse, scraped procedures are written to be interpreted in web pages, rather than in voice interactions (i.e., the steps may not be conversational) which can lead to issues when reading them out loud or utilizing them in a conversational agent.

Attempts have been made in research to develop guiding agents by either developing systems that focus on transmitting instructions in a comprehensible manner or assuming that experts will give the essential knowledge for systems to function (3; 4). Other types of systems have tried to do procedure abstractions, but they use parts of the text verbatim and require external knowledge always (5; 6). Furthermore, neither of these approaches handles exceptions in procedures.

To power a conversational agent that can guide people on procedures, and can handle exceptions in procedures, we present a work-in-progress system that is capable of, from a list of contextually derived facts and a goal, select the “key” facts, order them, and convert them into a procedural step. In contrast to other methods(7), our system leverages both neural (i.e., transformer-based large pretrained models (8; 9) and planners (10)), and symbolic (i.e., knowledge graph (11)) components to perform this procedural step generation.

The system that we have described thus far, which generates a set of contextual facts and a goal, uses a neural planner along with encodings of prior history of steps, to produce an ordered set of facts, and finally translates these with a sequence to sequence model into a procedural step, may seem complex, however we now describe the reasoning for this. By abstracting procedures into knowledge graphs, we are no longer tied to how they are written, or represented, this permits us to implement an open-domain, web-powered system in which there could be a set of parsers to ingest results from a search engine, to be able to power a conversational agent to guide people in procedures. Apart from this disentanglement of sources, we also gain the ability to perform operations in the abstracted, knowledge graph space. Some examples of things that we could do are: modifying the knowledge representation (e.g., adding in facts that may be personalized to a user’s preferences), combining knowledge representations (i.e., facts from different sources on how to perform a step), or even combining procedures to handle exception cases. The way that we formulated our system, it takes a goal and facts, an avenue of research that we are pursuing is actually changing the goal halfway through and adding in a new set of facts such that it represents exceptions in a procedure. One example is changing the goal of how to write a check to that of how to correct a check, when a mistake is made while writing a check.

In more detail, the system that we present abstracts procedures into their essential procedural knowledge facts (i.e., assertions) in the form of a knowledge graph. With this graph, we would then be able to reason about the procedure and deduce a path through the graph or an ordered set of facts that can be translated into steps. What makes this abstraction particularly interesting, is that if an error occurs during the procedure, the abstracted knowledge can be changed or updated to handle error-handling sub-procedures. Additionally, the paths or ordering can be recalculated to provide alternative explanations for procedures. When leveraged with natural conversation patterns (12), the presented system can be utilized to convey the steps in a procedure in a manner akin to receiving assistance from a real person. In summary, we present a system that is capable of procedure abstraction, procedural knowledge reasoning, and knowledge to step generation, allowing conversational agents to lead individuals through procedures.

2 Related Work

There has been work on systems that to teach humans through explicit instruction. Some recent guidance systems utilize augmented reality to highlight objects and in a user’s field of view project instructions to guide people but require that the procedure be given in advance. This can be a problem because procedures may change as time passes, documentation may not be available, and sometimes they may require experts to generate these instructions. One is the Cognition-based interactive Augmented Reality Assembly Guidance System (CARAGS) (5) and Intelligent Augmented Reality Training for Motherboard Assembly (6). Other systems take the approach of abstracting a procedure to guide people. One such is DESIRE (3), a system that was built by IBM to improve the handling of "How-to" support ticket questions by building an instruction flow-graph-like structure that could be traversed to relay a procedure through a chat-bot. Another work is Enabling Interactive Answering of Procedural Questions (4) which presents a way of generating tree-like structures from procedural instructions. However, these systems do not consider whenever a user runs into an issue in the procedure and how that can be resolved, nor how to handle unregistered/new procedures, nor utilize additional knowledge that is not found in the procedure to give helpful insight.

In addition to these guidance systems, there has been development in procedural understanding utilizing modern language models. Explaining Action Dependencies (XPAD) (13) is a model which tries to understand dependencies between steps and how these change the states of entities. The system produces a state matrix for every step and a dependency graph of steps that shows which step contributed to which change. DynaPro (14) goes one step further and incorporates a language model to further understand how entities change through steps. These procedural understanding projects, although promising in their capability to track states and entities, are disconnected from how they could be employed in practice to help with procedure guidance. An interesting read with more historical works and the current status of procedural understanding and reasoning can be found in (7).

