Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Leveraging Inter-step Dependencies for Information Extraction from Procedural Task Instructions

Nima Nabizadeh, Heiko Wersing, Dorothea Kolossa

2021

Preprint:

This is a post-peer-review, pre-copyedit version of an article published in Text, Speech, and Dialogue. The final authenticated version is available online at: https://doi.org/ 10.1007/978-3-030-83527-9

Leveraging Inter-step Dependencies for Information Extraction from Procedural Task Instructions

Nima Nabizadeh[1], Heiko Wersing[2], and Dorothea Kolossa[1]

[1] Cognitive Signal Processing Group, Ruhr University Bochum, Germany
{nima.nabizadeh, dorothea.kolossa}@rub.de
[2] Honda Research Institute Europe GmbH, Germany
heiko.wersing@honda-ri.de

Abstract. Written instructions are among the most prevalent means of transferring procedural knowledge. Hence, enabling computers to obtain information from textual instructions is crucial for future AI agents. Extracting information from a step of a multi-part instruction is usually performed by solely considering the semantic and syntactic information of the step itself. In procedural task instructions, however, there is a sequential dependency across entities throughout the entire task, which would be of value for optimal information extraction. However, conventional language models such as transformers have difficulties processing long text, i.e., the entire instruction text from the first step to the last one, since their scope of attention is limited to a relatively short chunk of text. As a result, the dependencies among the steps of a longer procedure are often overlooked. This paper suggests a BERT-GRU model for leveraging sequential dependencies among all steps in a procedure. We present experiments on annotated datasets of text instructions in two different domains, i.e., repairing electronics and cooking, showing our model's advantage compared to standard transformer models. Moreover, we employ a sequence prediction model to show the correlation between the predictability of tags and the performance benefit achieved by leveraging inter-step dependencies.

Keywords: Information Extraction, Procedural Task, Instructional Text, Long-term Dependencies

1 Introduction

Procedural task instructions describe how to perform various tasks to accomplish the desired goals, usually by spelling out the procedure step by step. An instruction is comprised of a list of inter-related sub-tasks, i.e., steps, for achieving a goal that can be cooking a particular dish for dinner or repairing a broken device. Information Extraction (IE) from written instructions has gained more attention in recent years since the number of free instructions on the internet has increased remarkably, providing a wealth of knowledge in various domains. Moreover, AI systems advanced rapidly in their capabilities, tackling ever more challenging tasks. As a result, equipping ML systems with the ability to interpret instructions originally written for human recipients, and extracting their practical knowledge, would confer immense capabilities and benefits.

The optimal representation structure for this knowledge depends on the application and the intended purpose of the extracted information. In robotics, one popular type of knowledge is commonsense, which is often depicted as relation-entity triples. Commonsense knowledge might contain general facts about objects and actions (*e.g., knives are used for cutting*) [12], object models for grasping and recognition [29, 13], or context knowledge about human environments, relating objects to their potential locations [26, 11]. The task of commonsense knowledge extraction from text usually begins with tagging the words in particular sentences with corresponding labels for specifying the elements of relation-entity triples among the entities [18, 9].

Apart from sentence-level information extraction, researchers also pursued the extraction of major workflow from procedures [20, 19, 1, 36, 37]. Extracting the workflow from instructions requires identifying specific entities in the text, for example by word labeling, and then capturing the temporal relationships among those entities, often utilizing graphical models. In both types of tasks, including sentence-level information extraction and extracting the workflows, labeling the words in the text, i.e., token classification, appears to be an inseparable part of the information extraction. Therefore in this paper, we frame our goal of information extraction from instructions as a token-classification task, where various elements, e.g., tools, actions, and objects, need to be mapped to their respective token labels.

The process of token classification regularly begins with generating a high-dimensional representation from the text at each timestep, e.g., each word token, and then classifying the produced embeddings over a set of labels. The transformer-based language models such as BERT [8] constitute a drastic leap forward in generating contextualized word embeddings. These models distinctly improved benchmarks in sequence tagging tasks, such as named entity recognition [34], part-of-speech tagging [31], semantic role labeling [40], and many more. Such massive transformer models are often pre-trained on a considerable amount of data in an unsupervised fashion and can be fine-tuned to various downstream tasks, requiring comparably few labeled examples for information extraction [39].

