Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 5813–5820
Marseille, 20-25 June 2022

Comparing Approaches to Language Understanding for Human-Robot Dialogue: An Error Taxonomy and Analysis

Ada D. Tur[†], **David R. Traum**[††]
[†]Los Altos High School, Los Altos, CA, 94022, USA,
[††]University of Southern California Institute for Creative Technologies, Playa Vista, CA 90094, USA
adadtur@gmail.com, traum@ict.usc.edu

Abstract

In this paper, we compare two different approaches to language understanding for a human-robot interaction domain in which a human commander gives navigation instructions to a robot. We contrast a relevance-based classifier with a GPT-2 model, using about 2000 input-output examples as training data. With this level of training data, the relevance-based model outperforms the GPT-2 based model 79% to 68%, and an Oracle combination set an upper-bound of 85%. We also present a taxonomy of types of errors made by each model, indicating that they have somewhat different strengths and weaknesses, so we also examine the potential for a combined model.

Keywords: Dialogue, Human-Robot Interaction, Evaluation, Error Taxonomy

1. Introduction

There are many approaches toward language understanding for human-robot interaction. Some involve domain-specific or general grammars of understandable utterances, combined with appropriate actions to perform. Another approach is to learn an appropriate output from supervised training data which pairs appropriate outputs with given inputs. A third approach is to "fine-tune" a general purpose predictive language model with domain-specific data. We compare two instances of the latter two approaches in a domain where the task is to "translate" natural language instructions to a more restricted language that can be directly executed by a robot navigator. Actionable commands can be directly translated, while other utterances require communication directly with the user.

In previous work, (Gervits et al., 2021) was able to achieve over 75% accuracy using a classification approach. We have retested with additional training data on the same test data, both with the same classifier software used by (Gervits et al., 2021), as well as a new approach involving fine-tuning a large scale pre-trained transformer-based language model (GPT-2) (Radford et al., 2018). The original model shows a small improvement over the previous results with a smaller training set, while the GPT model is not quite as accurate. An error analysis shows that the models make some different errors, so we also investigate the potential for combining the two approaches.

We first constructed a revised taxonomy of errors and classify each of the 183 examples in the test set for both models as to whether they are correct or which category they fall in. We then create a confusion matrix showing which types of errors are more common to each model. We conclude with some preliminary steps toward combining the models to reduce the total number of errors.

2. Related Work

Many different forms of statistical analysis have been utilized for the task of human-machine interaction (Serban et al., 2016; Bonial et al., 2017). For instance, the "corpus-based approach", where the system is trained on data from a target domain, has been used for human-robot dialogue (Marge et al., 2017b; Marge and Rudnicky, 2011).

In the past few years, there have been several corpora related to human-robot interaction for navigation and object manipulation in 3D environments. In the Room-to-Room (R2R) (Anderson et al., 2018) dataset, each example includes a natural language instruction that the agent needs to follow to navigate in a real-world environment from the Matterport3D dataset (Chang et al., 2017). Cooperative Vision-and-Dialog Navigation (CVDN) (Thomason et al., 2020) is a also a natural language dataset situated in the Matterport Room-2-Room (R2R) simulation environment, however, in this dataset, the human and agent engage in a multi-turn conversation to complete the navigation task.

One barrier for many human-robot interaction tasks is the problem of misinterpreting, or not having adequate context to complete a task. There can be circumstances where a robot is told to move towards a specific landmark, without having an understanding of the specific setting and the landmarks around it. However, this issue has been addressed with changes to how data is annotated for a model, such that it is uniquely marked for turns with language that is grounded to the conversational or situational context. There is also a separation between data that is used for training dialogue systems in general contexts from data intended for a particular situated environment (Bonial et al., 2021).

