WikiHow Dungeon: An Interactive Text Adventure Game

Gizem Dal[1], **Sanjeevini Ganni**[1], **and Andrew Martin**[1]

[1]**University of Pennsylvania: Interactive Fiction & Text Generation (CIS700)**

ABSTRACT

We have developed an interactive text-based game experience named WikiHow Dungeon where the player can solve WikiHow puzzles either by picking an article category from the menu or uploading an article file to play with. Apart from the game component, we have also created a training component where we trained a transformer based GPT-2 model on WikiHow articles to generate new puzzles to play.

Keywords: POS-tagging, Imperatives, Object detection, Embeddings, Interactive puzzle, GPT-2

INTRODUCTION

We are providing an interactive text-based game experience where the players try to solve puzzles from WikiHow articles (wik). Such articles can either be pre-existing articles from the WikiHow database or they can be generated by training a transformer based GPT-2 model on the WikiHow article dataset. Each game experience involves telling the player the goal of the article, and asking them to guess the correct steps from a given list of options each time in order to solve the puzzle. Our project has multiple moving parts such as parsing the articles to extract steps and descriptions, running an imperative check on steps to determine which steps require an action, changing the descriptions to be printed after every action from third-person to second-person, finding articles similar to the game article for extracting wrong choice options per step, grouping articles into 20-30 predetermined categories for the player to pick from if they choose to and creating a GPT-2 model to train on the WikiHow dataset and generate new exciting puzzles. What makes our project interesting is we are using a very wide variety of WikiHow articles to provide a fun interactive experience. WikiHow is a website known for its 'How To' articles on almost any topic one can think of and it is also a rich source for NLP research. However, there isn't much work in the field that focuses on how such articles can be used to create interactive text adventure games, thus our project opens space for a lot of further work to be done in this area.

RELATED WORK

The closest existing game experience to our project is AI Dungeon (aid). Similar to how new puzzles and scenarios are created in AI Dungeon after each action, we wanted to be able to create new setting and problem to solve for the player after the correct action is taken.
One paper that inspired us for our project is (6)- *Neural text generation in stories using entity representations as context*, where the researchers are introducing an approach for tracking entities in a generated text and generating following sentences that mention such entities. Such a task requires keeping track of the state of the story such that if an event occurs and there are outcomes, the following sentences generated should include references to the previous characters and the results of the interactions.Although we're not implementing our game structure the same way that they approached this problem, we're still similarly trying to build a system where the player is picking actions to progress in the game within a given context and each subsequent action involves using items or interacting with people that were introduced before to solve the next problem.
Another paper that inspired us is (7) *Plan-and-write: Towards better automatic storytelling*, where the researchers are introducing a solution to open-domain automated story generation where their system can generate a story given a topic by first planning a storyline and then generating this storyline through using the ROCStories dataset. Although we are using a GPT-2 model as we covered in class to train rather than the Seq2Seq model this paper is using, this is still a useful source to understand how we can generate

WikiHow article games given the topic of the article for the second half of our project.
Another related research in progress is being conducted by two PhD students, Veronica Qing Lyu and Harry Zhang, at the University of Pennsylvania which similarly involves extracting WikiHow article steps and determining the closest choice options for each step. They have provided us their dataset of WikiHow articles in JSON format (that we explain further in the Dataset section) which we used as our main dataset for our project.

DATASET

The dataset we are using consists of articles scraped from the WikiHow website and put into JSON format. Each JSON file contains:

title Title of the article

url Article URL

title_description Short description of the article

category_hierarchy Category tree the article belongs to, listed from most general to most specific

method/part Groupings of steps for different methods (optional)

steps Ordered list of steps with step headline, description and image url for each step

author Author of the article

time_updated Date of last update

n_views Number of views

rating Number of votes, helpful percentage

related_articles List of related article titles

tips_warnings Tips and warnings related to the article

QAs Questions about the article and their answers

refs List of references (optional)

Among these fields, we are only using the title, title description, step headlines, step descriptions and category hierarchy for building the puzzle. The title of the article is used for determining the goal of the puzzle. The title description is used in the beginning of the game to provide puzzle context. The step headlines are used for determining the list of commands the player needs to enter in order to solve the puzzle. The step descriptions are used to let the player know what happens when the corresponding correct command is executed. The categories are used for finding articles similar to the game article and providing the player the list of some categories to play a game from if they do not want to choose the game themselves.
One difficulty we faced with this dataset is that not every article has all the information available. For instance, some articles do not have any categories listed or their steps are not contained in the method/part section. In order to handle the lack of information, we included additional checks in our implementation to check if the steps are contained anywhere else in the JSON, or find similar articles based on the title instead of category. Another technical difficulty was making use of the 'Related Articles' information. Since every JSON file only contains the titles of related articles but not their file names, we instead followed a different approach that consists of using universal embeddings to find similar articles from the dataset, which we explain further in the Methods section.
Apart from the technical challenges, another difficulty we encountered is more related to the nature of articles: not all WikiHow articles are goal-oriented, meaning not every article has steps that have an ordered relationship with each other. One example for such non-puzzle articles is 'How to Say Yes in Different Languages'(say). In this article, each step corresponds to saying yes in a different language where we cannot create a puzzle with hierarchical steps. Although this requires further work, we solved

this problem by simply excluding such articles from the dataset if we ran into one for the moment. Some WikiHow articles have multiple methods where each method has their own steps; however, these methods are not independent from each other every time. Therefore, we decided to use all methods together and create an hierarchy between steps such that a step from an earlier method is considered an earlier step compared to one from a later method.

METHODS

We had to implement the following methods for the game creation:

JSON Parser: The JSON parser simply takes in the file path of the WikiHow article and extracts the title, title description, category hierarchy, step headlines and descriptions. If an article has separate methods, the steps and their descriptions are taken from methods. Otherwise, the steps are read from the separate steps field. Once the step headlines are extracted, they are formatted such that they do not contain unnecessary punctuation or white space before they are passed to imperative detection.

