Teaching Robots to PAANIC: Personal Assistive Agents Noticing Impending Catastrophes

Liam Pavlovic, Brant Wesley, and Abby Carr

Index Terms—**Natural Language Processing, Robotics**

***Abstract*—Robotic advancements call for the further application of Natural Language Processing (NLP) and the ability of robots to understand natural language instructions from humans. As such, we have developed a method to reduce mistakes in robotic following of human commands by implementing hazard tagging on natural language instructions, which can then be modified into the real-life application of robotic grip strength, general procedure for a task, and navigation of dangerous situations.**

I. Introduction

Robots have become ingrained into the world around us. They control a large number of functions from specific tasks in manufacturing cars to translating languages from around the world. We use them as a part of how we live, and as we have adapted to having these computers as a large part of our life, we have begun researching how to give them more and more complicated tasks. Instead of following a simple and single direction, we want robots to be able to adapt and complete more complete tasks with less knowledge. For example, instead of having a specific spoken or typed phrase that triggers a robot to move forward, we want that robot to be told to explore and be able to move freely around a given space. It is this example that encompasses the drive forward to robots who can act freely and, in our eyes, become more helpful.

While this expansion and goal applies to many different facets of robotics, for the scope of our task we focused on specifically home assistive robots. While you may think of an audio home robot connected to the electricity or thermostat of a home, in this case we mean a moving robot with free range to explore and assist throughout the home with physical tasks that may include tending a garden or assisting with cooking. They would be another piece of the home and likely have some level of autonomy. This may not be commonplace at the moment, but home-assistive robots (as you will continue to see them referred to as) have been depicted in all types of media and are a common picture you see when thinking about the future of technology.

Home-assistive robots are of special interest when it comes to the task of detecting hazards. These robots are likely to deal with one to many individual humans, along with pets, and will need to be able to adapt and respond based on these individuals in the given household. Whether instructions are given through natural language, typed, or a set list of options, a physical robot will need to complete tasks in a dynamic environment with real and present dangers to both the robot and the humans around it. It is unlikely that all owners or people interacting with a home-assistive robot will have the ability to keep track of and stop all potential dangers, meaning the robots will need to be able to acknowledge and respond to danger.

For our purposes, we wanted to focus on the detection of those hazards. We believe that there is value in robotic planning that will come in the reaction to the recognition of certain hazards, and wanted to set the foundation for planners by first detecting hazards themselves. Our focus lies in extracting hazards from sets of natural language instructions- those that may be typed or spoken. Our goal is to test the first round of models on extracting hazards from these instructions.

To this end, we have developed a dataset and a series of natural language models for tagging the hazards implicit in a set of instructions given to a robotic system. For the purpose of this paper, we have specifically focused on the context of a robot performing tasks in a kitchen - from preparing food to locating and washing dishes. We wanted to tag the directives to mime the way a human may know a knife is sharp and therefore dangerous, or how a oven can be hot.The tags given to the data correspond to various hazards present in a kitchen, and were applied to a selection of data points from the TEACh dataset by hand.

Upon completion of the creation of the dataset, we then trained several different neural networks with this dataset in order to assess how well various architectures could detect potential hazards in this data. We hoped to find the baseline performance capabilities of the models we created and find insights for what we might leverage for hazard tag modeling as well as provide a path for future action.

II. Motivation

As research in the world of robotics continues to blossom and become ever more integrated into everyday life, it is important that robots be able to recognize the potential hazards and missteps in the tasks they face. While robots are often given instructions that may not be inherently perilous, the unpredictable nature of real-world environments means that there will be factors that the robot cannot necessarily control for. Additionally, the instructions given to a robot may not necessarily present an immediate hazard - just the potential for an accident to occur should the robot malfunction or the state of the world around the robot suddenly changes (like the onset of an earthquake or the sudden appearance of a wild boar). Whatever may happen, it is imperative that a robot be aware

of how the objects it interacts with may become hazardous to the robot or those around it should something go wrong.

In the daily lives of many, dangers occur in many different ways and in many different forms. As we advance in the development of robotic implementations in the real world, one of the most important things to think about is adaptability. Without adaptability, robots may be able to cause harm to themselves and others- whether that is their intended purpose or not. As such, we intend to pursue one branch of the expansion by looking at hazard identification. As we will address in our related works, it is not novel for machine learning tasks to focus on extraction or safety. We do however plan to explore extraction and safety; more specifically, we wish to explore the novelty of targeting the safety surrounding robots and the scope in which they operate. It is our hope that with recognition programs for robots, we can minimize the danger that is created with an increasing implementation of them in our world.