ScoutBot is a dialogue interface for physical and simulated robots that supports collaborative exploration of

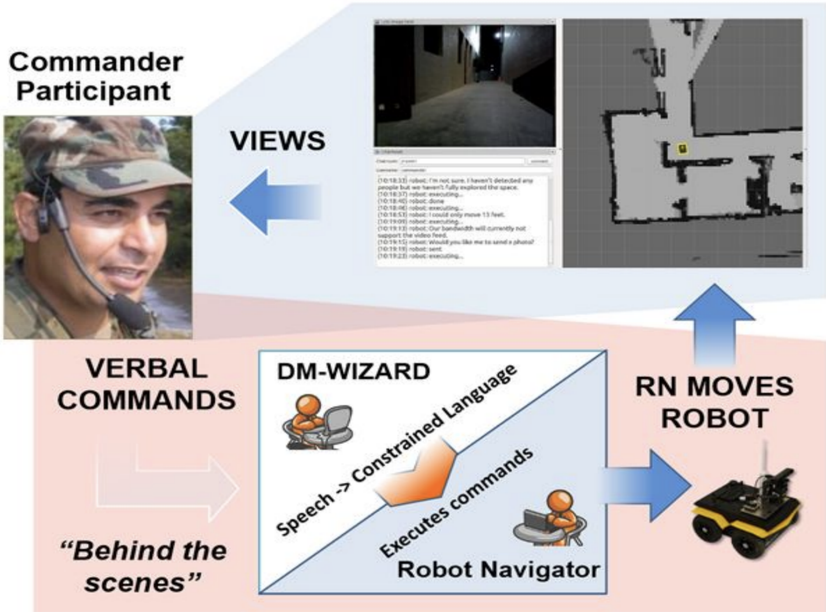

Figure 1: Human/Robot data collection setup (from (Marge et al., 2017a)).

environments (Lukin et al., 2018). Users can issue unconstrained spoken language commands to ScoutBot. ScoutBot prompts for clarification if the user's instruction is unclear or needs additional input. It is trained on human-robot dialogue collected from Wizard-of-Oz experiments, where robot responses were initiated by a human wizard in previous interactions. The demonstration shows a simulated ground robot (Clearpath Jackal) in a simulated environment supported by ROS (Robot Operating System) (Quigley et al., 2009). ScoutBot uses a classification approach described in (Gervits et al., 2021). It relies on the similarity score using a statistical language model as described in detail in (Leuski and Traum, 2010).

The recent ALFRED data set (Shridhar et al., 2020) focuses on human-robot interaction including multi-step everyday tasks. Transformer-based approaches have proven to be effective for this task (Jansen, 2020). For instance, OpenAI's GPT-2 model (Radford et al., 2018) has had success with the task, achieving an accuracy of 5% higher, on average, than approaches using Recurrent Neural Networks. As a result, we believe exploring the capabilities of transformer-based models for a similar task, such as ScoutBot, could be promising.

3. Task Description

The ScoutBot domain concerns collaborative navigation for urban search and rescue scenarios. In this task, a remotely located robot is performing certain navigation tasks as instructed by a Commander (Gervits et al., 2021). A participant, acting as the "Commander," issues verbal instructions to the robot partner. There are two main modules controlling robot behavior, which were operated by Wizards in Early data collection experiments. The "Dialogue Manager"(DM) interprets the commander's instructions and translates them to simplified versions and sends to the "Robot Navigator" (RN). The RN then controls the robot to execute the instruction. The Commander and DM-wizard can see the robot actions on a dynamically updated 2D LIDAR1 map. The RN then indicates completion or any problems, and DM conveys this information in Natural Language back to the Commander. Figure 1 depicts the Wizard-of-Oz-based data collection setup, and Figure 2 provides an example of how information is passed back and forth, using the dialogue structure annotations from (Traum et al., 2018).