Imperative Detection: After extracting steps from the JSON file, we put each identified step through our imperative detection component. Imperative detection consists of multiple parts: tokenizer, POS-tagging and imperative test.

1. Tokenizer: We create tokens by splitting each step headline by white space, punctuation and other non-alphanumeric characters are not included in tokens.
2. POS-Tagging: We use the Part-of-speech tagging library from *Natural Language Toolkit* to assign a tag to each token based on their definition and context.
3. Imperative Test: Once the POS-tags are generated, we send the list of token-tag tuples to our imperative test. The tokenized sentence must satisfy one of the conditions listed in order to be considered imperative:

    (a) The first token has the POS-tag ”VB” (verb, base form)
    (b) The second token has the POS-tag ”MD” (modal)
    (c) The first token does not have ”VB” POS-tag: for this scenario, we check whether there exists a token with both noun and verb meanings. We use the Wordnet library from Natural Language Toolkit to identify the synonyms. Each synonym returned is tagged with their context (noun/verb) and the synonyms are ordered from most common to least common. We take the number of the synonyms returned and check whether there exists a verb synonym written the exact way as the noun form in the first half of the synonym list. If we find such synonym, we mark the sentence as imperative. There two possible locations to check for:

        i. If the first token has the POS-tag ”RB” (adverb), we check if the second token has a verb synonym
        ii. If the first token has the POS-tag ”NNP” (proper noun, singular) or another noun tag, we check if this token has a verb synonym

Category Separation: We picked a subset of categories we would like to provide a game experience from. We identified 31 sub-categories that are extracted from the category hierarchies of articles and split them into 10 main categories. These categories are:

1. **School:** College/University, School Stuff
2. **Youth:** Youth Dating, Youth Culture, Social Interactions for Youth, Social Events for Youth
3. **Work:** Job Attendance, Socializing at Work
4. **Animals:** Dogs, Cats, Birds
5. **Fun:** Parties, Toys, Music, Drinks, Games, Fun Activities, Halloween
6. **Home and Family:** Parents, Weddings, Neighbors
7. **Pop Culture:** Fandom, Celebrities
8. **Relationships:** Single Life, Dating, Relationship Issues
9. **Friendship:** Friends, Traveling with Companions, Nicknames
10. **Appearance:** Fashion, Looking Good

We created a dictionary for categories such that each sub-category contains a list of JSON files from that sub-category. We determined these lists by running a separate script on our main dataset and querying for one sub-category each time. These category options are presented to the player when the program is run for them to select a category to play from, if they do not choose to upload a game file.

Embeddings: We needed to compute the similarity of the text entered by the user and the correct choice. To achieve this we used embeddings. We tried to compare the similarity of the entered commands to the list of choices the user has.
To compute the embeddings of the sentences we tried averaging the word embeddings obtained from pre-trained GloVe vectors. We also tried InferSent, a sentence embedding library. Both these libraries did not give satisfactory results. We ended up using Universal encoding, a sentence embedding library developed by Google.

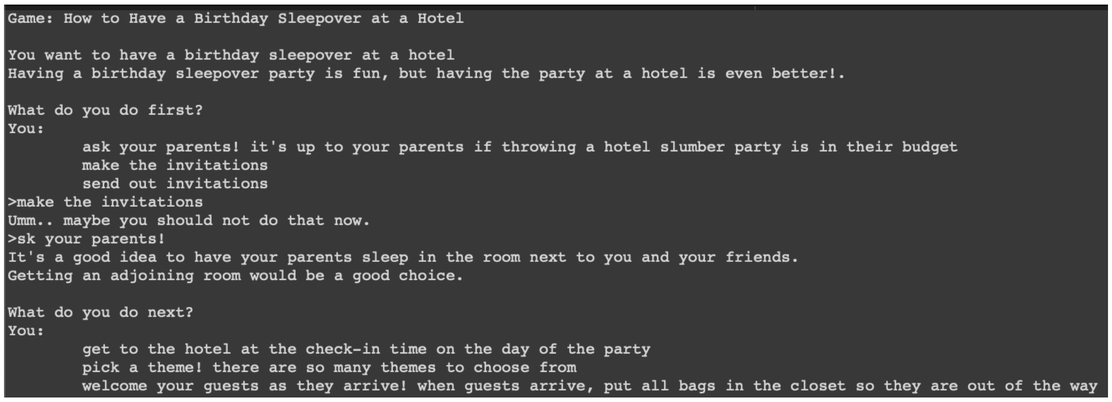
Sentence to Third Person: We want to be able to convert the first person statements entered by the user to third person. So that when the user enters - "I pick up the trash", the prompt says -"You pick up the trash".

Descriptions: To make the game more interactive we had to include responses to the players commands. As well as descriptions of the game to make the user understand the goal.
In the WikiHow dataset, the steps has two components- headlines and descriptions. The headline is the outline of the step whereas the description is comprehensive explanation with tips or warnings. We converted the step description to third person and included that to create interactive explanation. We used the title description in a similar way to explain the game to the user.
In case of a wrong answer we had list of prompts to select the response from.

Wrong Choices: For better game experience we wanted to provide the player with confusing(related to topic) wrong choices (Figure 1). For getting wrong choices that are similar to the article we first extract similar articles to the given article. This is achieved by using word embeddings of the title and finding titles with similar word embeddings. We pre-processed the title embeddings into a magnitude file to make the similarity calculation easier. To remove noise, we finally do a category check. We check whether the similar article has at least one common category with the given article. We then combine the steps from these articles along with the future steps from the articles to select the wrong choices.

To further improve wrong choices, for each step we check that the similarity is above a minimum threshold. To ensure that the wrong choices are not same as the correct choice(related articles have related steps), we make sure that the similarity is below a lower threshold.

