Home Butler
Creating a Virtual Home Assistant

Alexiei Dingli and Stefan Lia

Abstract Virtual butlers, or virtual companions, try to imitate the behaviour of human beings in a believable way. They interact with the user through speech, understand spoken requests, are able to converse with the user, and show some form of emotion and personality. Virtual companions are also able to remember past conversations, and build some sensible knowledge about the user. One major problem with virtual companions is the need to manually create dialogues. We shall introduce a system which automatically creates dialogues using television series scripts.

Key words: Virtual Butlers, Virtual Companions, Automatically Created Dialogues

Alexiei Dingli
Department of Intelligent Computer Systems, University of Malta, e-mail: `alexiei.dingli@um.edu.mt`

Stefan Lia
Department of Intelligent Computer Systems, University of Malta, e-mail: `slia0002@um.edu.mt`

1 Introduction

A home butler, also known as a virtual butler or virtual companion, is developed in order to help users perform a specific task or set of tasks. The home butler is able to help the user in various ways, such as suggesting possible actions or searching for a possible answer. Various uses for virtual companions have been found, ranging from helping students to acting as social companions to elderly people and as conversational partners. Given all these examples, a problem commonly found in virtual companions, especially conversational ones, is the creation of the possible dialogue the companion can take.

More often than not, the dialogue must be created manually. If the possible dialogue is limited, for example in direction asking in the Virtual Guide [1], this might be possible. However, when the dialogue can take any possible form, this is much harder to achieve. AIML can help achieve the required results, but these must be created and structured manually by human beings. This becomes a tedious process, with the probability that not all possible conversation states will be covered.

In this document, we shall present a virtual companion whose dialogue is automatically created using television series scripts. We shall explain the use of these television series scripts and the process of how these are transformed from scripts to possible dialogues.

2 Related Work

Many examples of virtual butlers exist. Of note is the COMPANIONS project, whose aim is to not only create task-based companions, but also socially interactive companions. The English Companion [2] created within this project is able to handle conversations with users, whilst trying to match the user's emotional state. The SERA project also tries to create an interactive companion, but with a different approach. The Virtual Receptionist, the Virtual Tutor, and the Virtual Guide [1] are three examples of companions that emerged from this project. All three use different methods to interact with the user, ranging from asking questions, non-verbal behaviour, as well as politeness levels.

Having achieved good results, the problem with these companions is still that they cannot achieve free flowing conversations with a user. The conversation between the companion and the user is limited, making the experience to the user less believable.

3 Proposed Solution

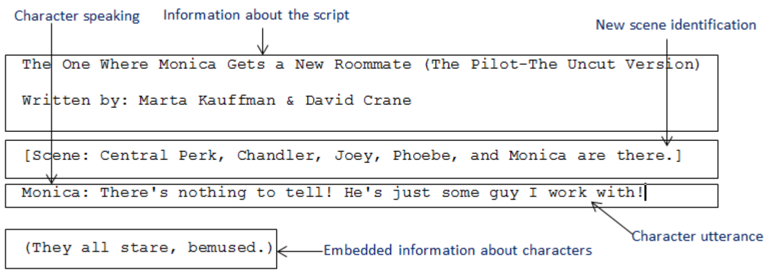
We propose a solution where the possible dialogues are created automatically using television series scripts. We shall explain the use of television series scripts as the source for dialogues, the process of creating the dialogues from the scripts, how the dialogues are stored, and how the companion will then search the store for a possible response to a user input.

3.1 Television series scripts as a source

Many forms of conversation sources are available. One of them is television series scripts. In television series scripts, a number of characters are always speaking to one another, creating a natural dialogue between two or more people. What we shall aim to do is to extract the necessary information from these scripts and automatically create the dialogue for the system to use. This dialogue will then be used to find an appropriate answer to the user's input.

Furthermore, these dialogues are easily accessible as text. The text is normally structured by first having the author and other information, which is followed by the actual script. The script, apart from including the actual conversation, also includes information such as the person currently speaking, information to what the character is doing and camera movements.

Television series scripts are split into multiple scenes. In each scene, there can be more than a single conversation taking place. Furthermore, a single conversation can be split over multiple scenes. As humans, we can easily identify these conversations automatically, however, this is much harder for a computerised system. The system should be able to identify these conversations, both spanning over multiple scenes and those within the same scene, and differentiate between them. This must be done in order to be able to better map the scenes into networks, and ultimately receive a better response from the system. The processes of how these are achieved will be explained in detail in the coming sections.


Character speaking
Information about the script
New scene identification
Character utterance
Embedded information about characters

```
The One Where Monica Gets a New Roommate (The Pilot-The Uncut Version)

Written by: Marta Kauffman & David Crane

[Scene: Central Perk, Chandler, Joey, Phoebe, and Monica are there.]

Monica: There's nothing to tell! He's just some guy I work with!

(They all stare, bemused.)
```

Fig. 1 Example of a script from the Friends television series which shows how the script is structured.

