CS3105 - The Imitation Game

140015554

10th of March 2017

1. Introduction

The practical specification requires to build a chatbot that would use artificial intelligence to conduct text based conversation with a human. The program should accept a block of input in text and respond with a block of text via a command-line interface at a terminal in Linux. Furthermore, an analysis has to be made of the techniques used in the development process and results achieved during the example conversations with the bot. Finally, a reflection has to be made on how effective the Turing test is as a measurement of machine intelligence and its implications to the design of general intelligence.

2. Design and Implementation

The solution was implemented using Python programming language. Python was chosen for the easiness of use when building relatively small applications and due to extensive amount of libraries and tutorials available regarding parsing and artificial intelligence in general.

The initial solution implemented an infrastructure for the inner workings of the chatbot. It provided a way to tokenise the input into separate sentences by recognising the end of a sentence word, which is identified if the word contains one of these characters: *',' '.' '!' '?' ';' '\n'*. Furthermore, for each sentence a unique response was generated using a simple keyword matching algorithm, which would parse the sentence into tokens and look though the hard-coded dictionary to see if any of the keywords are identical to the tokens. In case a matching keyword is found, a response would be chosen randomly from a predefined set of answers. This approach was very limited in respect to how many reasonable answers it could generate, but was useful when only a certain keyword in a sentence mattered. For example, identifying a greetings or farewell phrase in a sentence and responding accordingly.

Later on, various techniques were implemented and analysed in order to make the chatbot more dynamic in terms of what it can respond to and to improve the quality of the responses themselves.

2.1. Text Generation using Markov Chains

To generate responses for the inputs provided by the user, a text generation technique called Markov Chain model was chosen. The basic principle behind Markov Chains is the linking of word n-grams in a sentence. Each n number of words is linked to the next corresponding word in a sequence, constructing a graph of words that an arbitrary word sequence can be made of. The links are then used to generate new sentences by starting in an arbitrary point in the graph and following the links from that point until the end of sentence is reached.

The implementation uses a corpus of text data to build the Markov Chain database. It analyses the corpus text word by word and builds the dictionary of Markov Chains for that particular data set. Each sentence is split into tokens of words and two consecutive tokens are used as a key to the dictionary entry, while the next token in a sequence is the value of the corresponding entry. If the dictionary already contains the current token pair, it will append the next token in the sequence to the corresponding list of values. Moreover, if the token pair and the next token in the sequence already exists in the dictionary, the solution will increment a frequency counter indicating how often a particular word appears after the current token pair. Finally, the dictionary is stored in a binary file and is loaded during the start of the program.

To generate a response the solution selects two words from the users input at random and uses that pair of words as a seed to start the generation sequence. The next word in the sequence is selected by indexing the dictionary with the chosen pair and when choosing the most frequent word that usually fallows that pair. The cycle comes to an end when a word with an end of sentence delimiter is reached or after a certain sentence length is achieved (this prevent the chain from looping infinitely). Moreover, if the pair of words chosen is not present in the dictionary, a random seed is generated and the sentence is made using it.

Here is a diagram showing how a sentence is constructed using the Markov chains:

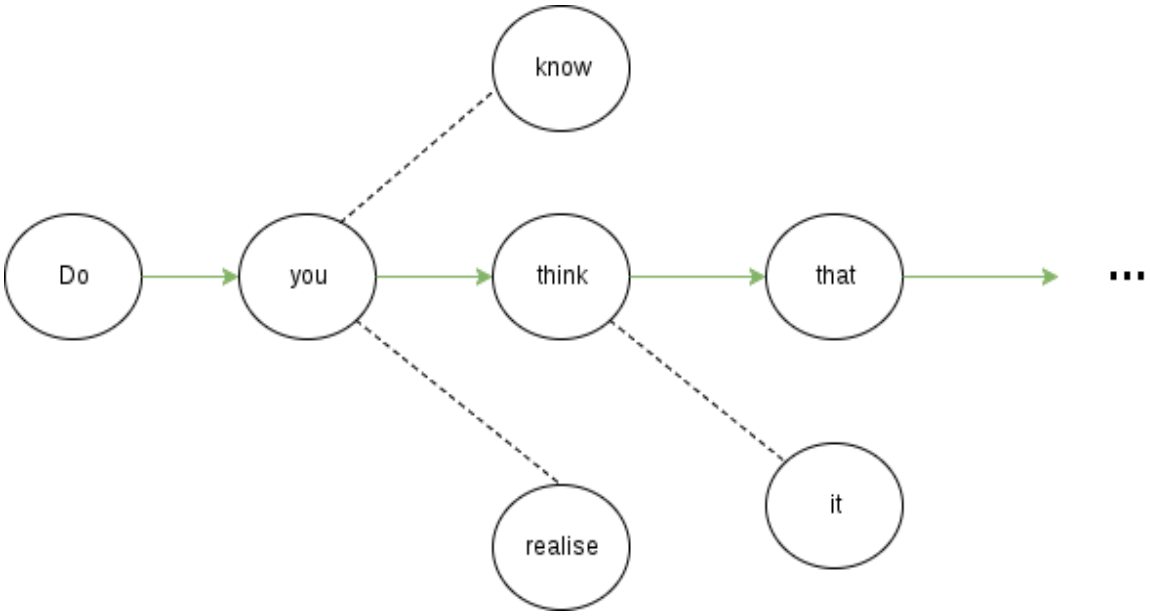

Figure 1. Markov Chain

"Do you" is the seed, while the green arrows indicate which words are chosen next in the chain. The dashed lines connect alternative paths that could have been chosen if they had a higher frequency of occurrence in the text corpus.

Advantages

- Markov Chaining allows to generate response for any input without the need to know the context or the semantic structure of the sentence.
- The grammatical structure of the generated sentence is usually correct, as only the words that are consecutive in a sentence are chosen.

- It replicates the most common patterns of words in a sentences and uses input keywords to generate a response, thus, in theory, making the response relevant and coherent to the conversation.

Disadvantages

- To capture the structure of sentences properly a huge corpus of text data is required.
- Even with relatively good text corpus the responses generated by the Markov chains are only sometimes relevant and acceptable, while usually it produces a completely nonsensical response, which might have a proper grammatical structure, but does not have any meaning in terms of the context of the conversation.

Example Dialogue

(For the example dialogue, a corpus of movie dialogues [1] was chosen as data to train Markov Model on)

User: *How is your day going?*

AI: *Your day job.*

User: *I am sorry?*

AI: *I am not.*

User: *I don't understand what are you trying to say.*

AI: *You trying to get a little more than a hundred. seriously. this guy is the.*

User: *What is the meaning to life, the universe and everything?*

AI: *Meaning of the world. you know what i mean i don't. The.*

Markov chains seems to be able to generate responses given a seed of two words, yet the responses lack meaning and usually can be discarded as complete nonsense. On the other hand, it might still be useful in case all the other text generation methods fail and a response is still needed to continue the conversation. Furthermore, a bigger and better corpus of data might improve the performance, as the patterns of the sentence structure become more widely established in the Markov chains.

2.2. Response Retrieval Based on Word Order

In order to improve the quality of the response, another technique was explored and implemented. It uses a retrieval based approach where a response is being generated by comparing the user's input sentence with a list of sentences stored in a database. If the user sentence resembles any on the sentences in the database, a reply will be retrieved from the database associated with that particular database entry.

The database is built using a corpus of dialogue text by mapping a sentence from one person to the reply of another. In the end, the database contains various replies that can be retrieved given an input similar to the one that was used to "trigger" the response in the original dialogue. For example, if one person said *"How are you?"* and the other one replied *"I am fine. Thanks"*, whenever someone asks a question similar to *"How are you?"*? it will retrieve the appropriate response *"I am fine. Thanks"* from the database.

To measure how close one sentence resembles another a method described in research paper [2] is used to calculate the similarity ratio between two sentences. This metric allows to determine how many words are shared between the two sentences or, in other words, how different the sentences are in terms of the words they contain. Also, it takes into account the order of the words, thus allowing to know if any of the words are swapped or are in different positions, hence telling us how distant apart two sentences are. Moreover, the proposed metric can also be improved by determining if the words that are different, when comparing the user's input and database entry, can be replaced by synonyms which would have a similar meaning and wouldn't change the context of the sentence too much. However, the implementation only uses sentence structure and word order as a measure of similarity and doesn't take semantics into account.