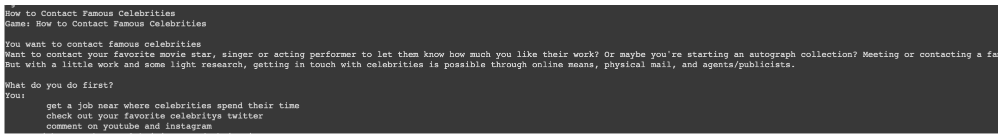
Figures 2 and 3 show wrong choice examples from 'How to Contact Famous Celebrities'. We can see in



```
Game: How to Have a Birthday Sleepover at a Hotel

You want to have a birthday sleepover at a hotel
Having a birthday sleepover party is fun, but having the party at a hotel is even better!.

What do you do first?
You:
        ask your parents! it's up to your parents if throwing a hotel slumber party is in their budget
        make the invitations
        send out invitations
>make the invitations
Umm.. maybe you should not do that now.
>sk your parents!
It's a good idea to have your parents sleep in the room next to you and your friends.
Getting an adjoining room would be a good choice.

What do you do next?
You:
        get to the hotel at the check-in time on the day of the party
        pick a theme! there are so many themes to choose from
        welcome your guests as they arrive! when guests arrive, put all bags in the closet so they are out of the way
```

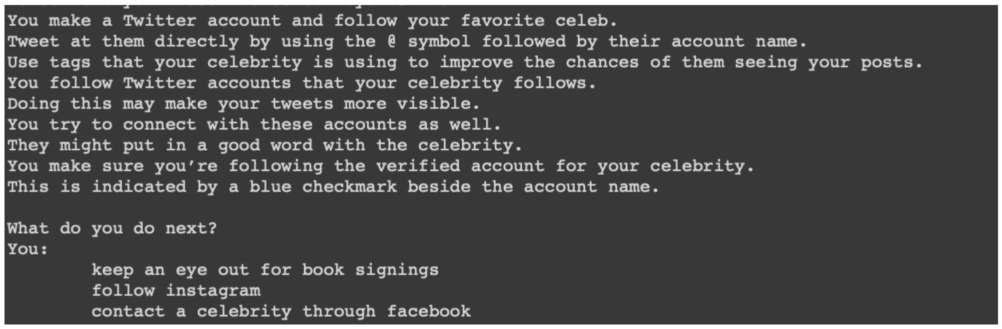
Figure 1. Example for wrong choices from 'How to Have a Birthday Sleepover at a Hotel'

the examples that the wrong choices provided were confusing and relate to the topic.



```
How to Contact Famous Celebrities
Game: How to Contact Famous Celebrities

You want to contact famous celebrities
Want to contact your favorite movie star, singer or acting performer to let them know how much you like their work? Or maybe you're starting an autograph collection? Meeting or contacting a fa
But with a little work and some light research, getting in touch with celebrities is possible through online means, physical mail, and agents/publicists.

What do you do first?
You:
        get a job near where celebrities spend their time
        check out your favorite celebritys twitter
        comment on youtube and instagram
```

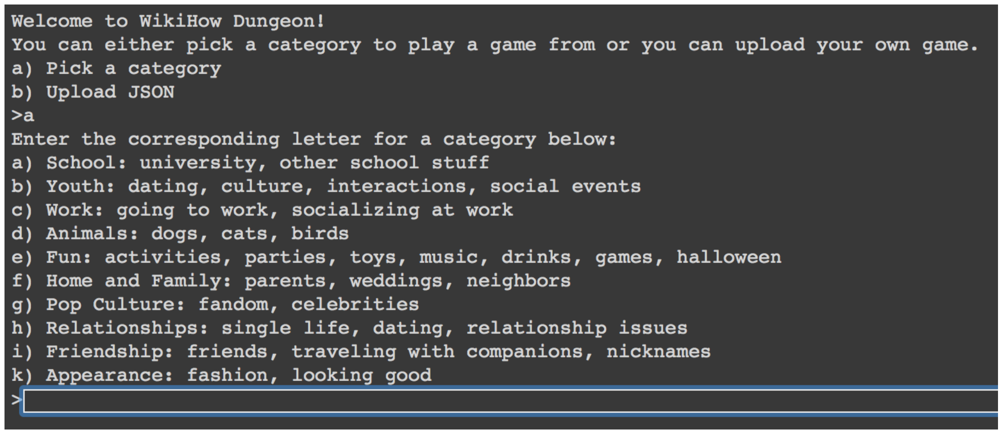
Figure 2. Example for wrong choices from ’How to Contact Famous Celebrities’



```
You make a Twitter account and follow your favorite celeb.
Tweet at them directly by using the @ symbol followed by their account name.
Use tags that your celebrity is using to improve the chances of them seeing your posts.
You follow Twitter accounts that your celebrity follows.
Doing this may make your tweets more visible.
You try to connect with these accounts as well.
They might put in a good word with the celebrity.
You make sure you’re following the verified account for your celebrity.
This is indicated by a blue checkmark beside the account name.

What do you do next?
You:
        keep an eye out for book signings
        follow instagram
        contact a celebrity through facebook
```

Figure 3. Example for wrong choices from ’How to Contact Famous Celebrities’

Game Loop: We built an interactive game loop, which is the compilation of all game components into the main game experience with text-based interface. The player has two options: they can either pick a category to play a game from or they can upload an article JSON file. If they choose the former, we provide a menu of categories to pick from (Figure 4). Once a category is selected, we pick a random sub-category from the selected category. After the sub-category is selected, we select a random article from this sub-category and check if the article has at least 1 imperative step. We keep picking a random article until we find one with imperative steps to play with. If the player chooses to upload a file instead, they are prompted to select the file in Finder. If the selected file is invalid, they are prompted to try again. A valid game file must be of JSON format and it must contain title and method/part among its fields.