Apart from television series scripts, other sources for dialogue are present, such as film scripts, telephone conversations, and chat history. Film script are very close in structure to television series scripts, however, due to the nature that television series span over many seasons, television series are the better option since they map out the lives of the characters, and the story is continuous. In films, one does not find this. To create a sufficiently good enough dialogue system with film scripts, one would need to use a large number of scripts, which would introduce many different stories and characters, which could be a potential problem if not handled correctly. On the other hand, telephone conversation can be a very good source to automatically create dialogue systems. The problem with telephone conversations is how to acquire the conversations. With the limitations of speech recognition, there will be many errors in the speech to text translation. Therefore, this source could become unreliable. Using chat history was another possibility, however one has to consider the implications of using personal data from users.

3.2 Creating the dialogues

The dialogues will be represented in the form of networks, where each node represents an utterance of the script. Arcs going out of a node represent possible responses that the system can choose to return for that node. This structure will then be indexed, in order to allow fast searching and retrieving of possible answers.

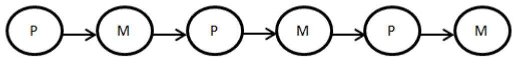
The extraction of information from the scripts is the first step in automatically creating the dialogues. Scripts contain a structure which allows the easy identification of data, such as scene boundaries, characters, utterances, and meta-data. Each script is first split into scenes. Each scene must then undergo a process which strips off any meta-data as well as extracting all the utterances from the script, and the character which was speaking. This data will be used later to create the networks. Each scene is then added to an inverted index [3] for the particular script.

An inverted index is built for each script in order to identify related scenes. If any are related scenes are found, these are appended to each other. This is done in order to preserve the conversation occurring within the two different scenes. Using the inverted index, we also find the topic of each scene, which also helps in identifying related scenes. The topic of the scene is found by extracting all nouns and verbs by making use of the GATE framework [4], since these give the most meaning, and finding the highest rated noun or verb from the inverted index. Once the inverted index is built, and the topic is found, we can search for related scenes. This is done by calculating the cosine similarity between two scenes. If two scenes have the same topic, their score is increased, since it is more likely that the scene has continued. Therefore, the final score is the cosine similarity, with the possible addition if their topics are the same. If the final score is greater than some threshold, these scenes are said to be related, and therefore merged together.

The next step is to pass the script for network creation. It is important to remember that each scene will most likely include multiple conversations between the characters involved. The process employed in this system tries to identify these conversations and model the networks accordingly. The reasoning used in order to find different conversations within a scene is as follows. Let us say that a particular scene is made up of 5 different characters. The simplest form a conversation can take is when two people are talking together, with not interruptions from anyone else. This is illustrated in figure 2(a). On the other hand, if the first character speaks, and this is followed by two other speakers, followed by the first speaker, we can assume that the conversation assumed the following pattern: the first person spoke, the other two persons replied to that person, and the first person spoke again. This is illustrated in figure 2(b). However, if we take the same scene, but change the order which persons speaks, we might get different results. If the first person speaks, but does not speak for a number of turns, then it can be assumed that a second conversation was taking place. This is shown in 2(c) We must therefore be able to model these situations in order to have the best model and ultimately find the best reply to the user.

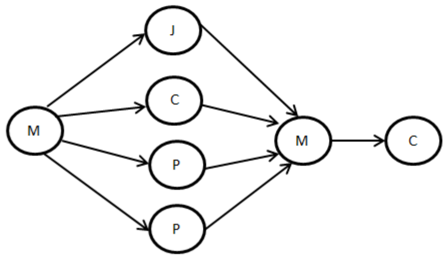
Once the networks are created, we must then index these networks for fast retrieval. Each node holds an utterance, and all possible responses, represented as arcs to other nodes. In the index, for each word in an utterance, we store which unique node ID and unique network ID that word occurred in. This

Paul: Ever since she walked out on me, I, uh...
Monica: What?..... What, you wanna spell it out with noodles?
Paul: No, it's, it's more of a fifth date kinda revelation.
Monica: Oh, so there is gonna be a fifth date?
Paul: Isn't there?
Monica: Yeah... yeah, I think there is. -What were you gonna say?



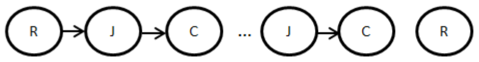
(a) Simple network

Monica: There's nothing to tell! He's just some guy I work with!
Joey: C'mon, you're going out with the guy! There's gotta be something wrong with him!
Chandler: All right Joey, be nice. So does he have a hump? A hump and a hairpiece?
Phoebe: Wait, does he eat chalk?
Phoebe: Just, 'cause, I don't want her to go through what I went through with Carl- oh!
Monica: Okay, everybody relax. This is not even a date. It's just two people going out to dinner.
Chandler: Sounds like a date to me.