Here is the procedure for calculating the similarity ratio between two sentences:

1. The sentences are split into tokens and vectors are constructed (Figure 2) indicating the position of words that are present in a particular sentence when compared to the source sentence. If a word is missing, 0 is being used as value of the position for that particular word.

$$\mathbf{r}_1 = \{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\}$$
$$\mathbf{r}_2 = \{1\ 2\ 3\ 9\ 5\ 6\ 7\ 8\ 4\}.$$

Figure 2. Word order vectors
Picture taken from research paper [2]

2. Then, the similarity ratio is calculated by normalising the difference of the word order (Figure 3). This ratio accounts for the difference and position of words in a sentence.

$$S_r = 1 - \frac{\| \mathbf{r}_1 - \mathbf{r}_2 \|}{\| \mathbf{r}_1 + \mathbf{r}_2 \|}.$$

Figure 3. Similarity Ratio
Picture taken from research paper [2]

The solution uses a database of movie dialogues [1] and compares each sentence given by the user to the list of entries in a database by calculating the similarity ratio. In case, the ratio is lower than a certain threshold, the response associated with the database entry is returned, otherwise empty string is retrieved.

Advantages

- Produces a response that has correct grammar, as the sentence is usually taken from human inspected text corpus.

- The response usually makes sense when associated with the input given, as the database contains similar questions or statements and an appropriate response can be retrieved.
- The quality of the responses can be improved if more dialogues are added by a human to the database or by some kind of process, that can learn from the AI to Human conversations, is implemented.

Disadvantages

- Fails to keep track of the context of the conversation and can only respond to a particular question or statement.
- Even with a nicely structured text corpus, the retrieved responses might not make any sense or be completely out of context.
- The amount of answers that can be retrieved is directly proportional to the size of the database and, therefore, is quite limited.
- Hard to scale the program, if the database becomes to big. Requires a complete traversal every time a query is made, hence, takes a lot of time to produce a response.

Example Dialogue

(For the example, a corpus of movie lines [1] and a handwritten list of various dialogues was chosen as data for the retrieval database.)

Dialogue 1:

User: *Hello, how are you?*

AI: *Greetings! Excellent. How are you? :)*

User: *I am good.*

AI: *How is your day going?*

User: *It's going well.*

AI: *Does it still hurt?*

User: *What do you mean?*

AI: *Why didn't George tell me?*

Dialogue 2:

User: *hey*

AI: *Hi there!*

User: *how are you?*

AI: *Not too bad. What about you?*

User: *I am good. Thanks*

AI: *Good to hear. Is the turkey organic?*

User: *I don't know*

AI: *Thanks, man. Appreciate it.*

The retrieval based approach seems to provide better responses than Markov Chain model. However, it still lacks the ability to keep track of the conversation context and respond meaningfully at all times. Moreover, the number of available responses is limited by the size of the database and thus it can only respond to certain type of questions and statements. Similarly to the Markov Chain model, if the quality and the size of the corpus is increased, the solution should respond to bigger variety of situations and provide more relevant responses.

2.3. ELIZA's Style Pattern Matching

To improve the responses provided by the retrieval method and be able to dynamically increase the response database, techniques used in a chatbot "ELIZA" [3] were analysed and implemented. ELIZA communicates in such a way that all the questions or statements made by the user are reflected back to the user. Ideally, the user always provides the topic of conversation and the chatbot just needs to curate the conversation by asking multiple questions and reflecting on what was already said.

ELIZA tries to match pattern match the sentence given with a list of rules. In case a match occurs, it will choose one of the responses associated with that rule and return it to the user as a response. Also, fills the response with keywords from the user input and changes the subject of the sentence (e.g., "you" to "me", "I" to "you") to properly reflect on what was said.

Advantages

- Simple rules can make the chatbot look sophisticated enough and even trick people into believing that they are communicating with an actual human [3].
- Easy to extend the rule base by introducing new patterns to match.

Disadvantages

- It never introduces new things to the conversation, thus, the conversation is very one-sided.
- Hard to match all possible input structures. In order to work even reasonably well, requires a huge list of rules.

The implementation described in the original research paper [3] pretends to be rogerian psychotherapist and is focused towards rephrasing of what was already said. While, in this chatbot, pattern matching techniques are used to try and learn from the users input. This is achieved by asking the user to tell how would he respond to such question in an indirect way and then storing the original question and the user's answer into the retrieval database for future use.