Still, transformer-based models have difficulties processing long sequences since the self-attention mechanism's computation and memory requirement scales quadratically with the sequence length. Consequently, the span of input text in such models is often truncated to different partitions with a limited number of tokens (e.g., 512 tokens for BERT). As a result, these models cannot capture dependencies among the various chunks of the input text corresponding to the different task steps, and the partitioning could potentially result in the loss of crucial cross-partition information. Such cross-partition dependencies, e.g., among the objects used in the task's various steps, inherently exist in many instructions. For instance, authors in [21] showed that in repair manuals, the next required tool at each step is predictable by only looking at the previously used tools without even processing the text. This is because the order of tools used in a repair task corresponds to the arrangement of partially shared components among different devices of related categories. Their work, utilizing a Variable-order Markov model, revealed that the dependency among the used tools in a repair task could go beyond ten previous steps; however, leveraging the information from ten previous steps often requires processing text with more than 512 tokens. Similar sequential dependen-

cies exist in cooking tasks: special ingredients are usually used in specific orders and with particular kitchenwares that are sometimes predictable by looking at the temporal order of steps. For example, the authors in [32] utilized this property of cooking objects. By training *bi-* and *tri*-gram predictors on cooking recipes, they improved the model's performance in interpreting referring expressions to the kitchenware.

Towards this aim, we simply combine a pre-trained BERT model with a bidirectional recurrent neural network, allowing us to leverage the inter-step information for token classification and consequently, for information extraction. In the following sections, after reviewing the related work in Section 2, we introduce the model in Section 3. In Section 4 we present the details of the datasets, the experiments and the results. Finally, conclusions are drawn, both on our achieved results and on future work, in Section 5.

2 Related Work

For the task of information extraction from how-to instructions, one generation of the methods utilized various types of pattern-matching techniques, often by designing domain-specific rules [16, 27]. For example, in [12] general dependency parsers were adopted to extract a structure from each sentence and then mapped these structures to the hand-crafted rules for inferring each token's role in the text. In light of recent advancements in deep neural networks for text processing, pattern-matching techniques were substituted by deep text processing models in later works.

Some researchers approached the interpretation of how-to procedures as a relation classification task. In [23] authors suggested a hierarchical bi-directional LSTM for predicting relations such as *is_next_of* or *is_subtask_of* between two steps in wikihow instructions. Their suggested two-level architecture, i.e., word- and step-level, outperformed comparable models based on convolutional neural networks. On similar data, [41] stacked three LSTM networks to encode task titles, step gist (a brief and concise summary of the step), and step explanations for predicting steps' relevance and ordering. Our method resembles these works in the sense that we also produce step-level representations along with the word-level ones. However, their work does not cover word labeling.

Since the introduction of the transformer-based language models, researchers have been fine-tuning pre-trained BERT or its offspring to information extraction tasks [28]. In procedural task instructions, researchers in [18] used BERT to extract commonsense triples from cooking recipes by labeling the words as tools, associated action, and object. Still, this work did not consider the long-term dependencies among different steps of their task and their corresponding entities. In [6, 30] a BERT-BiLSTM-CRF model is presented for relation extraction and named entity recognition. The BiLSTM-CRF layer classifies BERT's embeddings over the word labels, leveraging information among different word tokens but not among the different steps of the task.

Another research track aims to reduce the quadratic memory and computational requirements of self-attention mechanism in transformers, using some form of sparse attention pattern [15, 4, 3]. Among these works, Longformer [3] improved the state-of-the-art in long-document processing and also allows fine-tuning a pre-trained model.

Their approach employs attention patterns that combine local and global information, scaling linearly with the sequence length. The most advanced type of attention pattern of Longformer, i.e., global + sliding window, benefits from local attention on each word's immediate neighbors and the global attention on a few pre-selected tokens, e.g., the special classification token [CLS] in BERT. Although Longformer reduces the memory requirement and allows the model to process larger chunks of text, it still has a relatively high complexity of $O(nz)$, where n is the input sequence length, and z is the sliding window size. In our paper, we additionally compare our result with the Longformer's performance. One important characteristic of information extraction from instructions is that there are distinct boundaries in the text, i.e., the beginning and the end of each manual, inside which the information needs to be leveraged for the token classification task. However, in common language modeling, such boundaries do not exist in the text, and the long-document processing models attempt to capture as much context as possible.