Finally, we wish to mention the unique aspect of our work. While we wish to provide important and useful results from our work, the purpose and importance in this task additionally comes from the exposure our research brings with it. It may be commonplace to think of the pitfalls in safety that have potential to occur when it comes to a self-driving car, but it is not addressed in a wider and more applicable scope for the rest of robotics. Our work could start the groundwork of addressing flexible safety measures in robots, and can leave a path to follow and branch off of when it comes to seeing how we plan to insure robots.

III. Related Work

There is not much work regarding the specific tagging of elements of NLP instructions, but two interesting papers on extracting elements from unstructured text provide similar methods to those we will trace throughout PAANIC!. Kwayu's paper [1] on the extraction of hazardous actions from police statements utilized n-gram feature analysis to assist engineers in obtaining the necessary information to figure out what type of accident a given statement is about. This technique is one we want to investigate further as we build our model, as the ability to identify a 'hazard' is a shared part of our project like that of Kwayu. The other relevant paper pertains to extracting hazards from unstructured text with the additional task of finding the dynamics of said events. This paper by Wang [2] instead utilized geographic information system (GIS) mapping to extract the information from tweets. Similar to the first paper, we find the methods of extraction relevant to the work we wish to do in the NLP sector.

The other work related to our hazard tagging falls under a number of different umbrellas. Andrade [3] analyzes existing hazard data pertaining to wildfires which relates to the application aspect of our work. Johannes [4] identifies required elements from a textual write-up of chemical accident reports, while Li [5] and Ma [6] find key elements in the hazard and operability analysis text and geological hazard documents respectively. The extraction of data from hazards seems to be a common thread uncovered when searching for work with NLP and hazards. Another similar example lies with Daramola [7] who wrote about the development of an automated safety analysis tool and in Kwon [8] who made an automation of creating and finding construction safety requirements for a given project. All of the related work clearly shows that NLP analysis of hazards is not uncommon, and that moving in the direction of robotic hazard conscientiousness is a natural progression from existing work.

IV. Task Description

For data, we used the TEACh dataset for our inputs which is a "dataset of human-human interactive dialogues to complete tasks in a simulated household environment." [9] This dataset is publicly available for use and download.

We obtained the instructions by selecting the instruction given to the robot by a human in the dialogues used in the simulator. Due to time constraints, we used a subset of around 7000 instructions as a mini-batch to label to help us recognize any issues and standardize our labeling process before moving forward to a larger section of the TEACh data. In order to manually apply the labels, we read the sentences of instructions given to the robots and applied whatever labels we deemed relevant.

The focal point of the dataset for our project was the labeling and cementing of labels for our data. We labeled a small subset of data with a small set of tags: Fragile, Hot , Sharp, Slippery, Electricity, Wet, and, the default, No Hazard. Although all the tags deal with safety, they primarily focus on the safety of a robotic system while preforming actions with an easy expansion into tagging for the purposes of user safety. This could be expanded in the future with the addition of more tags based on different or more complex hazards, such as "falling hazard" or "sticky", or things that may be hazardous to a nearby human (or other organism) but not the robot itself, such as "biohazard".

While assigning labels to the data was in most cases relatively straightforward, there were some scenarios in which we felt that the necessary labels for a sentence were somewhat ambiguous and some assumptions were necessary. For instance, while we labelled most interactions with appliances such as toasters and microwaves with the "electric" tag, we chose not do the same for stoves, as a non-negligible subset of stoves are gas, not electric. Likewise, we also made the assumption that coffee would always be hot rather than iced, since the instructions in the dataset involving coffee always referred to a classic drip coffee machine, which always makes hot coffee.

Additionally, while instructions requiring the robot to interact with appliances almost always resulted in a "hot" or "electric" tag and utilizing the sink would result in a "wet" tag, did not apply these tags to instructions that used these appliances and objects as waypoints to tell the robot where

something is. For instance, if an instruction told the robot to grab the lettuce from the cabinet next to the microwave, this would not result in an ”electric” tag because the robot does not interact with the microwave. Because of these various nuances in the data, the models may require a greater amount of training data in order to learn these more subtle relationships between items and the hazards they may or may not present depending on the context of how they are referenced.

To clean the textual data we want to work with, we simply extracted the text instructions and removed commands less that 4 words in length. This cutoff was chosen in order to reduce the number of extraneous greetings and comments that are outside the scope of our project goals (ex. the inclusion of ”hi!” as an instruction).

When labeled, our data showed a slightly uneven distribution of tags with fragile being the most prevalent and slippery being the rarest. (Fig. 1.) This uneven result was not too severe considering the multi-label nature of each data point we had, and considering some points had no labels at all.

