GPT3-to-plan: Extracting plans from text using GPT-3

Alberto Olmo, Sarath Sreedharan, Subbarao Kambhampati

Arizona State University

{aolmoher, ssreedh3, rao}@asu.edu

Abstract

There is an ongoing interest for the performance assessment of generalized systems, these systems are known for having enough plasticity to perform well on different never-before-seen specialized tasks. The trend, is all the more apparent in Natural Language Processing tasks. In light of the advancements in generalized systems, in this paper we show the applicability of the state of the art language model for plan extraction tasks. We want to investigate how well GPT-3 a state-of-the-art language model compares to the the some of the latest plan extraction techniques with just a few-shot training examples. To the best of our knowledge, this paper is the first to empirically evaluate GPT-3 (Brown, Mann, and et al. 2020) on its performance on plan extraction tasks.

Introduction

Following sequential procedures and plans undergird many aspects of our everyday lives. As we look at many vital and consequential industries, including finance and banking, the ability to identify the correct procedures and adhere to them perfectly, becomes essential. So it is of no surprise that many enterprises invest heavily in accurately documenting these workflows in forms that are easy for their employees to follow. As we start automating many of these day-to-day activities, it becomes important that our automated systems are also able to pick up and execute them. Unfortunately, having these procedures documented is not the same as them being easy and readily available for an AI system to use. Additionally, in many of these high-risk domains, the agent cannot just try to figure out these procedures on their own through trial and error. Instead, we would want to develop ways wherein we can convert these procedures designed for human consumption to easier forms for agents to use. Within the planning community, there has been a lot of recent interest in developing *plan extraction* methods that are able to take natural language text describing a sequential plan. Some of the more recent works in this direction include, works like Feng, Zhuo, and Kambhampati (2018); Daniele, Bansal, and Walter (2017), which have proposed specialized frameworks for performing sequence-to-sequence transla-

tion that maps natural language sentences into structured plans.

On the other hand, the mainstream Natural Language Processing (NLP) has started shifting its focus from more specialized translation methodologies to developing general purpose models such as transformer networks (Radford et al. 2019; Brown, Mann, and et al. 2020). These networks have already shown very encouraging results in many tasks and proven their ability to generalize to unseen ones. These are task-agnostic language models trained on large general web corpora and have shown to be comparable (and in some cases better than) their state-of-art task-specific counterparts. Examples of some tasks these models have been tested on includes, question-answering, translation, on-the-fly reasoning and even generation of news articles that are arguably indistinguishable from human-written ones. In light of these advancements, we try to answer the following question: *to what extent can the current state-of-art in general natural language models compete against task-specific action sequences extractors?* These papers have generally looked at employing task-specific learning based methods that expect access to pre-processed/task-specific data, including annotations that allow mapping of text to the required structured output.

In this paper, we want to investigate how GPT-3 (Brown, Mann, and et al. 2020), one of the most recent transformer-based language models, can be used to extract structured actions from natural language texts. We find that these models achieve comparable, and in some cases better scores than previous state-of-the-art task specific methods. We make use of natural language text from three domains and measure the performance of the model in terms of its F_1 score, a commonly used quantitative measure for the task. We then compare it to previously published results for task-specific action extractors which use a varied range of solutions, including, reinforcement learning, (Feng, Zhuo, and Kambhampati 2018), sequence-to-sequence models (Daniele, Bansal, and Walter 2017), Bi-directional LSTMs (Ma and Hovy 2016) or clustering of action templates (Lindsay et al. 2017). Up front it is not obvious whether a general purpose LLM such as GPT3 will be able to compete with special purpose translators. We thought that the few shot learning ability of GPT3 might help in this regard, and decided to investigate. As we shall see, our suspicion wound up being valid–GPT3 does

indeed compete quite well with special purpose translators. Also, note that all results reported here are directly calculated from the best GPT-3 raw predictions, with no additional filtering or reasoning employed atop of it. We expect most of the results reported here to improve should we additionally exploit domain-level or task-level insights to filter the results from these models.