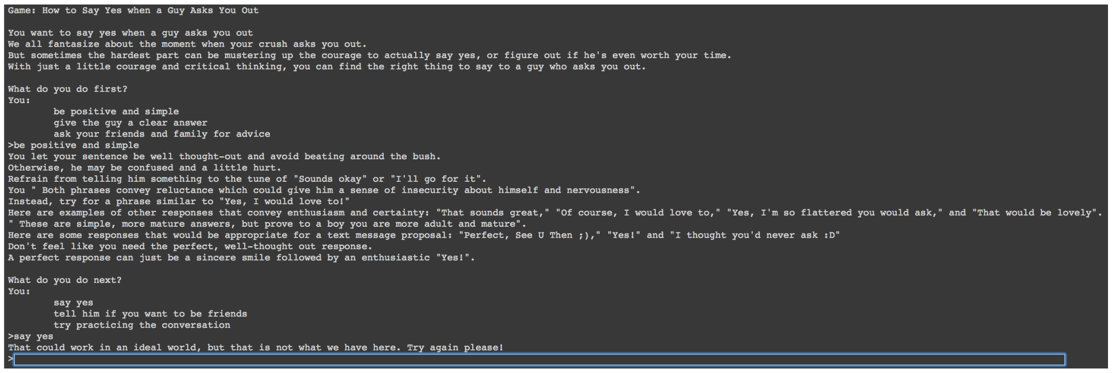
```
Welcome to WikiHow Dungeon!
You can either pick a category to play a game from or you can upload your own game.
a) Pick a category
b) Upload JSON
>a
Enter the corresponding letter for a category below:
a) School: university, other school stuff
b) Youth: dating, culture, interactions, social events
c) Work: going to work, socializing at work
d) Animals: dogs, cats, birds
e) Fun: activities, parties, toys, music, drinks, games, halloween
f) Home and Family: parents, weddings, neighbors
g) Pop Culture: fandom, celebrities
h) Relationships: single life, dating, relationship issues
i) Friendship: friends, traveling with companions, nicknames
k) Appearance: fashion, looking good
>
```

Figure 4. Initial game loop interface

We extract the list of commands, descriptions, title and title description from the selected article. We use the extracted title to find similar articles and extract their steps as possible wrong choice options throughout the game.

The game loop keeps track of the current step index to determine whether the player has reached the end of the game and which commands the player has not executed successfully yet. The commands that are for future steps are used as wrong choice option candidates for a given step along with the steps from similar articles. We provide three options to the player at each step (Figure 5) where only one of the

options is correct.
When the player enters an input, we compare it with all the commands from the article, excluding the ones the player has already executed in previous steps, by using vector similarity. If the player input is closest to the correct choice, the step description is printed and the next step options are presented. Otherwise, the player is warned with a negative statement and is asked to try again (Figure 5).
If the player reaches the end of the game, we print *You reached the end of the game* to notify and the loop terminates. The player can alternatively choose to quit mid-game, which requires them to enter **quit** as their input.



```
Game: How to Say Yes when a Guy Asks You Out

You want to say yes when a guy asks you out
We all fantasize about the moment when your crush asks you out.
But sometimes the hardest part can be mustering up the courage to actually say yes, or figure out if he's even worth your time.
With just a little courage and critical thinking, you can find the right thing to say to a guy who asks you out.

What do you do first?
You:
        be positive and simple
        give the guy a clear answer
        ask your friends and family for advice
>be positive and simple
You let your sentence be well thought-out and avoid beating around the bush.
Otherwise, he may be confused and a little hurt.
Refrain from telling him something to the tune of "Sounds okay" or "I'll go for it".
You " Both phrases convey reluctance which could give him a sense of insecurity about himself and nervousness".
Instead, try for a phrase similar to "Yes, I would love to!"
Here are examples of other responses that convey enthusiasm and certainty: "That sounds great," "Of course, I would love to," "Yes, I'm so flattered you would ask," and "That would be lovely".
" These are simple, more mature answers, but prove to a boy you are more adult and mature".
Here are some responses that would be appropriate for a text message proposal: "Perfect, See U Then ;)," "Yes!" and "I thought you'd never ask :D"
Don't feel like you need the perfect, well-thought out response.
A perfect response can just be a sincere smile followed by an enthusiastic "Yes!".

What do you do next?
You:
        say yes
        tell him if you want to be friends
        try practicing the conversation
>say yes
That could work in an ideal world, but that is not what we have here. Try again please!
>
```

Figure 5. Gameplay with multiple choice options and step descriptions. The player is warned with a negative statement if they pick the wrong option.