Following (Gervits et al., 2021), the task for the dialogue manager is to interpret the commander's utterance and either translate it to an equivalent robot command using simplified language, or to give feedback to the commander if it is not executable without more information. Below are examples of natural language utterances with robot commands. If multiple commands are required they are separated by the keyword "then".

taskDesc: please move forward um five feet
annotation: move forward five feet

taskDesc: move west fifteen feet
annotation: turn to face west then move forward fifteen feet

4. Approach

We compare several different NLU methods. First, we report the best results from(Gervits et al., 2021), using the NPCEDitor and a training set containing 966 user utterances, 483 distinct system responses, and 995 links between user utterances and system responses. The current ScoutBot system uses an updated training set,

	Left Floor		Right Floor		Annotation	
#	**Commander**	**DM ->Commander**	**DM ->RN**	**RN**	**Ant.**	**Rel.**
1	rotate to the right ninety degrees					
2	and take a photo				1	continue
3		ok			2*	ack-understand
4			turn right 90 degrees		1	translation-r
5			then...		4	link-next
6			send image		2	translation-r
7				done and sent	6*	ack-done
8		done, sent			7	translation-l

Figure 2: Example annotation from human-robot interaction corpus (from (Gervits et al., 2021)).

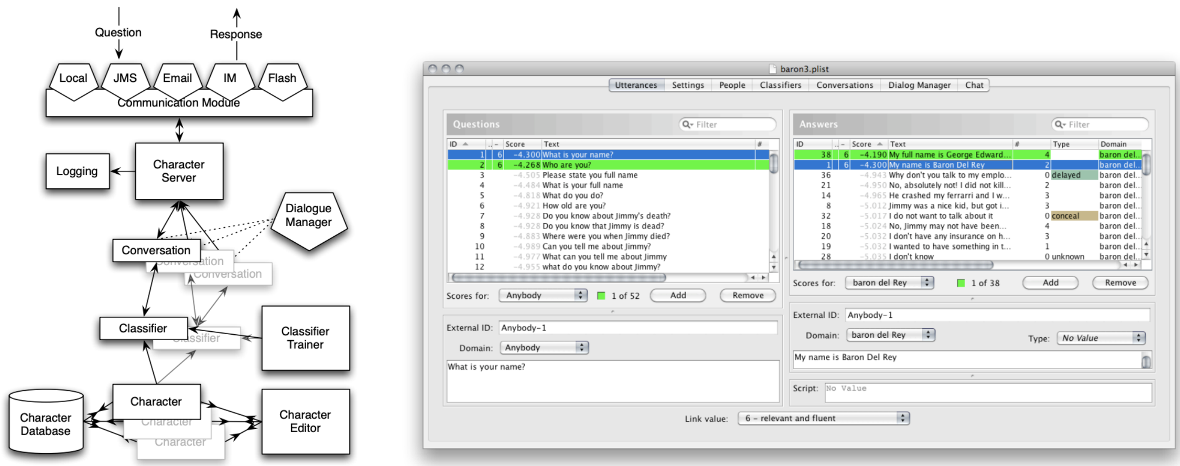

Figure 3: The NPCEditor system design (left) and character editor screen (right)

including 2058 user utterances, 493 system responses, and 2153 links. We also report on the performance of the NPCEditor on this extended dataset. The same extended training set is also used to fine-tune a generative model. We also report on combinations of the latter two models.

4.1. NPCEditor

NPCEditor is a system for building a natural language-processing component for virtual humans capable of engaging a user in spoken dialogue on a limited domain (Leuski and Traum, 2011). It uses statistical language-classification technology for mapping from a user's text input to system responses. NPCEditor provides a user-friendly editor for creating effective virtual humans quickly. It has been deployed as a part of various virtual human systems in several applications (Traum et al., 2012; Traum et al., 2015). NPCEditor functions primarily with the usage of a 'Dialogue Manager', which utilizes a classifier-based approach. The classifier consists of a statistical language model for each class, which is used to compute the cross-language relevance for each commander instruction. See (Leuski and Traum, 2010) for more information.