Background and Related Works

The Generative Pre-trained Transformer 3 (GPT-3) (Brown, Mann, and et al. 2020) is the latest version of the GPT models developed by OpenAI[1]. A 175 billion parameter autoregressive language model with 96 layers trained on a 560GB+ web corpora (Common Crawl[2] and WebText2 (Gokaslan and Cohen 2019)), internet-based book corpora and Wikipedia datasets each with different weightings in the training mix and billions of tokens or words. Tested on several unrelated natural language tasks, GPT-3 has proven successful in generalizing to them with just a few examples (zero in some cases). GPT-3 comes in 4 versions, Davinci, Curie, Babbage and Ada which differ in the amount of trainable parameters – 175, 13, 6.7 and 2.7 billion respectively (Brown, Mann, and et al. 2020). Previous work on action sequence extraction from descriptions has revolved around specific models for action extraction, some of them trained on largely task-specific preprocessed data. (Mei, Bansal, and Walter 2016; Daniele, Bansal, and Walter 2017) use sequence-to-sequence models and inverse reinforcement learning to generate instructions from natural language corpora. Similarly, Feng, Zhuo, and Kambhampati (2018) uses a reinforcement learning model to extract word actions directly from free text (i.e. the set of possible actions is not provided in advance) where, within the RL framework, actions select or eliminate words in the text and states represent the text associated with them. This allows them to learn the policy of extracting actions and plans from labeled text. In a same fashion, Branavan et al. (2009) also use Reinforcement Learning, a policy gradient algorithm and a log-linear model to predict, construct and ultimately learn the sequence of actions from text. Other works like Addis and Borrajo (2010) define a system of tools through which they crawl, extract and denoise data from plan-rich websites and parse their actions and respective arguments with statistical correlation tools to acquire domain knowledge.

However, to the best of our knowledge this paper is the first work to assess the performance of a general purpose NLP language model on action sequence extraction tasks compared to its current state-of-art task-specific counterpart.

Experiments

Datasets and GPT-3 API   We use the three most common datasets for action sequence extraction tasks used in evaluating many of the previous task-specific approaches, including Feng, Zhuo, and Kambhampati (2018) or Miglani and Yorke-Smith (2020). Namely, the "Microsoft Windows

	WHS	CT	WHG
Labeled texts	154	116	150
Input-output pairs	1.5K	134K	34M
Action name rate (%)	19.47	10.37	7.61
Action argument rate (%)	15.45	7.44	6.30
Unlabeled texts	0	0	80

Table 1: Characteristics of the datasets used.

Length	Temp.	Top P	Freq.	Pres.	Best of
100	0.0	1	0.0	0.0	1

Table 2: GPT-3 parameters used for all our experiments.

Help and Support" (WHS), the "WikiHow Home and Garden" (WHG) and the "CookingTutorial" (CT) datasets. The characteristics of these datasets are provided in Table 1.

The GPT-3 model is currently hosted online[3] and can be accessed via paid user queries with either their API or website in real time. Some example use cases of their service include keyword extraction from natural text, mood extraction from reviews, open-ended chat conversations and even text to SQL and JavaScript to Python converters amongst many others. In general, the service takes free natural language as input and the user is expected to encode the type of interaction/output desired in the input query. The system then generates output as a completion of the provided query. The API also allows the user to further tweak the output by manipulating the following parameters: `Max Tokens` sets the maximum number of words that the model will generate as a response, `Temperature` (between 0 and 1) allows the user to control the randomness (with 0 forcing the system to generate output with the highest probability consistently and rendering it effectively deterministic for a given input). `Top P` also controls diversity; closer to 1 ensures more determinism, `Frequency Penalty` and `Presence Penalty` penalize newly generated words based on their existing frequency so far, and `Best of` is the number of multiple completions to compute in parallel. It outputs only the best according to the model. In Table 2 we show the values that we used for all our experiments to ensure the most consistency in the model's responses.