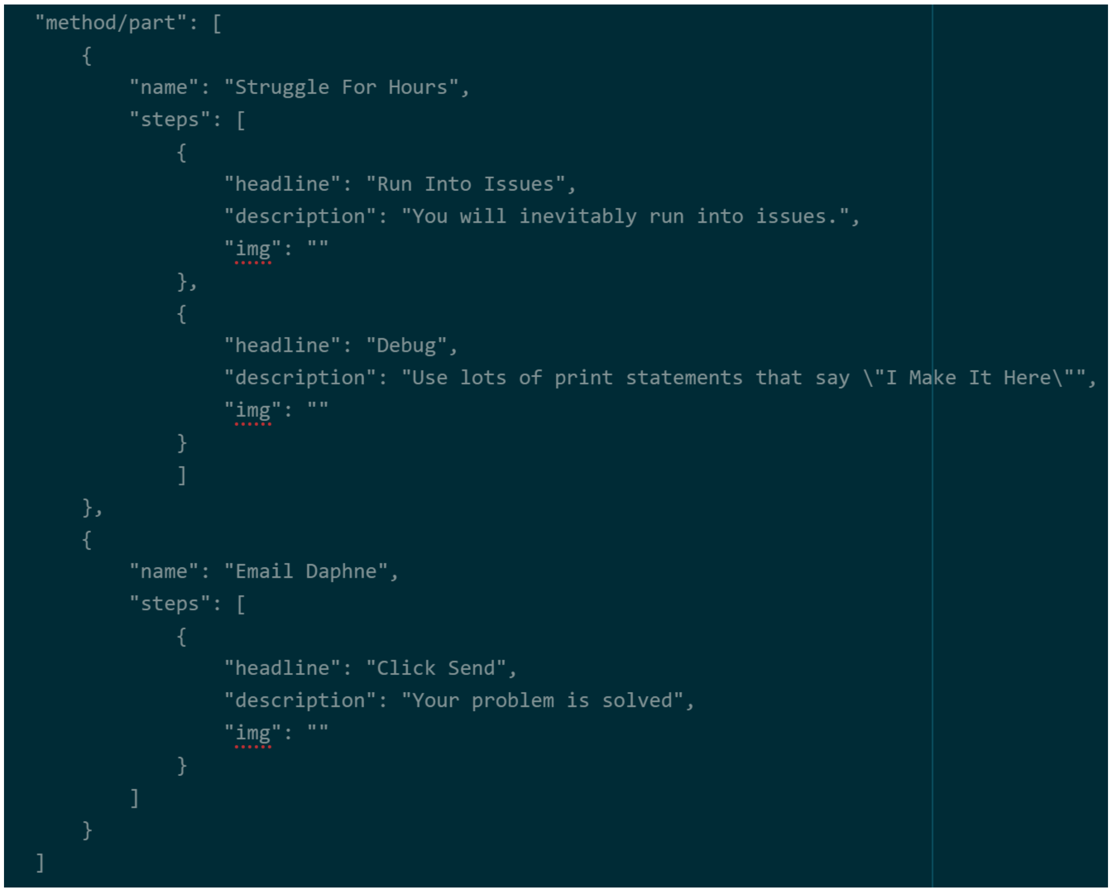
GPT-2 Finetuning: We trained and fine-tuned a GPT-2 model on the task of generating new wikihow articles. Based on the code from Homework 4, and using huggingface's transformers, the model itself took text files (.txt) for each of the training, validation, and test datasets, and fine-tuned itself in order to generate similarly structured text files. The training data comprised 80 percent of the dataset, and both the validation set and test set were 10 percent each.

The language model was run and fine-tuned for one epoch, for a total of 3603 iterations, using batch size of 2 for both the training and evaluation stages, a block size of 128 and 5 gradient accumulation steps. The generations were created using with temperature scaling 0.5, k=50 top-k sampling, with no nucleus sampling or repetition penalty. Each generation was length 1000 since any larger ran into issues with Pytorch dimensions. For a given prompt, 5 total sequences were generated, and the sequence with the lowest individual perplexity was chosen as the final output. The prompt for a given set of generations was the title of the article.

Pre/Post-Processing JSON for Training: Of particular interest was the chosen method for pre and post-processing the wikihow article data. One method could have been simply taking the .json file for a given article and converting it to .txt and training the model on that. However, given how the function for converting .txt to .json is often finicky and requires precise formatting to work, this method was not chosen since even a single misplaced bracket in the generation could ruin the conversion back to .json. Instead, a given .json was converted to an easily parse-able piece of text that would also make it easier for the model to learn the structure of, and to convert back into a valid .json file.

The important information for a wikihow article consists of the ”title” field, and the ”method/part” field, which itself is a list of dictionaries, each of which with the fields ”name” and ”steps”, where ”name” refers to the subtitle used to group a list of steps. ”steps” was also a list of dictionaries, each representing a particular step in the process with fields for ”headline”, ”description”, and ”img”. As long as this hierarchy was maintained, we could get all the interesting information from a wikihow article and easily convert between .json and .txt.

This was done by first converting the JSON to a Python dictionary. Then, the title, and method/part fields were taken. The title would be associated with the string ”title:”, and each part within method/part would be associated with ”name:” followed by the steps for that given name using ”headline:” and ”description:”. An brief example is shown below of the conversion process (which is the same for .txt to .json but in the reverse direction).



```
"method/part": [
    {
        "name": "Struggle For Hours",
        "steps": [
            {
                "headline": "Run Into Issues",
                "description": "You will inevitably run into issues.",
                "img": ""
            },
            {
                "headline": "Debug",
                "description": "Use lots of print statements that say \"I Make It Here\"",
                "img": ""
            }
            ]
    },
    {
        "name": "Email Daphne",
        "steps": [
            {
                "headline": "Click Send",
                "description": "Your problem is solved",
                "img": ""
            }
        ]
    }
]
```

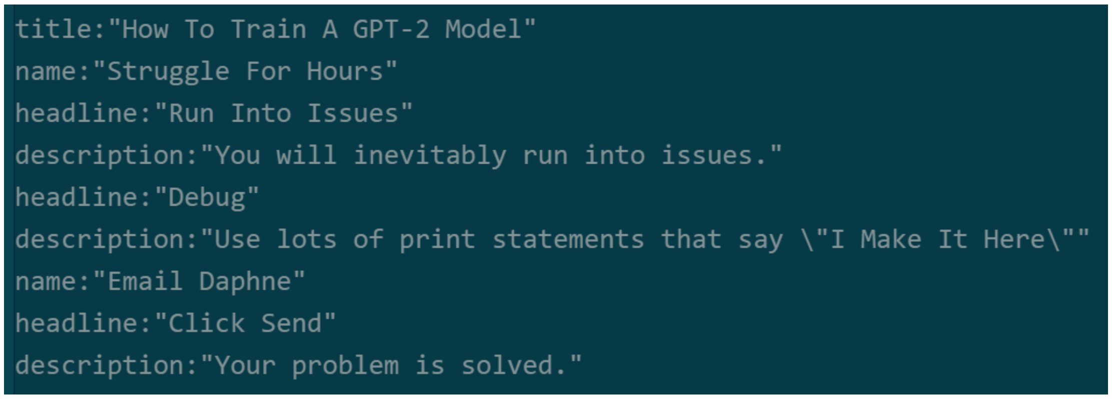
Figure 6. The ”method/part” field of the article titled ”How To Train A GPT-2 Model”

Note: Originally the format was ”<title=”How To Train A GPT-2 Model”>” but GPT-2 had difficulties with HTML-style syntax.