The system design of the NPCEditor program is shown in the left part of Figure 3. Each character has a respective fine-tuned classifier such that the inputs and outputs associated with each character and its classification are distinct from others in the server. The character editor screen is the interface used by users to define the inputs and outputs specific to a character. Each output is paired to at least one input, and each pair is defined in the interface as a training, evaluation, or testing example for the classifier. In this way, users can define the sizes of each data set, and the number of examples in each by providing the examples for the classifier to use.

4.2. Generative Model Training

The conversational deep learning model used is the OpenAI GPT-2 model (Radford et al., 2018), an auto-regressive model, utilized for the generation of responses to model tasks that a robot can interpret directly, rather than classification. We decided to utilize GPT-2 for this generative task based on prior work with other generative NLP tasks, like question-answering, textual entailment, and textual summarization (Radford et al., 2019). In GPT, the

Approach Name	Accuracy
NPCEditor	79.23%
(Gervits et al., 2021)	75.41%
OpenAI GPT-2, 25 epoch	68.31%
Oracle Combination	84.69%
Decision Level Combination	79.78%

Table 1: Experimental results using the NPCEditor and GPT-2 based approaches.

attention layer only attends to earlier positions in the output sequence using the causal language model objective (in contrast to masked language model objective). The model is fine-tuned on the ScoutBot data, a list of natural language commands a user may make to the robot, along with a gold standard format of the simplified command that the OpenAI GPT-2 should aim to predict. For each training instance, we separate each line by:

<Directive> [SEP]
<CommandTuple1> [CSEP]
<CommandTuple2> [CSEP]
... [CSEP]
<CommandTupleN> [EOS]

For example:

move west fifteen feet [SEP] turn to face west [CSEP] move forward fifteen feet [EOS]

The GPT-2 Medium transformer model used consists of 24 layers, 16 attention heads, and 325 million parameters, and contains decoder cells, meaning it uses masked self attention, where attention heads consider only what has appeared previously in a sequence, making it an auto-regressive generative model. During the generation process, top-k and nucleus sampling (Holtzman et al., 2019) were employed with beam search using the Huggingface Transformers library[1]. The OpenAI GPT was trained on 25 epochs.

5. Experiments and Results

As described above, the training set used in this study consists of 2058 manually annotated examples, with robot commands. The test set used for this task is the same as that used in (Gervits et al., 2021) and was derived from previously unseen, annotated dialogues, which remained unprocessed, with the exception of instruction-response extraction for each dialogue. This test set consists of 183 instruction-response pairs, and each instruction was input to both the NPCEditor and the OpenAI GPT. Overall, we attempted separate trials for both models, and also different combinations between them, in order to accommodate for the strengths and weaknesses between both models. The results are presented in Table 1. The NPCEditor trained on the extended dataset achieved an accuracy of 79.23%, improving slightly on the previously reported results from (Gervits et al., 2021) (75.41%). The GPT-2 model performed worse, with an accuracy of 68.3%. However, the GPT-2 model did accurately recognize some instances that were missed by the NPCEditor, so we also computed the performance of an "oracle" that could correctly choose the best response when the two differed. This oracle combination had an accuracy of 84.7%. We also tested a decision level combination, which divided tasks between the GPT-2 and the NPCEditor, giving all tasks with numerical values to the GPT-2, and all others to the NPCEditor, and this combination resulted in an accuracy of 79.78%.

[1]https://huggingface.co/docs/transformers/model_doc/gpt2

6. Error Analysis and Error Taxonomy

In order to get a more detailed sense of the types of errors that each model made, we manually classified each error according to a new error taxonomy, extending the analysis from (Gervits et al., 2021), which included the following error categories: 'Felicitous', 'Approximate', 'Context-Dependent', 'Wrong', and 'No Response'. Instead of looking at whether an answer was wrong or close, we focused on several identifiable sources of error. Our presented taxonomy is shown in Table 2. We define each of these below, with examples from the test set, then present a full confusion matrix showing the distribution of errors of each model.