Query generation   Each query consists of a few shot training in natural language text and the corresponding structured representation of the plan. For each example, we annotate the beginning of the natural language text portion with the tag `TEXT` followed by the plan (annotated with the tag `ACTIONS`). In the structure representation, each action is represented in a functional notation of the form $a_0^j(arg_0^0, arg_1^0 \ldots arg_k^0) \ldots a_n^j(arg_0^n, arg_1^n \ldots arg_k^n)$ where a_i^j represents action i in sentence j and arg_k^n is the *kth* argument from action a_n in the text. After the training pairs, we include the test sample in natural language text after another tag `TEXT` and then we add a final tag `ACTIONS`, with the

expectation that GPT3 will generate the corresponding plan representation after that.

Evaluation and Metrics In order to directly compare the performance of GPT-3 to Miglani and Yorke-Smith (2020), the current state-of-art, we followed a translation scheme with three types of actions, namely, *essential* (essential action and its corresponding arguments should be included in the plan) *exclusive* (the plan must only contain one of the exclusive actions) and *optional* actions (the action may or may not be part of the plan). We use this scheme to generate both the example data points provided to the system and to calculate the final metrics.

In particular, we will use *precision*, *recall* and *F1*, similar to Feng, Zhuo, and Kambhampati (2018); Miglani and Yorke-Smith (2020) to measure the effectiveness of the method.

$$Precision = \frac{\#TotalRight}{\#TotalTagged}, \ Recall = \frac{\#TotalRight}{\#TotalTruth}$$
$$F_1 = \frac{2 \times precision \times recall}{precision + recall}$$
(1)

Note that the ground truth number and the number of true extracted actions depend on the type that each action in the text corresponds to. For example, a set of exclusive actions only contribute one action to *#TotalTruth* and we only count an extracted exclusive action in *#TotalRight*, if and only if, one of the exclusive actions is extracted. Both *essential* and *optional* actions only contribute once to *#TotalTruth* and *#TotalRight*.

Baselines In Table 3 we compare GPT-3 to several action sequence extractor models:

- EAD: Mei, Bansal, and Walter (2016) design an Encoder-Aligner-Decoder method that uses a neural sequence-to-sequence model to translate natural language instructions into action sequences.

- BLCC: The Bi-directional LSTM-CNN-CRF model from Ma and Hovy (2016) benefits from both word and character-level semantics and implement an end-to-end system that can be applied to action sequence extraction tasks with pre-trained word embeddings.

- Stanford CoreNLP: in Lindsay et al. (2017) they reduce Natural Language texts to action templates and based on their functional similarity, cluster them and induce their PDDL domain using a model acquisition tool.

- EASDRL and cEASDRL: Feng, Zhuo, and Kambhampati (2018) and Miglani and Yorke-Smith (2020) use similar reinforcement learning approaches; they define two Deep Q-Networks which perform the actions of *selecting* or *rejecting* a word. The first DQN handles the extraction of Essential, Exclusive and Optional actions while the second uses them to select and extract relevant arguments.

The corresponding *precision*, *recall* and *F1* scores for each method were picked directly from their respective papers.

Model	Action names			Action arguments		
	WHS	CT	WHG	WHS	CT	WHG
EAD	86.25	64.74	53.49	57.71	51.77	37.70
CMLP	83.15	83.00	67.36	47.29	34.14	32.54
BLCC	90.16	80.50	69.46	93.30	**76.33**	70.32
STFC	62.66	67.39	62.75	38.79	43.31	42.75
EASDRL	93.46	84.18	75.40	**95.07**	74.80	75.02
cEASDRL	**97.32**	**89.18**	**82.59**	92.78	75.81	**76.99**
GPT-3						
Davinci	**86.32**	**58.14**	**43.36**	22.90	**29.63**	**22.25**
Curie	75.80	35.57	22.41	**31.75**	22.16	13.79
Babbage	62.59	20.62	14.95	22.91	12.59	7.33
Ada	60.68	14.68	8.90	17.91	4.13	2.27