3 Model

This section describes the proposed method for modeling long-term dependencies among the entities in various steps of procedural task instructions. Our proposed model comprises four general parts: 1- Generating token-level representations, 2- Obtaining step-level representations, 3- Leveraging the obtained step-level information, and 4- Classifying the final representations.

3.1 Token-level representation

The first step of the token classification task is to produce a contextualized embedding for each token by processing the chunk of text that the token exists in it. We used a pre-trained BERT base-cased model for this task. In our experiments, we also tried the Roberta base model [17]; however, it did not give a better result than BERT.

3.2 Step-level representation

Leveraging the inter-step dependencies becomes easier in a hierarchical setting by producing step-level representations for the chunks of the token-level representations processed by BERT. We tried multiple approaches for condensing a group of token-level information into a single embedding representing a step, including averaging and min- and max-pooling of the word embeddings. In another experiment, we also used BERT's special sentence classification token, [CLS], for the step-level representation. Since min-pooling yielded a slightly better result, we did not consider the other methods in the later evaluations. Formally, if $\mathbf{v}_i = \mathrm{emb}(\mathrm{w_{ij}})$ is BERT's embedding of the ith word token in the jth step of the procedure, the pooled representation of the jth step ($\boldsymbol{s_j}$) becomes

$$s_{j,d} = \min_{i \in [1,T]} v_{i,d}, \tag{1}$$

where minima are computed per dimension d and T is number of tokens in step j.

3.3 Leveraging step-level information

After generating the step-level representations, we need to determine how much information from the past and future steps should be involved in the current step's token classification. To this aim, the step-level embeddings are passed through a bi-directional recurrent layer with Gated Recurrent cells (GRU) [5] that have 256 units in each direction. GRU cells store and filter the information from all steps using their update and reset gate, determining the amount of step-level information that is beneficial for the ultimate classification task.

$$\begin{aligned} \overrightarrow{h_j} &= \overrightarrow{\mathrm{GRU}}(\boldsymbol{s_j}), j \in [1, L] \\ \overleftarrow{h_j} &= \overleftarrow{\mathrm{GRU}}(\boldsymbol{s_j}), j \in [L, 1] \end{aligned} \tag{2}$$

In Equation 2, $\overrightarrow{h_j}$ is the forward hidden state and $\overleftarrow{h_j}$ is the backward hidden state of the GRU at step j, and L is the index of last step in the procedure.

3.4 Classifying the final representations.

In the last level, we concatenate the token-level representations with the hidden state of the GRU at each step. A fully connected (FC) layer of size 512 performs the final classification over a pre-defined set of labels. The probability distribution for all possible tags of the ith word token in the jth step (P_{ij}) would be derived by:

$$P_{ij} = \mathrm{FC}(\mathrm{emb}(w_{ij}) \oplus \overrightarrow{h_j} \oplus \overleftarrow{h_j}), \tag{3}$$

where $\oplus$ denotes concatenation.

Figure 1 shows the model architecture with two consecutive steps in a repair manual as examples.

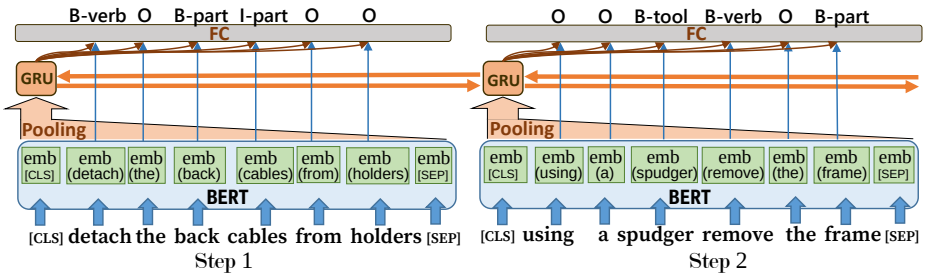

Fig. 1: The complete architecture of the proposed BERT-GRU model with two short, subsequent steps in a repair manual as examples.

4 Experiments and Results

In this section, we describe the datasets we used for evaluating the performance of the proposed model and we assess our results in comparison to the baselines.

4.1 Datasets

We evaluate the model of Section 3 on two datasets of instructional text: the MyFixit dataset [22], a collection of repair manuals for electronics, and the "recipe named entities" (r_NEs) from the English Recipe Flow Graph Corpus [37], containing annotated cooking recipes.