3 System Overview

We now give a detailed description of the subsystems of the work-in-progress system that we are presenting. A system overview of the can be seen in Figure 1, along with a general description of how steps are generated. Broadly our system works by taking a set of contextual facts and a goal, using a neural planner along with encodings of prior history of steps, to produce an ordered set of facts which are then translated with a sequence-to-sequence model into a procedural step. We now go into details on how specific aspects of the process are done.

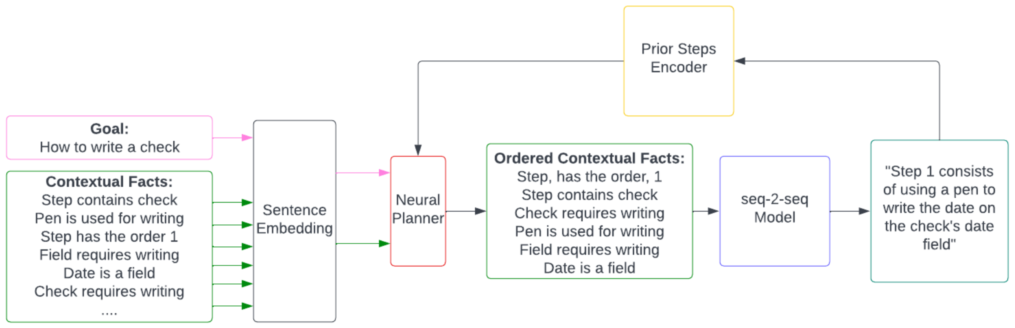
Prior Steps Encoder
Goal: How to write a check
Contextual Facts: Step contains check Pen is used for writing Step has the order 1 Field requires writing Date is a field Check requires writing ....
Sentence Embedding
Neural Planner
Ordered Contextual Facts: Step, has the order, 1 Step contains check Check requires writing Pen is used for writing Field requires writing Date is a field
seq-2-seq Model
"Step 1 consists of using a pen to write the date on the check's date field"

Figure 1: Architecture overview of graph2step system. From left to right, the system is trained by giving a set of contextual facts and a goal, which are embedded and passed into a neural planner along with any prior step encodings, to produce an ordered set of facts which are then translated into a procedural step.

3.1 Procedure Abstraction: Contextual Fact Generation

To generate contextual facts, which serve as an abstraction of our procedure, we utilize a custom contextual commonsense (i.e., dialogue-aware) inference model similar to ParaCOMET (15). Our model is capable of ingesting both a text and a target sentence from that text. It then infers a commonsense assertion from that target sentence given the text context. In contrast to ParaCOMET, our single model is trained on multiple knowledge sources (ATOMIC 2020(16), ConceptNet (17), and GLUCOSE (18)). Our model is also trained with a mechanism called hinting, a prefix prompting strategy that uses both hard and soft prompts(19; 20), which permits controlling the contextual commonsense inference. An example of how hinting works, and a contextual commonsense inference for our model, is seen in the following example:

> **Model Input:** <story> The hockey game was tied up. The red team had the puck. They sprinted down the ice. They cracked a shot on goal! They scored a final goal! <sentence> *They scored a final goal!* (<subject>The red team scores)
> **Model Target/Output:** The red team scores, causes or enables, they win the game

In the example, the highlighted part of the text is the hint which serves as a control signal to tell the model to predict a fact or assertion whose subject[1] is: The red team scores. Without this hint, our model is allowed to predict about entities freely. When we perform hinted contextual commonsense inference with our model, we can also extract keywords, nouns, entities, and others from the text, and use these as hints to infer facts related to these. When we perform contextual commonsense

[1]In our work, facts are represented as tuples. They are similar to a subject, verb, object tuple, but in our case it is subject, relation, object in which a relation can be something broader than a simple verb or verb phrase. Our relations are derived from ATOMIC and ConceptNet relations.

inference on every sentence in a procedure, we get a contextualized knowledge graph of every step in our procedure, which we can see as an abstraction of our procedure.