Error Types
Genuine
Hallucination
No Response
Contextual
Numerical
Directional
Felicitous

Table 2: Error types

6.1. Error Types

Errors made by the OpenAI GPT-2 and the NPCEditor can be categorized into 7 different groupings: Genuine Errors, Hallucinations, No Response Cases, Contextual Errors, Numerical Errors, Directional Errors, and Felicitous Errors.

The first error category is the **Genuine Errors**. Genuine Errors are errors where a model (the OpenAI GPT-2 or NPCEditor) produces output that has great variance semantically from the gold standard, such that if executed the robot would do the wrong thing. This is equivalent to the 'Wrong' error category in (Gervits et al., 2021). Examples of Genuine Errors include:

- **taskDesc:** go one foot north
  annotation: move forward one foot
  predicted: turn to face north

- **taskDesc:** center in front of calendar
  annotation: move forward to front of calendar
  predicted: move into room

Hallucinations are errors where the model inserts objects, locations, or entities that do not exist in the task description or context, such that an extraneous addition to the description of the task completion is added unnecessarily. While in theory, a classifier model like NPCEditor could produce hallucinations, where the closest match includes additional information, in practice we saw only the GPT-2 model make them. Examples of Hallucinations are:

- **taskDesc:** go three feet
  annotation: move forward three feet
  predicted: move three feet towards green arrow

No Response cases are circumstances where a model returns no response to the task description, usually when potential responses have too low of a confidence level to be returned. No Response errors were made only by the NPCEditor, as the GPT-2 model always produces some output, regardless of its confidence, whereas the NPCEditor has an option to return no response (in which case the dialogue manager would ask the commander to say it again or rephrase). Generally, a No Response is preferred to other error types, assuming the response of the model is executed in a real-world scenario, as the No Response implies that the Commander must repeat the task with a different wording that the model may understand, whereas other error types may result in an incorrect execution of the command. Examples of No Response include:

- **taskDesc:** go back <pause> to table
  annotation: move back towards table
  predicted: [no response]

Contextual Errors occur in situations where the gold standard has more information than a model, in the form of the layout of a building, the direction the robot is facing at the time of the task, objects in the environment, etc. Usually, the input includes some underspecified referring expressions that require contextual information to fully disambiguate into actionable commands. The model does not resolve the intended referent and merely passes on an equivalently context-dependent referring expression. Both the OpenAI GPT-2 and the NPCEditor commonly make Contextual errors. This category is equivalent to the 'Context-Dependent' category in (Gervits et al., 2021). Contextual errors can be potentially solved using either a map of the environment as a further parameter for the training and testing of both models, or with the inclusion of computer vision into the algorithm, such that the robot has the capacity to analyze and interpret its surroundings. Instances where the models make Contextual errors are:

- **taskDesc:** go towards poster on left
  annotation: move to budapest poster
  predicted: move to poster on left

- **taskDesc:** go forward to nearest door well
  annotation: move to dark room hall doorway
  predicted: move forward to nearest door well

Numerical Errors are where a model makes errors regarding a numerical feature of the task. For instance, the model may return an incorrect value for degrees turned, distance moved, and other numerically-based descriptions. An approach toward solving and numerical errors would be to use a number tagger during preprocessing. Examples of some Numerical errors are:

- **taskDesc:** five degrees to your left
  annotation: turn left five degrees
  predicted: turn left 45 degrees

Directional Errors occur when a model returns a result that includes an incorrect direction parameter, for example direction to turn, or direction to move in. There are also instances where either model makes both a Directional and Contextual Error, such that the direction may be correct, but it does not match the gold standard. A direction tagger could be used in preprocessing, similar to Numerical errors, in order to handle them. Cases where Directional errors are made are:

- **taskDesc:** go one foot north
  annotation: move forward one foot
  predicted: turn to face north