MyFixit dataset The first part of the evaluation is on the human-annotated part of the MyFixit dataset; a semi-structured dataset scraped from the iFixit website. In the MyFixit data, all manuals have been divided into their constituent steps, where at each step, the user typically disassembles one particular device component. The steps of manuals in the "Mac Laptop" category are annotated with the step's required tools, detached component, and the removal verbs, i.e., the verbs that indicate the acts of detachment from the device. These entities are annotated in IOB (Inside–Outside–Beginning) format. In total, 1,497 manuals with 36,973 steps have human annotations. The MyFixit data instructions are often longer than 512 word tokens, with an average size of 944 words per manual and a maximum manual length of 3509 words.

Recipe named entities dataset The r_NEs dataset was initially introduced in [35], and later extended by [37]. It contains 300 cooking recipes from the Allrecipes UK/Ireland website, which are annotated for recipe named entities. The annotation tags of this dataset include Food (F), Tool (T), Duration (D), Quantity (Q), Action by chef (Ac), Action by food (Af), Action by tool (At), Food state (Sf), and Tool state (St). The instructions in r_NEs are often shorter than those in the MyFixit dataset, with an average size of 132 words per recipe and a maximum size of 444 words.

4.2 Experiments

We implemented the proposed model in PyTorch using the Hugging Face Transformers library [33]. The maximum sequence length per training example for BERT-GRU was chosen as 40 word tokens, which is close to the average number of words at each step for both datasets. Surprisingly, increasing the sequence length did not improve the performance of our model. In contrast, for the baseline BERT model, the maximum sequence length was kept at 512 tokens. As another baseline, we also compare with the performance of the BiLSTM-CRF model [10] previously introduced in [22]. The BiLSTM-CRF receives a stack of FLAIR [2], and GloVe [24] word embeddings as input. For the Longformer, we employed the pretrained model with default hyperparameters, including the maximum sequence length that is 4096 word tokens. We used sliding windows plus global attention on the [CLS] token for the attention pattern.

For the evaluation, we did 10-fold cross-validation, randomly splitting the data into 80% training, 10% hyperparameter tuning (development set), and 10% testing. During training, only the parameters of the GRU and the fully connected layer are learned using the Adam optimizer [14]. The batch size for the BERT and BERT-GRU models is set to 10 instructions, and for the Longformer, the batch size is one. We report the average F1 score of labeling each tag in the test set as the evaluation measure.

Table 1 shows the F1 scores for word labeling in the MyFixit dataset. It can be seen that our model improves the performance for every tag in the MyFixit dataset. Previously, the classification of tokens that represent disassembled objects had yielded the poorest performance among all other tags for both BERT and BiLSTM-CRF models. The reason for this could be the widest variety of word tokens that are tagged with this label in the dataset (for a detailed analysis of dataset labels, please refer to [22].) It can be seen that the BERT-GRU model shows the most F1 score improvement for the disassembled objects.

Table 1: Average F1 score of word labeling for the MyFixit dataset.

Model	Tag			
	Tool	Removal verb	Disassembled object	Average
BiLSTM-CRF [10]	0.95	0.95	0.88	0.93
BERT [8]	0.95	0.97	0.90	0.94
Longformer [3]	0.96	0.97	0.96	0.96
BERT-GRU	**0.97**	**0.98**	**0.97**	**0.97**

Similarly, Table 2 shows the performance of BERT-GRU for labeling the r_NEs dataset. We did not include the Longformer in this evaluation, since the instructions' lengths in this dataset are less than BERT's maximum allowed length.

Table 2: The average F1 score of word labeling in recipe named entities (r_NEs) dataset.

Model	Tag										
	F	T	D	Q	Ac	Ac2	Af	At	Sf	St	Average
BiLSTM-CRF	0.91	0.88	0.90	0.75	0.91	0.47	0.37	0.25	0.65	0.82	0.69
BERT	0.92	0.89	0.89	0.74	0.93	0.45	0.50	0.17	**0.68**	0.82	0.70
BERT-GRU	0.92	**0.90**	**0.92**	**0.76**	**0.95**	**0.56**	0.50	**0.29**	0.67	**0.86**	**0.73**

From Table 2, we can observe that the BERT-GRU model again improves the average F1 score. The F1 score of most tags except for F (food), Af (action by food) and Sf (food state) is improved. We hypothesize that the BERT-GRU model enhances word labeling performance more for the entities that have more substantial sequential dependencies. In other words, there are some entities with particular tags, e.g., tools,

that are predictable better by looking at the previous and next steps. For examining this hypothesis, we developed a sequential model that reflects the predictability of various entities mentioned in the procedure, ignoring the words' semantics.