(b) Multiple responses to a single utterance

Ross: (squatting and reading the instructions) I'm supposed to attach a brackety thing to the side things, using a bunch of these little worm guys. I have no brackety thing, I see no whim guys whatsoever and- I cannot feel my legs.
Joey: I'm thinking we've got a bookcase here.
Chandler: It's a beautiful thing.
Joey: (picking up a leftover part) What's this?
Chandler: I would have to say that is an 'L'-shaped bracket.
Joey: Which goes where?
Chandler: I have no idea.
Joey: Done with the bookcase!
Chandler: All finished!
Ross: (clutching a beer can and sniffing) This was Carol's favorite beer. She always drank it out of the can, I should have known.



(c) New conversation within the same scene

Fig. 2 Three possible structures of a network, given sample scripts.

allows us to find the related nodes to any user input very efficiently. When indexing an utterance, we remove any stop words and stem [5] all words before the actual indexing. It was noted that if this step is not done, results might not be as accurate. This can be explained through the fact that two words can be derived from a single word, but in different forms. If no stemming is done, these two words will not be matched, therefore accuracy would be less. It was also noted that without removing stop words, the index would be very large, thus leading to slow responses from the system. The stemming of words also reduces the size of the index.

3.3 Finding a response to user input

To return a response to some user input, the network index is used extensively. The user input is first split into single words, and all nodes containing at least one of these words is retrieved from the index. From these possible nodes, we calculated the similarity between the utterance stored in the network by making use of the Jaccard Index [6], and the user input. Further processing will occur on the node with the highest similarity. If this node contains a single outward arc pointing to some other node, then the utterance stored in that node will be returned as an answer. If the node has more than one outward node, the returned response will be that which achieves the highest sentence similarity.

As a precaution, the system is also connected to an existing AliceBot. The AliceBot is only used when the response found from the indexed networks is not good enough. It also ensures that the user always receives a response back from the system.

4 Evaluation

The system was exposed via a website and people were asked to evaluate the system based on the following criteria: response accuracy to what the user said, natural word usage in responses, conversation handling, topic handling and response to topic changes, and whether the system responded in a timely fashion. The study showed that the weakest part of the system is conversation handling. The reason for the system achieving low score for conversation handling is the dialogue manager. The dialogue manager implemented within this companion is a primitive one, which does not, for example, collect information about the user and their personality, or does not connect to an online question/answering system to answer general knowledge questions.

However, when the user steered away from general knowledge questions, and engaged the companion in a normal conversation, the responses were better suited, more appropriate, more natural, and more enjoyable by the user.

5 Conclusions and Future Work

We have shown with this system how to automatically create dialogues based on television series scripts and how to store these scripts for later use. We have also explained the process of how query this storage of dialogues and of how to choose the best response to some user input.

5.1 Future Work

The system developed has a lot of room for improvement. As mentioned in previous sections, the system would greatly benefit from the introduction of user profiling. At the moment, the system does not store any information about the user. Such information can be easily accessible from the conversation itself. The system should ideally be able to extract information related to the user from the user's input and store it for later use. Such information can make the conversation seem more natural and be on a more personal level to the user.

Another aspect of the system that can be improved is question/answering. Having the system connected to an external question/answering systems, such as Ask [1]. The system would have the capabilities of sending requests to such systems, parsing the results returned, and present the information back to the user. Such capabilities would be used alongside the current dialogue management to provide a broader knowledge base from which the system can find answers.

We would also like to include speech recognition, speech synthesis, and an avatar. These three elements were to be included in order to achieve a more

[1] www.ask.com

natural feel to the system. These components allow the user to interact with the system by using natural speech, and also be able to look at the avatar whilst talking and listening. Speech synthesis has been included within the application, but due to the lack of the other elements, it does not make much sense to have it implemented. Once all three elements would be in place, we believe that the system has the potential to achieve better results.

References

1. D. Heylen, M. Theune, R. Akker, and A. Nijholt. Social agents: the first generation, 2009.
2. M. Cavazza and et al. How was your day? an affective companion eca prototype, 2010.
3. Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *COMPUTER NETWORKS AND ISDN SYSTEMS*, pages 107–117. Elsevier Science Publishers B. V., 1998.
4. Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. Gate: an architecture for development of robust hlt applications. pages 168–175, 2002.
5. M. F. Porter. *An algorithm for suffix stripping*, pages 313–316. 1997.
6. Palakorn Achananuparp, Xiaohua Hu, and Xiajiong Shen. The evaluation of sentence similarity measures. In *Proceedings of the 10th international conference on Data Warehousing and Knowledge Discovery*, DaWaK '08, pages 305–316, Berlin, Heidelberg, 2008. Springer-Verlag.