Here is an example dialogue what shows how the program will learn from the input if the user provides a statement or a question that matches one of the patterns:

User: *Do you know why the sky is blue?*

AI: *No, I've never heard of why the sky is blue. Could you tell me why?*

User: *Sky is blue because molecules in the air scatter blue light from the sun more than they scatter red light.*

AI: *Thanks. This is helpful.*

The user response is stored in the retrieval database and will be found the next time user asks ”Do you know why the sky is blue?”. This way of learning is extremely slow for a chatbot and usually relies heavily on the answers that the user provides. The bot has no means to refine the learned knowledge and improve its understanding, thus, it can lead to a database that contains nonsensical responses to the given questions. On the other hand, it allows to increase the database dynamically and thus provide more relevant responses to the user queries.

2.4. N-gram Based Text Categorisation

The last technique that was explored and implemented provides a way to categorise text using a n-gram based approach. The method, described by W.Cavnar and J. Trenkle [4], generates a text profile that can be compared with a list of profiles in a database to identify the category of the input. Profiles are made by splitting each word into n-grams of characters and when making a list of the most frequent n-grams for that particular text input. When two profile are compared, the most frequent n-gram positions in the list are checked and the difference score is calculated by determining how out-of-place the positions are. According to the Zipf’s law [4], this approach should be able to identify different text categories by looking at unique n-gram distributions of a certain topic.

Here is an example taken from the research paper [4] stating how the comparison is done between two different profiles:

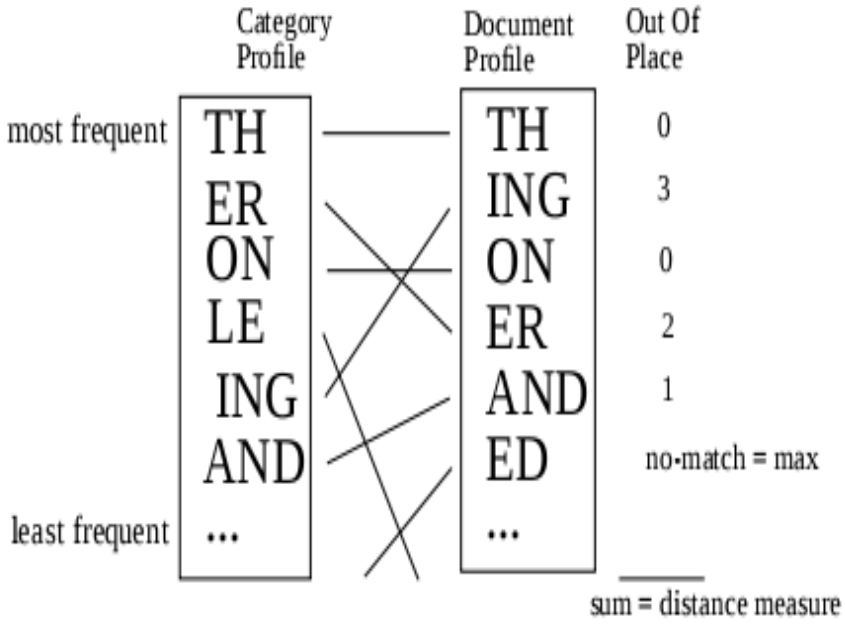

Figure 4. Distance measure of two profiles

The implementation uses a corpus of data containing bunch of articles on different topics, for example: art, archaeology, computer, politics, etc., and uses an ordered list of 1-5 grams for each word to generate the unique profile of a particular category. Whenever the user provides an input, a profile is generate for that imputed text and the distance score is calculated by comparing the user’s text profile with the ones in the database. Finally,

the profile that has the lowest distance score implies the category the user's input should be in.

Here is a list of examples showing which category the text was identified to belong to:

Input: *Usually, I drive home by car. I really like my new Tesla 3.*

Category: *archaeology*

Input: *Painting is the practice of applying paint, pigment, color or other medium to a solid surface.*

Category: *space*

Input: *Meanwhile, cook the pasta according to pack instructions. Add the mushrooms to the frying pan, stir and cover for 8 mins, or until the mushrooms are soft.*

Category: *hardware*

The results seem to indicate that the method completely fails to identified the category of the topic. However, this could be explained due to the limited input size and the lack of corpus data that could be used to generate more accurate n-gram distribution profiles. Ideally, text categorisation techniques could be used to determine the category of the input and increase the accuracy and scalability of the retrieval based approach by only searching through the inputs and responses stored in a particular category.