4.3 Analysis of Sequential Dependencies

The purpose of the model in this section is to illustrate the degree of predictability of different entities in a task, only by looking at the sequential dependencies among the entities in a procedure, but forgoing any additional text-processing. This work resembles event ordering [7], or script learning [25] tasks.

In this experiment, every word token within the entity spans, i.e., each token annotated with a tag except "O" is encoded as a one-hot vector in a vocabulary of entities. Let $\mathcal{V}$ denote the vocabulary of lemmatized entity names. A complete step is represented by a multi-hot vector that sums all entity vectors of the step. The model is trained to predict the probability of observing each entity from $\mathcal{V}$ in a step, based on the sequence of the entities seen in the preceding and following steps. Every instruction has a first step (Start) and a final step (End) that are added to all instructions.

Similar to the model in Section 3, the sequential model consists of a bidirectional GRU layer but with state size 128. The output layer has $|\mathcal{V}|$ neurons and a sigmoid activation function that estimates the probability of observing each entity in $\mathcal{V}$. The model takes the multi-hot vector of the next step as the ground truth during training, and its parameters are learned by minimizing the binary cross-entropy loss in Equation 4 using the Adam optimizer [14] with a learning rate of 0.001.

$$\mathrm{H}(y, \hat{y}) = -\frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} y_i \cdot \log(\hat{y_i}) + (1 - y_i) \cdot \log(1 - \hat{y_i}) \tag{4}$$

Here, $\hat{y_i}$ denotes the probability of the i-th entity in the model output and y_i is the corresponding target value. Figure 2 illustrates the unrolled graph of the sequential model used for analyzing the predictability of various entities.

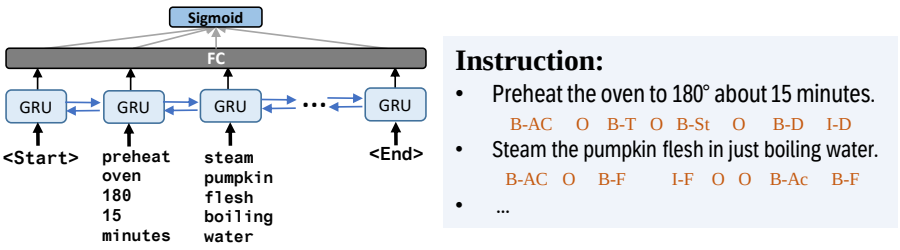

Fig. 2: Unrolled graph of the sequential model for analyzing the predictability of entities in procedural text.

For the inference, we sampled entities from the output distribution that have the probability ($\hat{y_i}$) above a certain threshold. For the evaluation, the datasets are split identical to Section 4.2, with the same portions and random seeds. The threshold value and

the training stop criteria are determined using the development set. As in Section 4.2, we look at the average F1 score of returning correct entities of each step as the evaluation measure.

Table 3 shows the results of this evaluation. Here, the second row shows the F1 score of the predictor model, and the last row shows the difference between the performance of baseline BERT and the BERT-GRU extractor models. It can be seen that the greatest improvement of the BERT-GRU model is achieved for the tags with the highest predictability. There is a strong correlation between the sequential dependencies and the BERT-GRU extraction improvement in the r_NEs dataset (Spearman rank correlation ρ = 0.81) and a moderate correlation in the MyFixit dataset (ρ = 0.5). This experiment shows that if sequential dependencies exist among entities in a task, the BERT-GRU model can leverage it for the token classification task. The correlation is better observable in the r_NEs dataset due to the larger verity of labels, while in MyFixit, the low number of label types reduces the power of the test.

Table 3: Comparison between the predictability of different tags and the performance benefit achieved by BERT-GRU over BERT.

