Robin Osekowsky
LIGN 6: Styler
Final Project

Hexagon

Improving Settlers of Catan

One of the biggest obstacles to starting a new board game is learning and applying the rules appropriately. Many rule books are poorly written, missing pieces, or confusing. Hexagon aims to alleviate these pains, starting with Settlers of Catan. Players using Hexagon will be able to query rules specific to their game and expansion, without shuffling through rule books. Players can instead get to the point of their query rather than spending time trying to translate explanatory sentences to goal-oriented language. Similarly game-play is difficult to track in Catan, and here again Hexagon can help. Since Settlers of Catan is primarily a strategic game, tracking and revealing all information about the game would take the fun out of it, but Hexagon is designed to simplify tracking (revealed) victory points, as counting up a players points on the board can be confusing. Hexagon will track settlements, cities, achievement cards, and expansion-specific victory points like settling an island so a player can query their current win status and plan accordingly. Named 'Hexagon' for the game's iconic board shape, this virtual assistant will make Settlers of Catan much more accessible to new and returning players.

Wake-Word Recognition

The feature that the Virtual Assistant track the progress of the game requires that the system hears and processes information that will not be directed to it. Hexagon will need to always be listening, distinguish between different voices at varying distances, and may respond to confirm facts that were not directed to it. This means that the system will process the conversation looking for particular speech patterns as part of its game-tracking (where the underlined terms are what we are tracking for):

- "I am placing 4 more roads, that gives me 'Longest Road'."
- "I am buying a settlement."

If this is done responsibly and with regard to privacy, there is no reason to store audio data that does not directly match the patterns we are looking for, or analyze that same audio for other patterns or keywords. Additionally since the system is only designed to run while the game is active, most of the audio that would be recorded would be game-specific, further reducing (but not eliminating) privacy concerns.

When the system is called upon to respond to something directly, the trigger or wake-word will be its name: “Hexagon”, which can then be followed by a query, such as “how close is Georgia to winning?”, or “how many points do you get for settling an island?” The name of the system is an unusual word to use in conversation, and uses unusual phonetic patterns, so as to minimize accidental activation.

Recording

Recording will be comparable to current systems in terms of difficulty and goals. Current systems’ specs place sampling rate somewhere around 24kHz for speech. Since our use cases are all only dialog, and the virtual assistant does not need to handle music or other sounds for the game, 24kHz should suffice for our sampling rate. Like Alexa, or Google Assistant, this assistant will need to filter background noise, and allow for interaction from multiple distances/locations in the room.

Transmission to Servers

Fortunately, since Settlers of Catan is a board game with a lot of pieces that requires a large space to play in, our virtual assistant need not be particularly portable. It is safe to assume that the game will be played in a home or location where WiFi is accessible (eg. a game lounge), so the assistant should not need to function without WiFi connection. Otherwise, transmitting data to and from the server should be very similar to existing systems. In an average game, Hexagon will likely be questioned directly sparsely, and the game itself will not last more than a few hours. Do to these low-frequency primary use cases, Hexagon will prioritize exactness of the data over reducing data usage. In other words, because we are not using the assistant for an extremely long period of time, or asking much of the system in terms of response data, we would rather the assistant got the questions and answers correct than that it conserved data usage by reducing the quality of the audio.

ASR Processing

The language used in Settlers of Catan is largely standard English, with a few grammar and word frequency changes. The vocabulary of the game is built on simple but not often used English words like ‘ore’ and ‘settlement’. Since current systems can already process these words correctly with decent accuracy even when used in sentences with odd grammatical structure, Hexagon would need minimal updates in order to accomplish the necessary ASR processing tasks, tweaking the expected frequencies of these words. I tested this functionality with Catan-specific sentence structure:

Transcription correct:

- How close is Nicole to winning?

- How many points do you get for settling an island?
- What are the rules for choosing starting positions?
- How does the robber work?
- Can you build a ship on a coastline?
- How many roads do I need to get Longest Road?