2.5. Final solution

In the end, all techniques discussed above were combined together into a single chatbot program. The final solution tries to respond to the user's input as relevantly as it can. It will first check if the input contains one of the greetings or farewell keywords and respond accordingly. If that fails, it will try to retrieve the input from the database. In case of another failure, the ELIZA module will be used to try and learn from the user that the response should be. Finally, if none of the above works, the Markov Chains are used to generate some kind of response in order to continue the conversation.

Here is a conversation that implemented program had with an award winning chatbot called Mitsuku:

Mitsuku: *Hi there. How are you?*

Bot: *Hi! Not too bad. What about you?*

Mitsuku: *Glad to hear it. What's new?*

Mitsuku: *Nothing much. Just chilling on here.*

Bot: *You. Will you tell her I came by?*

Mitsuku: *Yes I plan to use your remarks as gossip in later conversations.*

Bot: *With three left breasts, behemoth hips and one testicle ! now, there's no way i'm gonna' take it as a guy, great, but if not... don't even know where i'm.*

Mitsuku: *You and your three left breasts behemoth hips and one testiclenow there is no way you are going to take it as a guy great but if not.*

Bot: *Going to kick in, what it's gonna' come back. drink this! what're you talkin' about chromosomes. most people have forty six two x if you're a girl,.*

The chatbot seems to be able to hold a conversation for a while. However, after some time it loses the the track of context and responds with almost meaningless replies. Furthermore, the award wining robot seems to also have difficulty with keeping track of the context and starts breaking when the reply is a bit too unusual. Even with a combination of sophisticated natural language processing techniques and text generation methods the implemented chat bot seems to be barely able to have a conversation with a human. This implies the underlying complexity of making an artificial intelligence that could understand and respond as the humans would.

3. Turing Test

The Turing test was developed by Alan Turing in 1950 in order to provide a way to distinguish a machine from a human or at least introduce the idea of having a situation were people are tricked into thinking that an intelligent behaviour of an entity was produced by a human being and not by a machine.

The test is done by a human using a chat program. He communicates with another intelligent entity and has to decide, from the interaction, if the other entity in the chat is a human or a computer program made by humans. The test is quite open-ended and leave the judgement to the participant, making it hard to quantify the measurement of the intelligence. Also, it allows the participant to be exploited by targeting certain flaws in the human cognitive abilities and trick them into thinking that a machine is human, when in reality it is not.

Even though the test is very subjective, it gives an insight into the underlying complexities that are present to those who try to replicate human intelligence. Some intelligent behaviour is inhuman and sometimes humans don't act intelligently. Therefore, it is hard to quantity intelligence into a set of rules that could be easily replicated by a machine.

The techniques used in the implementation tries to achieve some part of the intelligence exhibited by humans. However, it struggles to do that. To achieve really good performance many of the methods require significantly huge data sets and computing power and even then the models usually fail to generalise and lack robustness and flexibility. It does seem to imply that holy grail of the AI - the general intelligence, is still far from being made. Furthermore, it reflects on how good our brains are at manipulating information. On the other hand, a lot of promising techniques are currently being researched in the field of AI, such as deep learning and neural nets, which try to replicate the behaviour of neurons in the brain and create programs that could train as humans would. This will hopefully lead to programs that can learn more dynamically and produce more generalisable behaviours.

4. Conclusions

The solution fulfilled all the requirement detailed in the practical specification. It provided a working chatbot that can take a text input and respond with an answer. Furthermore, it explained and analysed different natural language processing and generation techniques

that are used in the field of artificial intelligence. Finally, it discussed the underlying complexities of chat bots and how those problems correlate to the development of general intelligence.

References

[1] Natural Language and Dialogue Systems. Film corpus 1.0. https://nlds.soe.ucsc.edu/fc.

[2] Zuhair A. Bandar James D. O'Shea Yuhua Li, David McLean and Keeley Crockett. Sentence similarity based on semantic nets and corpus statistics. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 18(8), August 2006.

[3] Joseph Weizenbaum. Eliza — a computer program for the study of natural language communication between man and machine. https://web.stanford.edu/class/linguist238/p36-weizenabaum.pdf.

[4] William B. Cavnar and John M. Trenkle. N-gram-based text categorization. http://odur.let.rug.nl/~vannoord/TextCat/textcat.pdf.