A: Recipe named entities (r_NEs) dataset

Tags	F	T	D	Q	Ac	Ac2	Af	At	Sf	St
Predictability F1 score	0.55	0.63	0.66	0.33	0.63	0.74	0.43	0.80	0.44	0.65
Extraction Improvement	0.00	0.01	0.03	0.02	0.02	0.11	0.00	0.12	-0.01	0.04

B: MyFixit dataset

Tags	Tool	Removal verb	Disassembled object
Predictability F1 score	0.73	0.70	0.71
Extraction Improvement	0.02	0.01	0.07

5 Conclusion

We propose the BERT-GRU architecture for information extraction from procedural task instructions. The model combines the power of BERT for text processing with the capabilities of recurrent neural networks in sequence prediction. As a result, the GRU layer allows for leveraging long-term dependencies among the entities in text from the beginning to the end of each instruction and vice versa. Our experiments show that limiting the span of the self-attention mechanism to smaller chunks of text and using a GRU layer for transferring long-span dependencies improves the model performance compared to a pure transformer architecture. Our design requires less memory when

compared to the transformer-based models for long document processing and does not need a special CUDA kernel.

We additionally employ a sequence prediction model to predict the correct sequence of entities, e.g., the used tools, only by observing the entities mentioned in the surrounding steps and ignoring the full text. This experiment is used to verify our hypothesis that the extraction of entities with higher predictability benefits more from leveraging inter-step dependencies.

Our approach for constructing the step-level representation would be improved if the model could focus more on the specific tokens inside the entity spans that carry the essential information for the IE task. It is possible to partially filter out some words outside the entities and reduce the noise from steps' representations by part-of-speech tagging and syntactic parsing. Transformers with a random attention pattern, similar to a recently published work [38], may also provide the model the chance to attend more strongly to those specific tokens. We intend to incorporate these ideas in our model to make optimal use of any available contextual information while keeping the model efficient in size and computational effort.

References

1. Abend, O., Cohen, S.B., Steedman, M.: Lexical event ordering with an edge-factored model. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 1161–1171 (2015)
2. Akbik, A., Bergmann, T., Vollgraf, R.: Pooled contextualized embeddings for named entity recognition. In: NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics. p. 724–728 (2019)
3. Beltagy, I., Peters, M.E., Cohan, A.: Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150 (2020)
4. Child, R., Gray, S., Radford, A., Sutskever, I.: Generating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509 (2019)
5. Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. pp. 1724–1734. ACL (2014)
6. Dai, Z., Wang, X., Ni, P., Li, Y., Li, G., Bai, X.: Named entity recognition using bert bilstm crf for chinese electronic health records. In: 2019 12th international congress on image and signal processing, biomedical engineering and informatics (cisp-bmei). pp. 1–5. IEEE (2019)
7. Dalvi, B., Tandon, N., Bosselut, A., Yih, W.t., Clark, P.: 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). pp. 4486–4495 (2019)
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
9. Diwan, N., Batra, D., Bagler, G.: A named entity based approach to model recipes. In: 2020 IEEE 36th International Conference on Data Engineering Workshops (ICDEW). pp. 88–93. IEEE (2020)
10. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. CoRR abs/1508.01991 (2015), http://arxiv.org/abs/1508.01991