Transcription Buggy

- How do I get ore?
    - Transcribed “ore” as “or”, so training described below would help alleviate this

Settlers of Catan is an international board game, so in theory Hexagon should be extensible to every language that has its own version of Catan. This would complicate the training stage, but like most virtual assistants, Hexagon would have various language modes to switch between listening/parsing patterns. It would not really add any extra benefit to expect multiple languages to be used within the same game, so we could split the ASR training by language, and allow the players to set the mode in which they would like to play. Within a language though, we would have to take multiple accents into account, which would complicate training a lot. We would simplify this processing step by using the jargon of Catan to give us better expectations of the words being said. Some of the common words/phrases include:

- Ore (AO R)
- Wheat (W IY T)
- Brick (B R IH K)
- Sheep (SH IY P)
- Wood (W UH D)
- Settlement (S EH T AH L M AH N T)
- City (S IH T IY)
- Robber (R AA B ER)
- Victory point (V IH K T AH R IY . P OY N T .)
- Knight (N AY T)
- Road (R OW D)

Many of these words are homophones for other more common words (‘or’ and ‘would’), and others are very similar to common words (‘sheep’ and ‘she’), so we would train Hexagon more on sentences that include these situation-specific terms more than the common words, so that the default would be to expect these, rather than their similar counterparts, and it could fuzzy-match with some accuracy.

The last thing we would need to consider is using pronouns rather than outright names with the virtual assistant. If I asked "How many points do I need to win?”, I would hope that Hexagon would understand who was asking the question and answer me relative to my current victory point state. Unfortunately, due to the choice of making game session separate from one another so the system does not have to remember anything about the state of the world between sessions, training Hexagon to recognize voices with high accuracy would be very difficult. Even if players identified themselves at the beginning of the game, it would be difficult to provide enough data that Hexagon could distinguish between each vocal pattern, and it would need to confirm the speaker (or referenced person) every time a pronoun was used.

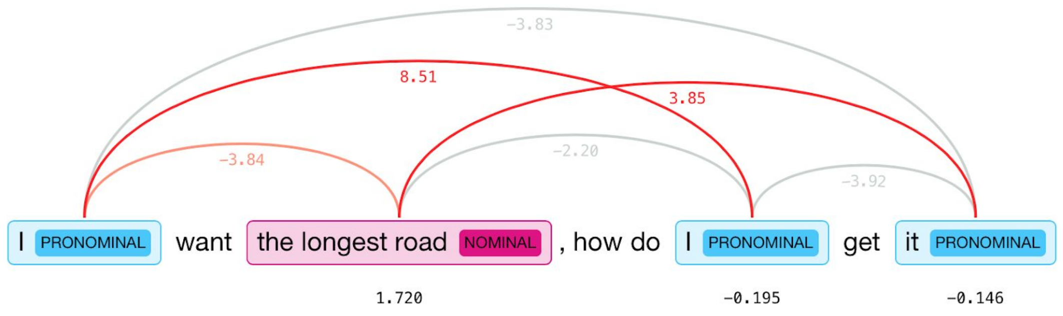
Linguistic processing of the text data

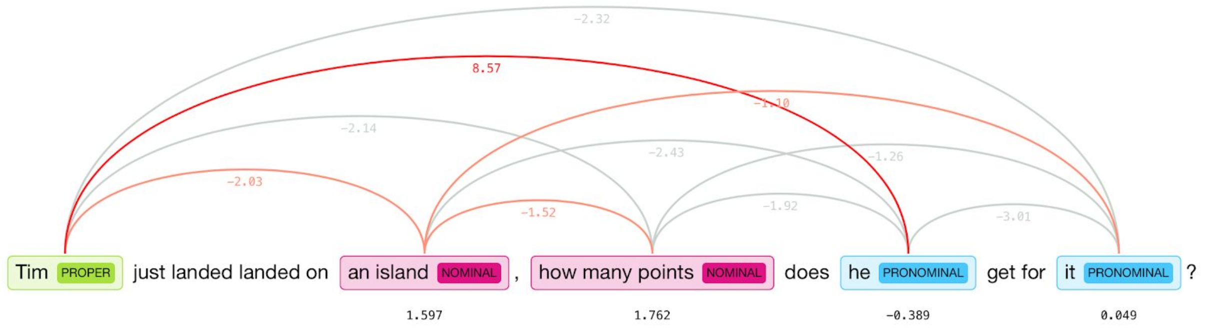
Overall, current systems can parse current sentences well, but do make mistakes related to situational vocabulary. For example, the CMU parser misparses " What are the rules for choosing starting positions?" as

```
    (S What
        (S are
            (NP (NP (NP the rules)
                    (PP for
                        (S (VP choosing))))
            (VP starting
                (NP positions)))))
```