- **taskDesc:** center in front of calendar
  annotation: move forward to front of calendar
  predicted: move into room

- **taskDesc:** turn right <pause> forty five degrees
  annotation: turn right forty five degrees
  predicted: turn left 45 degrees

Felicitous Errors are cases where a model's response returned does not match the gold standard, but the result is identical in meaning to the gold standard. For example, the gold standard may have a different ordering of words, or different terms used that the model was unable to match accurately. However, if the model were to execute the returned task with a robot, it would be able to complete the task correctly, even when it does not match the gold standard. Generally, Felicitous errors do not need further analysis to solve, as they do not effect robot performance. An example is:

- **taskDesc:** and move to the east five feet
  annotation: turn to face east then move forward five feet
  predicted: move to the east five feet

Errors	NPCEditor%	GPT-2 %
Genuine	10%	37%
Hallucination	0%	6%
No Response	8%	0%
Contextual	16%	16%
Numerical	24%	7%
Directional	26%	27%
Felicitous	16%	7%

Table 3: Distributions of error types between the GPT-2 and NPCEditor

6.2. Error Comparison

Table 3 shows the proportion of error types for each recognizer. While they make a similar proportion of direction and contextual errors, the GPT-2 model makes more frequent genuine and hallucination errors, while NPCEditor makes more no response errors and numerical and felicitous errors.

Table 4 shows a confusion matrix between the NPCEditor and GPT-2 results. Of the 183 test instances, 104 were correctly recognized by both models, while 52 were recognized correctly by only one of the models (41 for NPCEditor, 11 for GPT-2), and 27 were not recognized by either. Results from the confusion matrix depict an analysis of our results of both models. In many cases, the NPCEditor and GPT-2 made the same types of errors, with exceptions where one model was correct or made a different type of error. For instance, Hallucination errors are made only by the GPT-2 (likely because of low confidence in a response, or part of a response), and No Response errors are only made by the NPCEditor, because it returns no response when its confidence is below a threshold. The cases where both models made the same error are the Contextual and Directional categories. Contextual errors were more likely to cause an error in both models because neither model has pre-defined contextual information for a response, while the gold standard does. Directional errors were commonly made by both models due to common confusions between typical training examples and the specific test examples.

6.3. System Combination and Continuations

In this work we have further explored various ways of combining these two approaches, allowing for the optimization of the strengths of both models, as well as experiments into continuations of the improvements of the model performance.

We first analyzed how access to more training data affects the performance of the GPT-2 model. It was shown that with the presence of more training data, the GPT-2 could have reached a closer accuracy to the NPCEditor, and this was tested by limiting the existing training data and finding how the model performed. Given available training data, we limited the training data in three categories, 'Full Training Data', 'Half Training Data', and 'Quarter Training Data'. Its

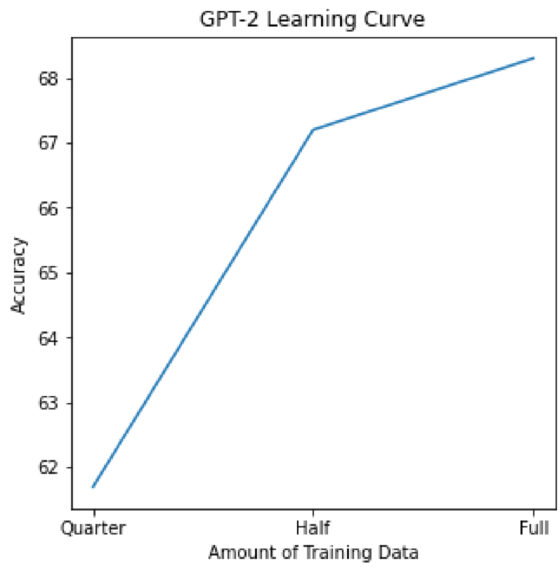

Figure 4: Learning Curve of GPT-2 with limited training data

learning curve below depicts an improvement in performance given more data, which can potentially indicate that once more training data is available, the GPT-2 may have better results.