Commonsense information refers to the millions of basic facts and understandings possessed by most people (21). We hypothesize that by doing contextual commonsense inference as we are doing, and generating contextual commonsense facts, these can serve as a prior to make/help in inferences in procedural related tasks. Some evidence of this being useful is in ProStruct (22). In this work, the authors state that "The commonsense constraints we have used for ProPara[2] are general, covering the large variety of topics contained in ProPara[...]". With this wide coverage, we could generalize better the ability to infer how an entity behaves in a procedure for unseen entities. One work shows some promise of this is (24). The work builds a collection of relevant "query cases" which are "link[s from] annotated answers to semantic interpretations (i.e. logical statements)". These query cases are essentially mappings of text examples to semantic assertions. If the query cases give evidence of the answer, then it is likely that the question at hand has a similar answer.

3.2 Procedure Reasoning and Generation: Step Planner and Step Generator

After generating a contextual knowledge graph of a procedure as mentioned in the prior subsection, we then encode the tuples in this knowledge graph using either a sentence encoder model (25), or a relational graph convolution network (RGCN) (26), and pass these on to a neural planner system similar to that of (10), which simply rates the most useful K facts to generate a procedural step text. With these K facts, we then pass them on to a sequence-to-sequence model, along with a sentence encoding of the goal, and utilize a language modeling objective to force the sequence-to-sequence model to convert the useful facts into a procedural step. We utilize the scaled encoded fact (usefulness score multiplied by the encoded fact) as inputs directly into the sequence-to-sequence model to be able to train end-to-end the whole system. Additionally, we utilize a step encoder, which serves as a memory to encode the most useful K facts, to serve as a prior into the neural planner system, such that it maintains the context of any previous steps.

3.3 Dataset

To train and eventually test our system, we scraped upwards of 93000 procedures from WikiHow(27), which are Creative Commons licensed and useable in research, along with their sub-procedures, requirements, warnings, among other information. We are currently working on cleaning up this dataset to release it in future work.

4 Limitations and Future Work

Our current system is under development, but early tests have shown that the system is indeed capable of transforming the contextual knowledge graph into procedural step text, but we have yet to quantify how well the system performs and are working on it at the time of writing. Some other limitations that have come up during development and are being addressed at the time of this submission are the following. There is a computational limitation on utilizing the RGCN, such that it requires training embeddings for every entity in the generated knowledge graphs. Further testing is required to determine whether the RGCN embeddings perform better than simple sentence embeddings. Additionally, the contextual graph generation is a computationally complex process because it requires performing beam search language generation for thousands of inputs per procedure. We are currently working on optimizing the system to work in parallel for every procedure. Another limitation that is not currently addressed, is that of exception handling in procedures. As we mentioned previously, by abstracting a procedure into a set of knowledge graphs, we could in theory add in the knowledge graph of an exception solving sub-procedure as described before, but we have yet to test this behavior. We plan on utilizing this system as part of an open-domain conversational agent capable of guiding people in procedures.

[2]ProPara is a dataset designed to train and test comprehension of simple paragraphs describing processes for the task of predicting, tracking, and answering questions about how entities change during the process(23)

References

[1] M. De Rijke *et al.*, "Question answering: What's next?" 2005.

[2] E. Eiriksdottir and R. Catrambone, "Procedural instructions, principles, and examples: How to structure instructions for procedural tasks to enhance performance, learning, and transfer," *Human factors*, vol. 53, no. 6, pp. 749–770, 2011.

[3] A. Gupta, A. Khosla, G. Singh, and G. Dasgupta, "Mining procedures from technical support documents," *arXiv preprint arXiv:1805.09780*, 2018.

[4] A. Maitra, S. Garg, and S. Sengupta, "Enabling interactive answering of procedural questions," in *International Conference on Applications of Natural Language to Information Systems*. Springer, 2020, pp. 73–81.

[5] X. Wang, S. Ong, and A. Y.-C. Nee, "Multi-modal augmented-reality assembly guidance based on bare-hand interface," *Advanced Engineering Informatics*, vol. 30, no. 3, pp. 406–421, 2016.