incorrectly interpreting "starting" as a verb rather than the adjective it is. These issues would be fixed by training the parser on more domain-specific language. Therefore Hexagon would be best trained on a unique corpus. This would minimize confusion by weird usage of normal words and would allow us to focus on domain-specific language. Hexagon would not need to learn a large portion of language as "tweeting", "democracy", and "homework" are not relevant to the game, but we would want the system to learn some fairly complex grammar to both interpret the rules of the game and correctly parse the grammatical structure of a player's question: "If I were to obtain the Longest Road card, would I be in the lead?". This would likely be best done by obtaining an existing corpus (to train grammatical structure generally) and then add the domain-specific grammar to it. To this end, I would use a corpus of conversational data, and add data for the Settlers of Catan rule book as well as some common sentences like the questions and statements discussed elsewhere in this write up. This would cover not only odd vocabulary usage but also odd phrasings like "three wheat".

Coreferencing will be more relevant for the background audio processing Hexagon will do, rather than queries. Most queries involving coreferences will be easily parsable (as tested with Neural Coref. below), but casual conversation does not always follow easy patterns. The expectation for Hexagon will be that the system will ask to clarify (relevant) coreferences in normal conversation rather than trying to parse them in order to be as accurate as possible in its knowledge of the game state.





Meaning Extraction

In terms of complexity, Hexagon's semantic parsing will be no harder than current systems. However, existing systems could not do this task for Hexagon as it requires domain-specific state representations. Common queries and corresponding elements include:

- How close is Nicole to winning?
    - Type: 'state of game'
    - Player: 'Nicole'
    - State of interest: 'number of points'
    - Goal: 'win'
- How many points do you get for settling an island?
    - Type: 'rule reference'
    - Goal: 'points received'
    - Action: 'placing a settlement on island'
- What are the rules for choosing starting positions?
    - Type: 'rule reference'
    - Goal: 'unspecified'
    - Action: 'placing starting settlements'
- How does the robber work?
    - Type: 'rule reference'
    - Goal: 'unspecified'

    - Action: ‘placing the robber’
- Can you build a ship on a coastline?
    - Type: ‘rule reference’
    - Goal: ‘build on coastline’
    - Action: ‘place ship’
- How many roads do I need to get Longest Road?
    - Type: ‘state of game’
    - Player: ‘unspecified’ [this field is necessary and missing information should prompt a clarifying question]
    - State of interest: ‘number of roads laid’
    - Goal: ‘Longest Road card received’

This is where the domain-specific language is particularly useful. The words ‘win’, a player’s name, important card names, and names objects in the game help us determine these fields. It is also useful to know the possible actions in the game (according to the specific rules). Each query in a session, would also have a hidden element field for the game version set at the beginning of the game.

In order to answer these questions, we would need to build an external knowledge representation for Hexagon. This would be necessary for the state-of-the-game representation, but for the rule structure, it would be helpful to build a framework for interpreting the rule text into rules rather than hard-coding anything. With such a framework, if someone ever creates new versions or expansions to the original game, we could just feed the rule text into the server for processing without having to update the programming. Similarly, it would be better to do this on an external server so that each device does not need to update or re-process the same information. The system would have to make many assumptions but do very little inference. Many of the elements of each query are closely linked. Asking “what do I need to get Longest Road?” should produce the same element breakdown as the query above as there is only one way to achieve the intended goal. The system could fill in the blanks like this for many unspecific queries according to its knowledge representation of the rules of the board game.

Response Planning

Some parts of response planning will be very easy for our system. For example, if a player queries general rules: “What are the rules for choosing starting positions?”, Hexagon should just give the player as many relevant rules as possible. These rules will generally come from the rule text, so the system could just read out the rules rather than framing its own sentences. The more complex cases include, specific rule questions and state-of-the-game questions.

For specific rule questions, the response frames should be fairly simple:

Q: How many points do you get for <action>?
A: A player receives $numberor”no” victory points for <action>

Q: Can you <action>?