It is also worth noting that since the generations focused on generating the steps and descriptions, category hierarchy, url, author, related articles, and other fields of the original JSON structure were ignored.

EVALUATION

For the game component, our evaluation is based primarily on human judgement and comparison of different models and tools. We have tested the quality of imperative detection, object detection, change from third person to second person, descriptions and wrong choice options by observing the outputs and determining whether they make sense or not.
We ran tests on imperative detection by using three different software (*Stanford CoreNLP*, *Natural Language Toolkit*, *AllenNLP*) with POS-tagging feature to determine which one gives better and more accurate tag results. We identified a list of imperative sentences to be tagged and compared the number of correct results from one software to the other one. We found out that NLTK and AllenNLP were able to identify more imperative sentences compared to Stanford CoreNLP. The results between NLTK and AllenNLP were almost similar. We ended up choosing NLTK for POS-tagging since we later had compatibility issues with AllenNLP.



```
title:"How To Train A GPT-2 Model"
name:"Struggle For Hours"
headline:"Run Into Issues"
description:"You will inevitably run into issues."
headline:"Debug"
description:"Use lots of print statements that say \"I Make It Here\""
name:"Email Daphne"
headline:"Click Send"
description:"Your problem is solved."
```

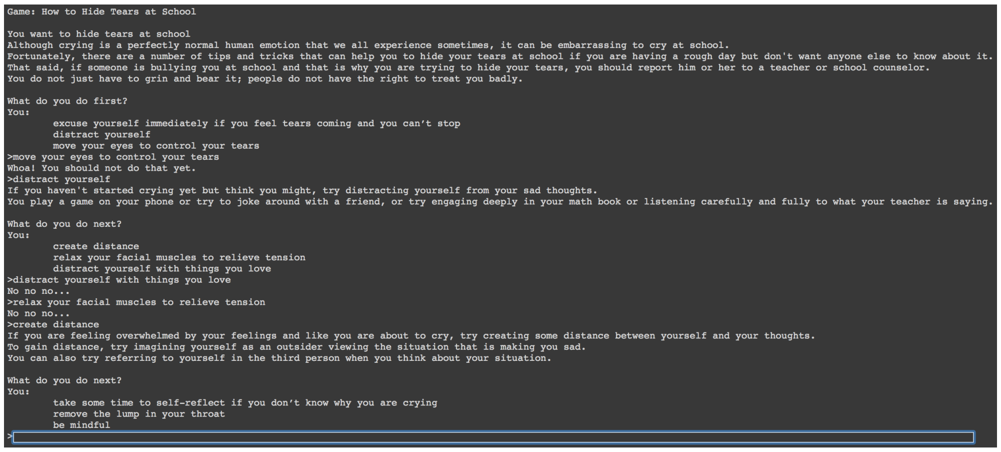
Figure 7. The resulting text representation to be used for training.

For sentence embeddings, we tried GloVe vetors in 300 dimensions, InferSent- a library for sentence embeddings by Facebook Research. We try compare the user input with the list of known commands that the user can choose from. In case of both GloVe and InferSent, we observed that the embeddings do not usually pick the user intended embedding from the list of known commands. But, in case of Universal sentence embeddings, the similarity function is able to correctly identify the intended command in most of the cases.
To evaluate the generations of the GPT-2 model, human judgement was used in order to tell how ”good” the generations were. Some ”metrics” looked at included whether a generation stayed true to the prompt, its ability to create creative generations, whether the text generated made sense or not, and how well it could combine two or more topics mentioned in the prompt.

RESULTS

We tested our program with articles picked from category as well as by uploading existing articles and GPT-2 generated articles.

1. **Article picked from category:** We picked **School** from the category menu and the game loop picked the article ’How to Hide Tears at School’ (tea) from the category dictionary. At each step the player is given 3 options to pick from where one option is correct and the other options are picked from articles related to the game article (Figure 8).

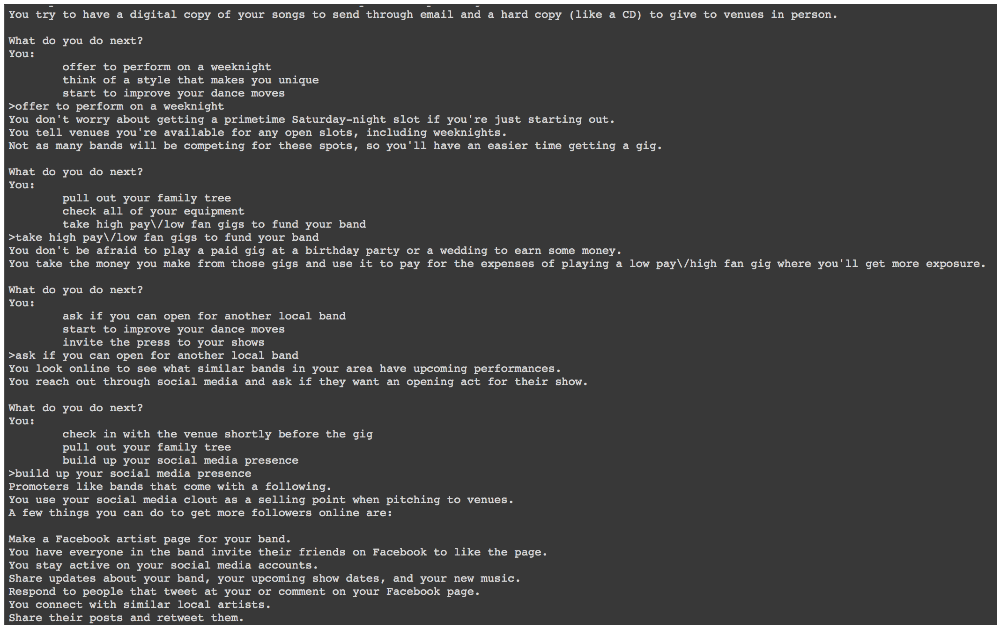


```
Game: How to Hide Tears at School

You want to hide tears at school
Although crying is a perfectly normal human emotion that we all experience sometimes, it can be embarrassing to cry at school.
Fortunately, there are a number of tips and tricks that can help you to hide your tears at school if you are having a rough day but don't want anyone else to know about it.
That said, if someone is bullying you at school and that is why you are trying to hide your tears, you should report him or her to a teacher or school counselor.
You do not just have to grin and bear it; people do not have the right to treat you badly.

What do you do first?
You:
        excuse yourself immediately if you feel tears coming and you can’t stop
        distract yourself
        move your eyes to control your tears
>move your eyes to control your tears
Whoa! You should not do that yet.
>distract yourself
If you haven't started crying yet but think you might, try distracting yourself from your sad thoughts.
You play a game on your phone or try to joke around with a friend, or try engaging deeply in your math book or listening carefully and fully to what your teacher is saying.

What do you do next?
You:
        create distance
        relax your facial muscles to relieve tension
        distract yourself with things you love
>distract yourself with things you love
No no no...
>relax your facial muscles to relieve tension
No no no...
>create distance
If you are feeling overwhelmed by your feelings and like you are about to cry, try creating some distance between yourself and your thoughts.
To gain distance, try imagining yourself as an outsider viewing the situation that is making you sad.
You can also try referring to yourself in the third person when you think about your situation.

What do you do next?
You:
        take some time to self-reflect if you don’t know why you are crying
        remove the lump in your throat
        be mindful
>
```