For combinations, there are many different practices that could results in improved performance. For instance, the NPCEditor may make mistakes with utterances consisting of numerical values, such as "rotate twenty five degrees to your right ", whereas the GPT-2 model cannot handle unseen utterance types, such as "take a picture looking west". This motivated us to experiment with further combination efforts.

We have explored two approaches to combine the NPCEditor and GPT-2 predictions:

- **Oracle combination:** We tested the Oracle combination to establish an upper-bound for the performance of our models and their combinations. The Oracle combination utilizes the responses from either the NPCEditor or GPT-2, dependent on which is correct for each task.
- **Decision level combination:** As an alternative, we have combined the predictions based on model characteristics, such as their confidence, or utterance characteristics, such as whether it contains numbers. We found that giving tasks that include numerical values to the GPT-2, and giving a majority of other commands to the NPCEditor resulted in considerably higher accuracy, as the GPT-2 was more easily able to analyze and return correct numerical values.

The Oracle combination improved our accuracy to 84.69%, and we received results that were 0.5% higher than our NPCEditor using the decision level combination, which had a 79.78% accuracy.

x	Genuine	Hallucination	No Response	Contextual	Numerical	Directional	Felicitous	Correct	Total
Genuine	4	0	1	0	0	0	2	18	25
Hallucination	0	0	0	0	0	0	1	3	4
No Response	0	0	0	0	0	0	0	0	0
Contextual	0	0	1	6	0	0	0	4	11
Numerical	0	0	0	0	2	0	1	2	5
Directional	0	0	0	0	0	6	0	12	18
Felicitous	0	0	1	0	0	0	2	2	5
Correct	0	0	0	0	7	4	0	104	115
Total	4	0	3	6	9	10	6	145	183 each

Table 4: Confusion matrix on error categories between the NPCEditor and GPT-2, where rows correspond to GPT-2 errors and columns correspond to NPCEditor errors

7. Conclusions and Future Work

We have presented our research on the intersections of previously existing technologies for human-robot interaction, compared and combined with more contemporary forms of deep learning-based approaches, particularly transformer-based models. Although the NPCEditor performs more effectively than the OpenAI GPT-2 on the available training data, deep learning-based models continue to show potential for growth and improvement in the future, especially when larger amounts of data is provided. One possible alternative to using GPT-2 for a generative task is to utilize the Bidirectional Encoder Representations from Transformers model (BERT)-style encoders for classification. To get the best of both worlds, we plan to explore the usage of an encoder-decoder architecture, such as the bidirectional and auto-regressive transformers (BART), as well as other examples. Additionally, we will continue to explore further model combinations as part of our future work.

ScoutBot can be used to test newly emerging deep learning approaches. In coming studies, it would be beneficial to research how modules consisting of different combinations between the NPCEditor and auto-regressive models can surpass current performance. For instance, tasks containing distinct traits can be divided between different models in order to leverage the strengths of each model. With the error analysis conducted, further context could be provided for this separation, showing where each model achieves higher performance and how models can collaborate to complete each task. Contextual tasks, particularly for specific landmarks in an environment, can be improved through context used for input for an NLU classifier, with transformations conducted on the input prior to classification. The action interpreter can also handle situated context dependent instructions.

Acknowledgements

The second author was supported in this work by the US Army. We thank Felix Gervits for sharing the test set used in (Gervits et al., 2021), and other members of the Scoutbot project at the Army Research Lab and the Institute for Creative Technologies for the updated training set.

8. Bibliographical References

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and Van Den Hengel, A. (2018). Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683.

Bonial, C., Marge, M., artstein, R., Foots, A., Gervits, F., Hayes, C., Henry, C., Hill, S., Leuski, A., Lukin, S., Moolchandani, P., Pollard, K., Traum, D., and Voss, C. (2017). Laying down the yellow brick road: Development of a wizard-of-oz interface for collecting human-robot dialogue. 10.