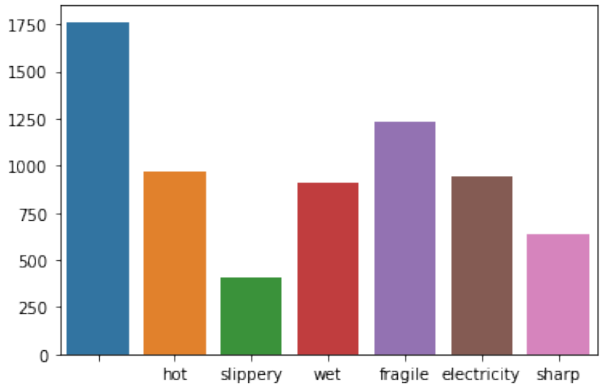

Fig. 1. The number of entries for each hazard tag as well as the number of entries with no hazard tags

V. Method

We developed and tested four different neural network architectures for the hazard tagging task. The simplest of these was a basic feed-forward network. The input of this network is a flattened sequence of GloVe embeddings representing the words of the input sentence. Since feed-forward networks must have constant input size, these sequences padded to a maximum length and sentences longer than this length are dropped from the training set. The GloVe embeddings used are the ”glove-wiki-gigaword-200” embeddings from the Python gensim library. This network has 4 hidden layers of size 1000, 500, 100, and 50, in that order. Each of these hidden layers uses ReLU activation. The final output layer uses sigmoid activation to compute an individual likelihood estimate for each of the six potential hazard tags.

Our next neural architecture utilized a basic recurrent neural network (RNN) to generate a semantic vector that was then passed through a feed-forward network. The inputs to the RNN were the same embeddings used for the feed-forward network, except they were input sequentially, rather than as a whole, and were not padded. The RNN has three recurrent layers that each had size 1000 hidden state. The final hidden states from each recurrent layer were combined into a 3000 long semantic vector which was then fed into a feed-forward network. This feed-forward network has the same architecture as our feed-forward model.

In order to expand upon the basic RNN, we also implemented an RNN with Long Short-Term Memory units (LSTMs). This model took inputs in the same manner as the basic RNN, as this type of network operates in essentially the same manner at a high level with network at each timestep receiving information about the previous timestep’s state as an input. However, the LSTM receives both the previous hidden state as well as the previous LSTM cell state as inputs, as opposed to a basic RNN which only receives the previous hidden state.

The LSTM unit includes several ”gates”: the forget gate (sigmoid followed by multiplication), which learns what information should be forgotten at the time step; the input gate (a tanh and sigmoid multiplied together, given the previous hidden state and input), which determines what values in the input are important for the cell state; and the output gate (the sigmoid of the input/previous hidden stated multiplied by the hyperbolic tangent of the cell state), which determines the output of the network at this time step (the next hidden state).

The LSTM unit relies largely on the cell state for the determination of its ouput, as this portion of the cell receives as input the previous LSTM cell state and receives information from the input and forget gates to help determine the output of the LSTM unit. The cell state is also passed on to the next timestep along with the hidden unit to help inform the cell state of the next time step (if there is one).

Our final and best performing network architecture uses a pre-trained BERT model to create a semantic vector for the input sentence which is then passed through a feed-forward network to classify the sentence into tags. The pre-trained BERT model utilized is HuggingFace’s ”bert-base-uncased” model. This model is pre-trained on the masked language modeling objective on a combined dataset of books and Wikipedia articles written in English. The BERT model receives two inputs the first of which is a sequence of numerical tokens representing the words of the input sentence produced by a one-to-one mapping of corpus words to numbers. This sequence is padded at the end to a maximum length for computational efficiency purposes. A [CLS] token is also inserted to the beginning of each sequence. The model also receives an attention mask as input, which is the same length as the token sequence and indicates which tokens are padding. The BERT model itself is a series of transformer encoder blocks. From these two inputs, the BERT model produces an embedding for each token in the original token sequence. Our model only utilizes the embedding for the [CLS] token, which is configured to pool the embeddings from the other tokens and thus is a semantic vector for the entire sentence. This embeddings is passed through a feed-forward network with

two hidden layers of sizes 100 and 50 in that order. Each of these hidden layers utilizes ReLU activation. The hidden layers are followed by an output layer which utilizes sigmoid activation to compute an individual likelihood estimate for each of the six potential hazard tags. (Fig. 2.)