Table 3: F_1 scores for all actions and their arguments across the WHS, CT and WHG datasets for the state-of-art sequence extraction models and GPT-3. State-of-art task-specific model F_1 scores are extracted from Miglani and Yorke-Smith (2020); Feng, Zhuo, and Kambhampati (2018) and represent their best possible recorded performance.
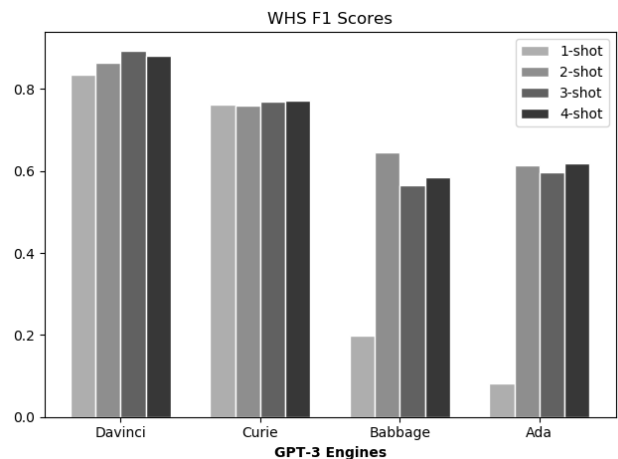

Figure 1: F_1 scores of the model on the Windows Help and Support dataset for 1 to 4 few-shot training

Results Given that GPT-3 is a few-shot learner we want to know how it performs given different amounts of training samples. To measure this, we query the language model with increasing numbers of examples (with a maximum of four examples) for all domains and report their F1 scores. We stop at the four-shot mark as the total amount of *tokens* or words that the request can contain is 2048. Additionally for the CookingTutorial and Wikihow Garden and Home datasets, 4-shot training examples already exceed this threshold, so we limit the length of input text to 10 sentences per training example. Specifically, we select the training examples as 1-shot (one datapoint is selected at random from the dataset), 2-shot (the two datapoints with the largest proportion of *optional* and *exclusive* actions from the dataset are selected), 3-shot (the three datapoints with the largest proportion of *optional*, *exclusive* and *essential* actions) and 4-shot (an additional random datapoint is added to 3-shot).

In Figure 1 we show how the F_1 score changes given 1, 2, 3 and 4-shot training samples when tested on the whole Windows Help and Support dataset. Unsurprisingly, Davinci, the model with the most amount of trainable parameters, performs best with over 80% F_1 score for each category. Both Davinci and Curie show the tendency to perform better the more examples they are given peaking at 3 and 4-shots respectively. Similarly, Babbage and Ada show their peaks given 2 and 4 examples while underperforming at one-shot training. This is unsurprising, given the fact that these models are simplified versions of GPT-3 which have also been trained on a smaller corpus of data for higher speed. Hence, they need more than one example to grasp the task.

In table 3 we compare the F_1 scores for action name and their argument extractions as reported by previous and current state of the art task-specific action sequence extractors against all GPT-3 engines: Davinci, Curie, Babbage and Ada, ordered from most to least powerful The scores are calculated based on 1 and account for *essential*, *exclusive* and *optional* actions and their respective arguments. All GPT-3 models are trained with two-shot examples. As expected, Davinci overall performs the best compared to the rest of engines. We can see that Davinci also outperforms the EAD, CMLP and STFC task-specific models for the Windows Help and Support domain on extracting actions. Even though it underperforms on the argument extraction task compared to the state of art, it's worth nothing that still obtains better than random extraction scoring.