Figure 8. Article from category gameplay

2. **Upload existing article:** We used an example article ’How to Get Gigs for Your Band’ (gig) to upload and play. Just like a game from category, at each step the player is given 3 options to pick

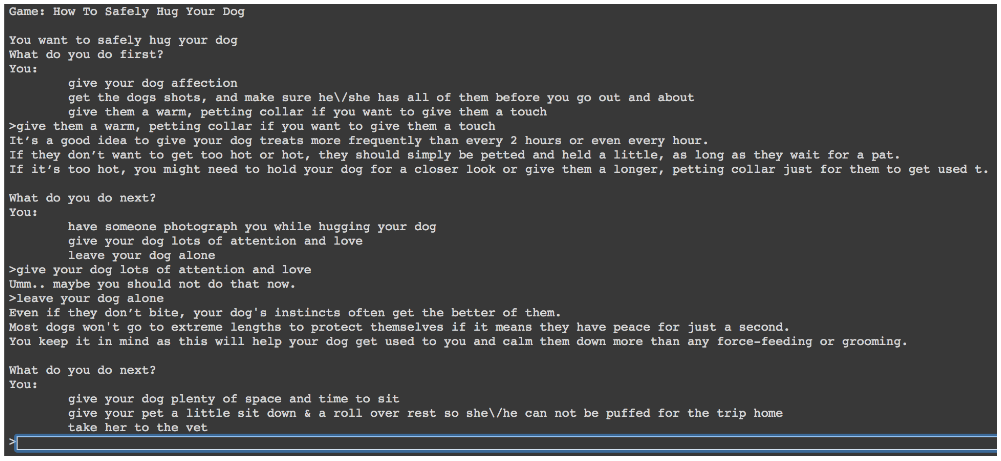
from where one option is correct and the other options are picked from articles related to the game article (Figure 9).



```
You try to have a digital copy of your songs to send through email and a hard copy (like a CD) to give to venues in person.

What do you do next?
You:
        offer to perform on a weeknight
        think of a style that makes you unique
        start to improve your dance moves
>offer to perform on a weeknight
You don't worry about getting a primetime Saturday-night slot if you're just starting out.
You tell venues you're available for any open slots, including weeknights.
Not as many bands will be competing for these spots, so you'll have an easier time getting a gig.

What do you do next?
You:
        pull out your family tree
        check all of your equipment
        take high pay\/low fan gigs to fund your band
>take high pay\/low fan gigs to fund your band
You don't be afraid to play a paid gig at a birthday party or a wedding to earn some money.
You take the money you make from those gigs and use it to pay for the expenses of playing a low pay\/high fan gig where you'll get more exposure.

What do you do next?
You:
        ask if you can open for another local band
        start to improve your dance moves
        invite the press to your shows
>ask if you can open for another local band
You look online to see what similar bands in your area have upcoming performances.
You reach out through social media and ask if they want an opening act for their show.

What do you do next?
You:
        check in with the venue shortly before the gig
        pull out your family tree
        build up your social media presence
>build up your social media presence
Promoters like bands that come with a following.
You use your social media clout as a selling point when pitching to venues.
A few things you can do to get more followers online are:

Make a Facebook artist page for your band.
You have everyone in the band invite their friends on Facebook to like the page.
You stay active on your social media accounts.
Share updates about your band, your upcoming show dates, and your new music.
Respond to people that tweet at your or comment on your Facebook page.
You connect with similar local artists.
Share their posts and retweet them.
```

Figure 9. Uploaded existing article gameplay

3. **Upload generated article:** We used a generated article with the generated title 'How to Safely Hug Your Dog' to upload and play. Since the generated articles do not have category hierarchies, we do not filter the similar articles we get by category. Just like existing articles, the generated article is passed through all the necessary components and it is passed to the game loop afterwards. The gameplay is also same such that at each step the player is given 3 options to pick from where one option is correct and the other options are picked from articles related to the game article (Figure 10). The only difference between generated and existing articles is a generated article doesn't have title description and category list.



```
Game: How To Safely Hug Your Dog

You want to safely hug your dog
What do you do first?
You:
        give your dog affection
        get the dogs shots, and make sure he\/she has all of them before you go out and about
        give them a warm, petting collar if you want to give them a touch
>give them a warm, petting collar if you want to give them a touch
It's a good idea to give your dog treats more frequently than every 2 hours or even every hour.
If they don't want to get too hot or hot, they should simply be petted and held a little, as long as they wait for a pat.
If it's too hot, you might need to hold your dog for a closer look or give them a longer, petting collar just for them to get used t.

What do you do next?
You:
        have someone photograph you while hugging your dog
        give your dog lots of attention and love
        leave your dog alone
>give your dog lots of attention and love
Umm.. maybe you should not do that now.
>leave your dog alone
Even if they don't bite, your dog's instincts often get the better of them.
Most dogs won't go to extreme lengths to protect themselves if it means they have peace for just a second.
You keep it in mind as this will help your dog get used to you and calm them down more than any force-feeding or grooming.

What do you do next?
You:
        give your dog plenty of space and time to sit
        give your pet a little sit down & a roll over rest so she\/he can not be puffed for the trip home
        take her to the vet
>
```