A: $yesnoanswer <action> is $allowedornotallowed

In both of these examples, the frames work for both positive and negative answers, which is largely true of rule-based questions.

State-of-the-game responses are the most complex that Hexagon would have to handle. These would generally be phrased "$current state of interest, $needsforgoal".

Q: How close is <playername> to winning?

A: <playername> has $number victory points. To win, they need $number victory points."

Q: What does <playername> need to get <achievementcard>?

A: $playername has $number $typeofachievementrequirement. To get <achievementcard>, they need $number $typeofachievementrequirement

Generally response planning should be easier than most systems, but relies very heavily on the system's knowledge representation of the rules of the game.

Information Retrieval and Command Implementation

My personal preference for this information retrieval would be a combination of an external but customized database and an internal knowledge graph. Hexagon's internal knowledge graph would be for tracking gameplay and retrieving the current state of the game. Here we will store Hexagon's current knowledge about what players have what achievements, and our constant for the game version:

[sessionid] -> [version]

[playername] -> [number roads] [number settlements] [number cities] [achievement cards held]

The external database would be for the retrieving knowledge of the rules and objectives of the game. Currently, there are websites that host the text of the rules, but in order to be optimized for Hexagon's use case, they would need to be pre-processed to translate text into actions, goals, and requirements. This could be done more effectively and more appropriately for Hexagon by building our own database directly from the rule text. Considering the rules of Settlers of Catan (including all expansion packs) is not a lot of information relative to most databases/servers, this is a reasonable design. With this database built, we would simply need to send requests to our server to query the database, and process the responses with our response frames. The database will store relationships:

[achievement] -> [achievement requirements]

[action] -> [results] [restrictions]

[version] -> [win requirements] [allowed actions] [possible achievements]

This allows us to fill in the variables in the previous section. With these pieces in place, retrieving information for Hexagon will not be more complex than existing systems.

Text-to-Speech

Settlers of Catan uses fairly generic language. Major words in the game such as “ore”, “wheat”, and “robber” are not common English words, but should be easily interpreted by most existing Text-to-Speech systems. CMUDict contains all the major terms in the game, and there are no proper nouns (names or products) that would need to be included as part of gameplay. This implies that translating the text of situation-specific responses to speech should not require more than is available of existing systems. I verified this by giving my Mac TTS system a few sentences and vocabulary words from the game, and it had no issues.

Date/time data is not part of Catan gameplay and therefore does not need to be handled by Hexagon. Similarly, the numbers involved in the game are generic counting numbers: “3 wheat”, “12 victory points”, “rolled a 7”. This means that TTS handling on numbers can actually be very specific to this use, and would really never have to take years, pins, addresses, etc. into account.

My friend Georgia graciously volunteered to be the voice-model for Hexagon. She has an “estuary” English accent, which seems appropriate for a game about settling lands. Though all the common Catan words are common Emglish words, they are used slightly differently in a grammatical sense (“Seven wheat”), so we would pre-record many of these words to be specific to the game usage. There are a few common phrases to pre-record as well:

- “______ has ___ victory points and needs ___ victory points to win.”
- “______ is worth __ victory points.”

It would also be useful to pre-record the official rules, so that they can be read naturally when queried directly.

Transmission to Device and Playback

Playback for Hexagon should be fairly standard. Volume should be constant (and set-able by players) but we do not need to take into account time of day or modifying volume based on the situation as the use case is always the same. Similarly, if connectivity is poor and the request fails, Hexagon should simply say so, so the players do not expect the system to continue to track the game while it does not have access to the server.

Ongoing interaction

There are multiple interaction types for Hexagon. When directly queried with a “Hexagon, $question”, the assistant can respond to the question, and assume the interaction over. If the question is unclear, Hexagon may ask for and expect clarification. Hexagon is listening to track the gameplay constantly, and could need clarification as the game progresses. In this case, Hexagon would hear something that ambiguously matches one of its important patterns or keywords, and would prompt the players to confirm an action:

Player X: “.... and that’s ‘Longest Road’.”

Hexagon: “Please confirm the owner of ‘Longest Road’”
Player y: “Player X now has longest road”
Hexagon: “Okay”

Once an action or state is clarified and confirmed, Hexagon can consider the interaction over. If a clarification is not supplied within 10 seconds, Hexagon will reprompt but only once.