Ordering   We want to assess whether GPT-3 is capable of inferring plan order from text. This is a feature which is mostly missing in previous task-specific state of the art like Feng, Zhuo, and Kambhampati (2018) or Miglani and Yorke-Smith (2020). As a preliminary evaluation, we create three examples (one for each dataset, shown in Figure 2), where order of the plans does not match how actions are listed in the text. In the Windows Help and Support example, we state on the second and third sentences that action *click(advanced)* must be performed eventually but only after *click(internet, options)*, and, even though the corresponding sentences appear in the opposite order, GPT-3 places them as expected. Similarly, in the CookingTutorial example, we state that *first* we need to *measure the quantity of oats* and *cook them* only later and once again, it generates the actions in correct ordering. For the last example, GPT-3 shows to understand that action *paint(walls)* has to be done before *remove(furniture)* and, interestingly, even though *decorate(floor)* is stated on the first sentence, the model seems to understand that it can be performed *anytime* and places the action last. Note that these are just anecdotal evidences and we would need to perform studies over larger test sets to further evaluate GPT-3's ability to identify the ordering of plans. Our current evaluation along this dimension is limited by the lack of annotation regarding the ordering in the currently available datasets and one of our future works would be to create/identify such text-to-plan dataset with additional annotations on action ordering.

Figure 2: Query examples on WHS, CT and WHG. Each query was input to Davinci along with two preceding training instances containing the largest proportion of *optional* and *exclusive* actions. The output is shown in regular text while the input is displayed in bold.

Discussion and Conclusion

In this paper we have shown that GPT-3, the state-of-art general purpose Natural Language model, can compete against task-specific approaches in the action sequence extraction domain, getting closer than ever to surpassing their performance. From the user's perspective, these transformer models pose the advantage of needing almost negligible computational resources from the user side by being readily available at just one query away and seem like a possible solution in the future to many natural language tasks should they keep up with their rate of improvement. However, some limitations are still prevalent on GPT-3. It is still far from being accurate for the more action-diverse natural text datasets. This becomes all the more apparent during argument extraction where, as shown, it generally fails to obtain competitive scores even on its most powerful Davinci version. This hinders the possibility of using GPT-3 directly for generally complex extraction tasks. For less diverse plans, it does show competing performance and we posit that it could be used as an intermediate step in a hybrid system. On the other hand, GPT-3 seems to show some ability to identify the underlying sequentiality of the plan by recognizing words like *before, after, first, anytime* or *eventually* and rearranging the plans accordingly. This is a capability generally missing from most state of the art plan extractors as they assume the ordering of the plan to be same as that of the sentences corresponding to each action in the text. Hence, ordering speaks for yet another potential advantage of using general models, as in they are usually not limited by specific assumptions made by system designers. Finally, note that the aforementioned strengths of the model could be further augmented should OpenAI allow for more finetuning in the future.

References

Addis, A.; and Borrajo, D. 2010. From unstructured web knowledge to plan descriptions. In *Information Retrieval and Mining in Distributed Environments*, 41–59. Springer.

Branavan, S.; Chen, H.; Zettlemoyer, L.; and Barzilay, R. 2009. Reinforcement Learning for Mapping Instructions to Actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 82–90. Suntec, Singapore: Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P09-1010.

Brown, T.; Mann, B.; and et al., R. 2020. Language Models are Few-Shot Learners. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 1877–1901. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Daniele, A. F.; Bansal, M.; and Walter, M. R. 2017. Navigational instruction generation as inverse reinforcement learning with neural machine translation. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, 109–118. IEEE.

Feng, W.; Zhuo, H. H.; and Kambhampati, S. 2018. Extracting action sequences from texts based on deep reinforcement learning. *arXiv preprint arXiv:1803.02632* .

Gokaslan, A.; and Cohen, V. 2019. OpenWebText Corpus. http://Skylion007.github.io/OpenWebTextCorpus.

Lindsay, A.; Read, J.; Ferreira, J.; Hayton, T.; Porteous, J.; and Gregory, P. 2017. Framer: Planning models from natural language action descriptions. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 27.

Ma, X.; and Hovy, E. H. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics. doi:10.18653/v1/p16-1101. URL https://doi.org/10.18653/v1/p16-1101.

Mei, H.; Bansal, M.; and Walter, M. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Miglani, S.; and Yorke-Smith, N. 2020. NLtoPDDL: One-Shot Learning of PDDL Models from Natural Language Process Manuals. In *ICAPS'20 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS'20)*. ICAPS.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1(8): 9.