Figure 10. Uploaded generated article gameplay

4. **Looking at generated articles:** The articles we will briefly discuss are: "How To Safely Hug Your Dog", "How To Steal Your Friend's 401k", and an empty prompt for generation.

”How To Safely Hug Your Dog” is a good example of a generation that appropriately stays on topic. In particular, the instructions are about dogs, and generally taking care of your dog and their needs. Headlines like ”Give your dog a gentle touch”, and ”Be attentive to what your dog needs” represent that the model understood that this article should be about being positive and affectionate with your dog. While the steps themselves do not lead through the process of hugging your dog, it is consistent with the overall theme of the topic, gave relatively creative descriptions to the individual headlines, and mostly made sense within itself. It could be improved by having a more direct path to the end goal of hugging your dog (though it has many headlines about gently touching the dog and giving it affection).

”How To Steal Your Friend’s 401k” is a prompt/title meant to test for some bizarre generations. A truly great model would be able to string together a set of instructions to somehow ”steal” the 401k of an individual. Arguably, this is a lot to ask for, even with the most advanced of models. However, there was some value to the generation. In particular, the model focused heavily on meeting with your friend, and it would not be surprising if the first step of stealing from someone is to meet with them. However, the model then strayed from the original goal and focused more on aspects of ”winning over” your friend. In fact, searching for similar articles showed that this generation was close to ”How To Admit Your Crush”. Given the difficulty of the ask (and the lack of training data on 401k theft), it makes sense that this generation did not stay on topic. The descriptions themselves made sense for their respective headlines, but it was clear that the model could not combine the two topics ”friends” and ”stealing 401k’s” and opted to generate for the topic it had the most data on.

The final prompt was an empty string, so there was no actual title generated for this last generation. This was done to see what the model would do without instructions. This article generated names for sections such as ”Discussing Abortion While Providing Care” and ”Working with support Groups”. Essentially, the model generated an article on how to deal with getting an abortion and seeking proper care during and after the process. Overall it was a well written generation that made a lot of sense within its chosen topic, despite being a little jolting to read without proper context.

DISCUSSION

We created this game as an interactive experience based on ”How to” articles from WikiHow. This project was a great learning experience where we worked on NLP based techniques, sentence embeddings and also fine-tuning a transformer model.

While we implemented many techniques, we also tried object detection to identify the available objects within the game for the user to interact with. We tried by extracting the list of concrete nouns in the title and title description. This was not successful, as the method resulted in many non-interactable or non-available objects.

While working on the project, we initially invested a lot of time in perfecting the methods like imperative detection to make sure the game component was viable even without the transformer.

We would have liked to work more on differentiating between puzzle like and non-puzzle like WikiHow articles. We could have implemented state tracking with the help of object detection. If the user makes a wrong choice, we pick one of the predefined generic messages to state that it was a wrong choice. We could use the context to generate a more specific error message.

Another interesting extension to this project would be addition of obstacles to the puzzles. WikiHow articles have question-answers and tips-warnings sections. Using these we could identify potential obstacles that could be used to make the game more challenging.

We also struggled with debugging and training the GPT-2 model for a while, and lots of time was required to make the model work on an example dataset (presidential speeches) before applying our own. Additionally, the pre/post-processing of JSON files proved to be a hefty endeavor that we were not expecting. However, the model benefitted greatly from doing so. That being said, given more time, we absolutely would invest more time in improving the GPT-2 generations to be more creative and robust in order to make the wikihow articles generated cover a larger range of topics, and create more intelligent responses to very specific or bizarre topics that the user could input in order to improve the experience of the game component and vastly increase the replayability.

ATTRIBUTION

Gizem Dal: Implemented the JSON parser, step headline trimming and imperative detection components after testing various POS-tagging libraries. Integrated category separation to create a dictionary of articles in selected categories. Built the interactive game loop which is the compilation of all game components into the main game experience with text-based interface and ran tests for various kinds of articles.

Sanjeevini Ganni: Implemented Universal sentence embeddings after experimenting with different embeddings, changing sentence to third person, generating descriptions of title and steps, object detection, finding similar articles based on sentence embeddings and further selecting the steps for wrong choices.

Andrew Martin: Implemented the heavy pre/post-processing of the JSON files in the database, allowing the model to more easily train and create outputs that could be used for valid .json files. Fine-tuned the GPT-2 model using huggingface's transformer for the task of wikihow article generation. Assisted in creating some helper functions for finding similar articles.

ACKNOWLEDGEMENTS

We would like to thank Veronica Qing Lyu and Harry Zhang from *University of Pennsylvania* for providing their WikiHow dataset and helping us understand how to determine the closest wrong choice options for each step.

REFERENCES

[aid] Ai dungeon.
[gig] How to get gigs for your band.
[tea] How to hide tears at school.
[say] How to say yes in different languages.
[wik] Online database.
[6] Clark, E., Ji, Y., and Smith, N. A. (2018). Neural text generation in stories using entity representations as context. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2250–2260.
[7] Yao, L., Peng, N., Weischedel, R., Knight, K., Zhao, D., and Yan, R. (2019). Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7378–7385.