11. Jebbara, S., Basile, V., Cabrio, E., Cimiano, P.: Extracting common sense knowledge via triple ranking using supervised and unsupervised distributional models. Semantic Web 10(1), 139–158 (2019)
12. Kaiser, P., Lewis, M., Petrick, R.P., Asfour, T., Steedman, M.: Extracting common sense knowledge from text for robot planning. In: 2014 IEEE International Conference on Robotics and Automation (ICRA). pp. 3749–3756. IEEE (2014)
13. Kehoe, B., Matsukawa, A., Candido, S., Kuffner, J., Goldberg, K.: Cloud-based robot grasping with the google object recognition engine. In: 2013 IEEE International Conference on Robotics and Automation. pp. 4263–4270. IEEE (2013)
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR (2015)
15. Kitaev, N., Kaiser, Ł., Levskaya, A.: Reformer: The efficient transformer. arXiv preprint arXiv:2001.04451 (2020)
16. Lau, T.A., Drews, C., Nichols, J.: Interpreting written how-to instructions. In: IJCAI. pp. 1433–1438 (2009)
17. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
18. Losing, V., Fischer, L., Deigmoeller, J.: Extraction of common-sense relations from procedural task instructions using bert. In: Proceedings of the 11th Global Wordnet Conference. pp. 81–90 (2021)
19. Maeta, H., Sasada, T., Mori, S.: A framework for procedural text understanding. In: Proceedings of the 14th International Conference on Parsing Technologies. pp. 50–60 (2015)
20. Malmaud, J., Wagner, E., Chang, N., Murphy, K.: Cooking with semantics. In: Proceedings of the ACL 2014 Workshop on Semantic Parsing. pp. 33–38 (2014)
21. Nabizadeh, N., Heckmann, M., Kolossa, D.: Target-aware prediction of tool usage in sequential repair tasks. In: Proceedings of The Sixth International Conference on Machine Learning, Optimization, and Data Science. pp. 869–880. Springer (2020)
22. Nima Nabizadeh, Dorothea Kolossa, M.H.: Myfixit: An annotated dataset, annotation tool, and baseline methods for information extraction from repair manuals. In: Proceedings of Twelfth International Conference on Language Resources and Evaluation (2020)
23. Park, H., Motahari Nezhad, H.R.: Learning procedures from text: Codifying how-to procedures in deep neural networks. In: Companion Proceedings of the The Web Conference 2018. pp. 351–358 (2018)
24. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014)
25. Pichotta, K., Mooney, R.: Learning statistical scripts with lstm recurrent neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 30 (2016)
26. Samadi, M., Kollar, T., Veloso, M.: Using the web to interactively learn to find objects. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 26 (2012)
27. Schumacher, P., Minor, M., Walter, K., Bergmann, R.: Extraction of procedural knowledge from the web: A comparison of two workflow extraction approaches. In: Proceedings of the 21st International Conference on World Wide Web. pp. 739–747 (2012)
28. Shi, P., Lin, J.: Simple bert models for relation extraction and semantic role labeling. arXiv preprint arXiv:1904.05255 (2019)
29. Tenorth, M., Klank, U., Pangercic, D., Beetz, M.: Web-enabled robots. IEEE Robotics & Automation Magazine 18(2), 58–68 (2011)
30. Tian, C., Zhao, Y., Ren, L.: A chinese event relation extraction model based on bert. In: 2019 2nd International Conference on Artificial Intelligence and Big Data (ICAIBD). pp. 271–276. IEEE (2019)

31. Tsai, H., Riesa, J., Johnson, M., Arivazhagan, N., Li, X., Archer, A.: Small and practical bert models for sequence labeling. arXiv preprint arXiv:1909.00100 (2019)
32. Whitney, D., Eldon, M., Oberlin, J., Tellex, S.: Interpreting multimodal referring expressions in real time. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). pp. 3331–3338. IEEE (2016)
33. Wolf, T., Chaumond, J., Debut, L., Sanh, V., Delangue, C., Moi, A., Cistac, P., Funtowicz, M., Davison, J., Shleifer, S., et al.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 38–45 (2020)
34. Yamada, I., Asai, A., Shindo, H., Takeda, H., Matsumoto, Y.: Luke: deep contextualized entity representations with entity-aware self-attention. arXiv preprint arXiv:2010.01057 (2020)
35. Yamakata, Y., Carroll, J., Mori, S.: A comparison of cooking recipe named entities between japanese and english. In: Proceedings of the 9th Workshop on Multimedia for Cooking and Eating Activities in conjunction with The 2017 International Joint Conference on Artificial Intelligence. pp. 7–12 (2017)
36. Yamakata, Y., Imahori, S., Maeta, H., Mori, S.: A method for extracting major workflow composed of ingredients, tools, and actions from cooking procedural text. In: 2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW). pp. 1–6. IEEE (2016)
37. Yamakata, Y., Mori, S., Carroll, J.A.: English recipe flow graph corpus. In: Proceedings of The 12th Language Resources and Evaluation Conference. pp. 5187–5194 (2020)
38. Zaheer, M., Guruganesh, G., Dubey, A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al.: Big bird: Transformers for longer sequences. arXiv preprint arXiv:2007.14062 (2020)
39. Zhang, R., Yang, W., Lin, L., Tu, Z., Xie, Y., Fu, Z., Xie, Y., Tan, L., Xiong, K., Lin, J.: Rapid adaptation of bert for information extraction on domain-specific business documents. arXiv preprint arXiv:2002.01861 (2020)
40. Zhang, Y., Wang, R., Si, L.: Syntax-enhanced self-attention-based semantic role labeling. arXiv preprint arXiv:1910.11204 (2019)
41. Zhou, Y., Shah, J.A., Schockaert, S.: Learning household task knowledge from wikihow descriptions. arXiv preprint arXiv:1909.06414 (2019)