[6] G. Westerfield, A. Mitrovic, and M. Billinghurst, "Intelligent augmented reality training for motherboard assembly," *International Journal of Artificial Intelligence in Education*, vol. 25, no. 1, pp. 157–172, 2015.

[7] L. Zhang, "Reasoning about procedures with natural language processing: A tutorial," *arXiv preprint arXiv:2205.07455*, 2022.

[8] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "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*. Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880. [Online]. Available: https://aclanthology.org/2020.acl-main.703

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[10] C. Zheng and P. Kordjamshidi, "Relevant CommonSense subgraphs for "what if..." procedural reasoning," in *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 1927–1933. [Online]. Available: https://aclanthology.org/2022.findings-acl.152

[11] Z. Chen, Y. Wang, B. Zhao, J. Cheng, X. Zhao, and Z. Duan, "Knowledge graph completion: A review," *IEEE Access*, vol. 8, pp. 192 435–192 456, 2020.

[12] R. J. Moore, *A Natural Conversation Framework for Conversational UX Design*. Cham: Springer International Publishing, 2018, pp. 181–204. [Online]. Available: https://doi.org/10.1007/978-3-319-95579-7_9

[13] B. Dalvi, N. Tandon, A. Bosselut, W.-t. Yih, and P. Clark, "Everything happens for a reason: Discovering the purpose of actions in procedural text," 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)*, 2019, pp. 4496–4505.

[14] A. Amini, A. Bosselut, B. D. Mishra, Y. Choi, and H. Hajishirzi, "Procedural reading comprehension with attribute-aware context flow," *arXiv preprint arXiv:2003.13878*, 2020.

[15] S. Gabriel, C. Bhagavatula, V. Shwartz, R. Le Bras, M. Forbes, and Y. Choi, "Paragraph-level commonsense transformers with recurrent memory," in *AAAI*, 2021.

[16] J. D. Hwang, C. Bhagavatula, R. L. Bras, J. Da, K. Sakaguchi, A. Bosselut, and Y. Choi, "Comet-atomic 2020: On symbolic and neural commonsense knowledge graphs," in *AAAI*, 2021.

[17] R. Speer, J. Chin, and C. Havasi, "Conceptnet 5.5: An open multilingual graph of general knowledge," *arXiv preprint arXiv:1612.03975*, 2016.

[18] N. Mostafazadeh, A. Kalyanpur, L. Moon, D. Buchanan, L. Berkowitz, O. Biran, and J. Chu-Carroll, “Glucose: Generalized and contextualized story explanations,” *arXiv preprint arXiv:2009.07758*, 2020.

[19] X. L. Li and P. Liang, “Prefix-tuning: Optimizing continuous prompts for generation,” *arXiv preprint arXiv:2101.00190*, 2021.

[20] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, “Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing,” *arXiv preprint arXiv:2107.13586*, 2021.

[21] H. Liu and P. Singh, “Conceptnet—a practical commonsense reasoning tool-kit,” *BT technology journal*, vol. 22, no. 4, pp. 211–226, 2004.

[22] N. Tandon, B. D. Mishra, J. Grus, W.-t. Yih, A. Bosselut, and P. Clark, “Reasoning about actions and state changes by injecting commonsense knowledge,” *arXiv preprint arXiv:1808.10012*, 2018.

[23] B. D. Mishra, L. Huang, N. Tandon, W.-t. Yih, and P. Clark, “Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension,” *arXiv preprint arXiv:1805.06975*, 2018.

[24] D. Ribeiro, T. Hinrichs, M. Crouse, K. Forbus, M. Chang, and M. Witbrock, “Predicting state changes in procedural text using analogical question answering,” in *7th Annual Conference on Advances in Cognitive Systems*, 2019.

[25] N. Reimers and I. Gurevych, “Making monolingual sentence embeddings multilingual using knowledge distillation,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2020. [Online]. Available: https://arxiv.org/abs/2004.09813

[26] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling, “Modeling relational data with graph convolutional networks,” in *European semantic web conference*. Springer, 2018, pp. 593–607.

[27] “Wikihow.” [Online]. Available: https://www.wikihow.com/Main-Page