Bonial, C., Abrams, M., Baker, A. L., Hudson, T., Lukin, S. M., Traum, D., and Voss, C. R. (2021). Context is key: Annotating situated dialogue relations in multi-floor dialogue. In *Proceedings of the 25th Workshop on the Semantics and Pragmatics of Dialogue*.

Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., Song, S., Zeng, A., and Zhang, Y. (2017). Matterport3d: Learning from rgb-d data in indoor environments.

Gervits, F., Leuski, A., Bonial, C., Gordon, C., and Traum, D. (2021). A classification-based approach to automating human-robot dialogue. In *Increasing Naturalness and Flexibility in Spoken Dialogue Interaction: 10th International Workshop on Spoken Dialogue Systems*, pages 115–127. Springer Singapore.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2019). The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Jansen, P. A. (2020). Visually-grounded planning without vision: Language models infer detailed plans from high-level instructions. *arXiv preprint arXiv:2009.14259*.

Leuski, A. and Traum, D. (2010). Practical language processing for virtual humans. In *Twenty-Second IAAI Conference*.

Leuski, A. and Traum, D. (2011). Npceditor: Creating virtual human dialogue using information retrieval techniques. *Ai Magazine*, 32(2):42–56.

Lukin, S. M., Gervits, F., Hayes, C. J., Moolchandani, P., Leuski, A., Rogers III, J. G., Sanchez Amaro, C., Marge, M., Voss, C. R., and Traum, D. (2018). ScoutBot: A dialogue system for collaborative navigation. In *Proceedings of ACL 2018, System Demonstrations*, pages 93–98, Melbourne, Australia, July. Association for Computational Linguistics.

Marge, M. R. and Rudnicky, A. (2011). The teamtalk corpus: Route instructions in open spaces.

Marge, M., Bonial, C., Byrne, B., Cassidy, T., Evans, A. W., Hill, S. G., and Voss, C. (2017a). Applying the wizard-of-oz technique to multimodal human-robot dialogue. *arXiv preprint arXiv:1703.03714*.

Marge, M., Bonial, C., Foots, A., Hayes, C., Henry, C., Pollard, K., Artstein, R., Voss, C., and Traum, D. (2017b). Exploring variation of natural human commands to a robot in a collaborative navigation task. In *Proceedings of the first workshop on language grounding for robotics*, pages 58–66.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A. Y., et al. (2009). Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training (2018).

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

Serban, I., Sordoni, A., Bengio, Y., Courville, A., and Pineau, J. (2016). Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. (2020). Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749.

Thomason, J., Murray, M., Cakmak, M., and Zettlemoyer, L. (2020). Vision-and-dialog navigation. In *Conference on Robot Learning*, pages 394–406. PMLR.

Traum, D., Aggarwal, P., Artstein, R., Foutz, S., Gerten, J., Katsamanis, A., Leuski, A., Noren, D., and Swartout, W. (2012). Ada and grace: Direct interaction with museum visitors. In *Intelligent Virtual Agents: 12th International Conference, IVA 2012, Santa Cruz, CA, USA, September, 12-14, 2012. Proceedings*, volume 7502, page 245. Springer.

Traum, D., Jones, A., Hays, K., Maio, H., Alexander, O., Artstein, R., Debevec, P., Gainer, A., Georgila, K., Haase, K., et al. (2015). New dimensions in testimony: Digitally preserving a holocaust survivor's interactive storytelling. In *International Conference on Interactive Digital Storytelling*, pages 269–281. Springer.

Traum, D., Henry, C., Lukin, S., Artstein, R., Gervits, F., Pollard, K., Bonial, C., Lei, S., Voss, C., Marge, M., et al. (2018). Dialogue structure annotation for multi-floor interaction. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.