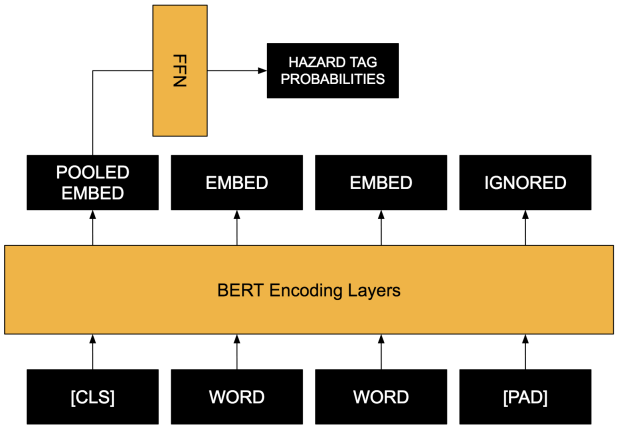

Fig. 2. A high-level overview of the architecture of our BERT model

Each model was trained until continued epochs no longer increased accuracy performance on the training set. The best performing models tended to require fewer epochs to reach this point, with BERT requiring the fewest epochs at 15 and the feed-forward network requiring the most at 50. The loss curves for each model can be seen in the figure below. (Fig. 3.)

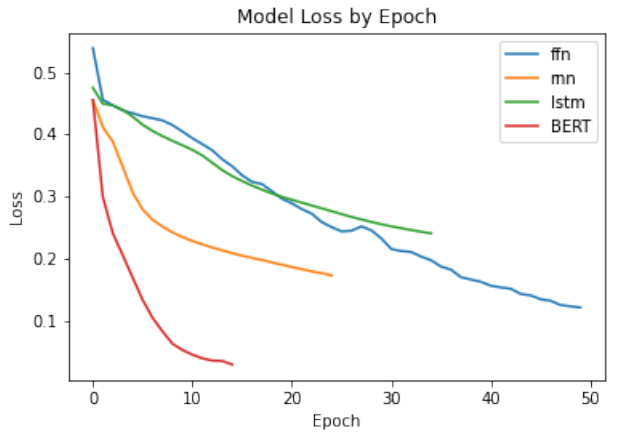

Fig. 3. Loss curves for our 4 models

VI. Evaluation

To evaluate our models, we used accuracy, precision, and recall metrics. Accuracy was measured by considering a model's prediction correct only if it predicted the exact set of golden hazard tags for that input sentence. Precision and accuracy were measured for each hazard tag separately and then averaged to measure overall performance Table 1 shows the performance of each of our four architectures on these metrics.

To better understand how the unbalanced nature of our data set affects the models, we examined the individual recall and precision scores achieved for each hazard tag by our best performing architecture, the BERT model. Table 2 shows these results.

Lastly, to present the differences in classification between the models, we ran the sentence "Boil a sliced potato and wash a plate" through each of our model's to see which hazard tags each model assigned to it. Table 3 shows these results.

VII. Results

Overall, the results in Table 1 show that non-recurrent models perform the worst on this task. Classical recurrent models perform better than non-recurrent and models with attention, like BERT, perform the best. These results are not particularly surprising as transformer's outperform other models on most natural language processing tasks. However, an interesting result to note is that the basic RNN model outperformed the LSTM model. LSTM architecture was originally conceived as a way to extend the interaction distance between tokens in a recurrent network. However, most relevant interactions for this task are fairly immediate and usually prepositional. For example, the sentence "get the knife from the sink" would have the wet hazard tag but the sentence "get the knife from the counter opposite the sink" would not. Thus, the LSTM architecture is not a great fit for this task and the semantic vector would capture many interactions that are irrelevant to the classification task. The noise introduced by this extraneous information could be the cause of the LSTM's lower performance.

The results from Table 2 suggest an overall trend that the more common a tag is, the better the model performs on it. This result is not very surprising, as model's usually perform better on tasks that they have more data for. One outlier from this overall trend is the electricity tag, which had worse performance than both the less common sharp and wet tags. This is likely because recognizing an electrical hazard often requires more advanced understanding of the input sentence. Recognizing the wet and sharp tags usually only requires recognizing the presence of a liquid or the usage of a knife. Meanwhile, recognizing the electrical tag usually requires extracting spatial information from the instruction. For example, the sentence "slice the potato next to the microwave" has the electrical hazard tag because the robot will be utilizing a metal tool near the cord of the microwave.

The results from Table 3 follow the same general trend of the BERT Encoder outperforming the model. Interestingly, the RNN is the only model that did not incorrectly classify the input as having the electricity hazard tag. Most likely, the other models have an incorrect association with some words in the sentence and the electricity label due to over-fitting on noise in the training set. The RNN did not over fit on this noise by coincidence. Another interesting result is that the BERT encoder was the only model to successfully recognize

the slippery tag. This tag has the least number of entries in the data set and is therefore the most difficult for the models to recognize.

VIII. Conclusion

The main contribution of this study is the introduction of a new task for cognitive robotics. As discussed earlier, teaching robots to recognize hazards in the instructions they are given has not been explored yet. The ability to recognize implicit meanings in sentences is an important feature for robot-human communication in general. That said, recognizing implicit hazards is of particular importance because of the potential damages, to both humans and property, that lacking this skill can cause.

In this study, we provide a dataset for learning this task and a performance baseline. An immediate next step would be to improve performance by expanding the dataset and adding more variety to the types of instructions. The instructions from the TEACh dataset are almost entirely for a small set of kitchen tasks. This expanded dataset could also include more hazard tags, representing hazards that were not really relevant to the instructions in the TEACh dataset. Adding variety to the dataset, particularly in the types of tasks the instructions are detailing, will help the model generalize better to all household tasks.

Another important next step is integration with a robotic controller. Now that the hazards in instructions can be identified, it's important that robots actually adjust their behavior to avoid the hazard. For example, when detecting the sharp tag, a robot may tighten its grip on held objects and move around its environment slower than it would normally.

To better enable integration with a robotic planner, exploring a multimodal approach for this task is another avenue worth exploring. Namely, having an algorithm that enables a robot to segment its environment to locate where specifically the hazards are will allow a robotic planner to generate a more effective plan for avoiding that hazard.

References

[1] K. M. Kwayu, V. Kwigizile, J. Zhang, and J.-S. Oh, "Semantic n-gram feature analysis and machine learning–based classification of drivers' hazardous actions at signal-controlled intersections," *Journal of Computing in Civil Engineering*, vol. 34, no. 4, p. 04020015, 2020.

[2] W. Wang, *Automated spatiotemporal and semantic information extraction for hazards*. The University of Iowa, 2014.

[3] S. R. Andrade and H. S. Walsh, "Wildfire emergency response hazard extraction and analysis of trends (heat) through natural language processing and time series," in *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*. IEEE, 2021, pp. 1–10.

[4] J. I. Single, J. Schmidt, and J. Denecke, "Knowledge acquisition from chemical accident databases using an ontology-based method and natural language processing," *Safety Science*, vol. 129, p. 104747, 2020.

[5] F. Li, B. Zhang, and D. Gao, "Chinese named entity recognition for hazard and operability analysis text," in *2020 Chinese Control And Decision Conference (CCDC)*. IEEE, 2020, pp. 374–378.

[6] Y. Ma, Z. Xie, G. Li, K. Ma, Z. Huang, Q. Qiu, and H. Liu, "Text visualization for geological hazard documents via text mining and natural language processing," *Earth Science Informatics*, pp. 1–16, 2022.

[7] O. Daramola, T. Stålhane, I. Omoronyia, and G. Sindre, "Using ontologies and machine learning for hazard identification and safety analysis," in *Managing requirements knowledge*. Springer, 2013, pp. 117–141.

[8] J. Kwon, B. Kim, S. Lee, and H. Kim, "Automated procedure for extracting safety regulatory information using natural language processing techniques and ontology," *GEN*, vol. 30, p. 1, 2013.

[9] A. Padmakumar, J. Thomason, A. Shrivastava, P. Lange, A. Narayan-Chen, S. Gella, R. Piramuthu, G. Tur, and D. Hakkani-Tur, "Teach: Task-driven embodied agents that chat," *arXiv preprint arXiv:2110.00534*, 2021.

TABLE I
MODEL PERFORMANCE

	FFNN	Simple RNN	RNN with LSTM	BERT Encoder
Accuracy	.5226	.5800	.5501	.6373
Avg Precision	.6255	.6947	.6988	.7232
Avg Recall	.6249	.6326	.5929	.6914

TABLE II
BERT INDIVIDUAL TAG RESULTS

Hazard Tag	**Precision**	**Recall**
Fragile	.8345	.8098
Hot (temperature)	.7257	.7611
Sharp	.7976	.8375
Slippery	.5873	.3854
Electricity	.6167	.6352
Wet	.8037	.7197

TABLE III
TAGGING THE EXAMPLE

	{sharp}	{fragile}	{hot}	{electricity}	{wet}	{slippery}
FFNN			x	x	x	
Simple RNN		x	x			
RNN with LSTM		x	x	x	x	
BERT Encoder		x	x	x	x	x
Golden Label		x	x		x	x