Conversational Agent that Models a Historical Personality

Adrian Bogatu, Dorin Rotarescu, Traian Rebedea, Stefan Ruseti

University Politehnica of Bucharest
313 Splaiul Independentei,
Bucharest, Romania

Autonomous Systems
22 Tudor Vladimirescu,
Bucharest, Romania

bogatu.adrian@gmail.com, dorinrotarescu@yahoo.com, traian.rebedea@cs.pub.ro, stefan.ruseti@cs.pub.ro

ABSTRACT

In this paper we discuss the current approaches in question answering (QA) and their applicability in building a conversational agent that models a historical figure that gives informative and relevant answers to user questions about the life of that personality. We analyze two main methods: one in which we use an ontology to build our knowledge base and one where we don't have a knowledge base and we solve the *answer sentence selection* problem for question answering. We observed that the first method is better for answering more general questions and the second method can deal with more specific and complex lexically and syntactically questions. The implementation of the conversational agent relies on the two combined approaches, the second being a fallback option if the first method is not able to provide an answer.

Author Keywords

Conversational agent; natural language processing; information retrieval; question answering; ontologies; answer sentence selection.

ACM Classification Keywords

H.3.4 Systems and Software: Question-answering (fact retrieval) systems.

INTRODUCTION

More than 60 years ago, Alan Turing raised the question "Can machines think?" in the book with the same name [10] where he devised the "Imitation Game" test, which is similar to the now known Turing test. Since then, computer scientists have tried to create programs that can interact with humans and maintain a human-like conversation.

Cleverbot[1] is one of the programs that accomplished that and also obtained a score of being 59.3% human from the jury of the competition, while humans achieved a 63.3% "humanness" score. Cleverbot uses a database of saved conversations updated constantly and replies with an answer by matching the user input to previous phrases found in the database.

Other conversational agents simulate the personality of a certain person or typology of person and usually use rule-based systems. An example is the bot ELIZA that tries to match the user input to a rule and output the answer associated with that rule [12]. Another example is Freudbot, a chat-bot similar to ELIZA that tries to impersonate the psychologist Sigmund Freud and talk about his theories and biography in the first person [6]. A different category is open domain question answering software systems like IBM's Watson, which uses Natural Language Processing (NLP), information retrieval, machine learning and other techniques to provide an answer to a question [2].

This paper presents the current approach towards the implementation of a conversational agent that models a historical character using basic NLP concepts, information retrieval methods and question answering techniques. The conversational agent can be used in museums to guide and inform visitors or in schools as an e-learning tool.

Building a conversational agent using NLP is not an easy task, firstly because we do not always speak in a grammatically correct fashion. Secondly, in a conversation, the speaker assumes that the listener knows and understands the details of the ongoing conversation. A big challenge is to determine the context in which a question is posed and to understand the meaning beyond the lexical structure of the question.

From a programming point of view, the state-of-the-art chat-bot relies on a set of files containing rules in the form of question-answer pairs, usually defined in AIML [11] or similar languages, which are constructed based on the way the answer is expressed.

However, in order to build a robust conversational agent it is expected that an input (question) to have multiple rules that can be matched against the question and provide a relevant answer. This implies that a large number of various rule-answer pairs are needed. On the other hand, it is hard to predict the interaction between rules when adding new rules to an existing set.

In order to model a historical figure, we started from that person's biography on Wikipedia and the associated DBpedia[2] page. Consequently, we can identify the way a certain property is expressed starting from the properties and their values from DBpedia correlated with the Wikipedia text.

The paper is organized as follows: the next section discusses some existing approaches on building a

[1] Cleverbot, online at http://www.cleverbot.com/.

[2] DBpedia, online at dbpedia.org.

conversational agent, using either an ontology or a text matching method in order to answer questions. The next section describes the steps we took to create a knowledge base and generate rules for our conversational agent. In the end of the implementation section the results using ontologies are presented. After that, the integration of the answer sentence selection approach, as a fallback method for the first approach, is analyzed. Finally, the conclusions are presented.

RELATED WORK

Our project is inspired from conversational agents implemented using ontologies as well as conversational agents that do not use a knowledge base and rely only on the question asked and a corpus from where the answer will be extracted. Next we will present some background on these two approaches.

Ontology approach

As defined in [5], an *ontology* is an explicit specification of a conceptualization, where *conceptualization* is defined as an abstract, simplified view of the world that we want to represent.

The Intelligent Verilog Compiler Project [9] is a tutoring system used for teaching the Verilog language. It is said to be intelligent in two ways: "it helps check the syntax and the semantics of the learner's program and it finds a technical or English definition, comparison or example suitable to the error being reported in the context of the piece of the code. It displays the information next to the incorrect code and errors in order to 'scaffold' learning without directly providing the answer" [9]. This method of interaction is accomplished using an ontology of the Verilog language.

Another dialog system where ontological resources are used is one personifying the author Hans Christian Andersen. The domains of discourse contain his fairy tales, his life and the user [8]. It is stated that the reasons to use ontologies are: faster development because of the shared ontology over different conversation domains; the fact that the application can be easily extended to support new domains of conversation.

A technical approach on using ontologies for question answering is described in [4] and given an implementation in [3]. The authors thought of *matching* as an operation on two graph-like structures that "produces a mapping between elements of the two graphs that correspond semantically to each other" [4]. Starting from the said concept of *matching*, the authors imagined the next step: semantic matching, which also analyzes the meaning behind nodes in the two graphs.

As described in [4], this approach has two main features:

- search for semantic correspondences by mapping meanings (concepts), and not labels, as in syntactic matching. As the rest of the paper makes clearer, when mapping concepts, it is not sufficient to consider the meanings of labels of the nodes, but also the positions that the nodes have in the graph.
- use semantic similarity relations between elements (concepts) instead of syntactic similarity relations. In particular, we consider relations, which relate the extensions of the concepts under consideration (for instance, more/less general relations).

In the case of ontologies, this approach works if we can construct an equivalent graph-like representation of a given ontology.

Answer sentence selection approach

Answer sentence selection is the task of finding a sentence from a set of candidate sentences that best answers a given question.

The method we rely on the most is the one described in [1]. In [1], there are three main approaches analyzed, but we are only interested in the first one: the approach that uses algorithms that rely on "sophisticated syntactic/semantic processing" [1]. The authors present three main types of extracting and using knowledge for the answer sentence selection problem. The first one is to determine the type of answer (also known as "Qtarget") to a given question. For example:

Question: What is the duration of the song "Hey, Jude"?
Qtarget: TEMPORAL-QUANTITY
or
Question: When were you (John Lennon) born?
Qtarget: DATE

After finding the type of answer needed, the set of possible answers can be filtered according to the semantic type of the question, and we can remove all the sentences that have a type that does not match the found Qtarget.

The second type of knowledge contains the semantic relations between constituents that appear in the question. The correct answer should preserve these relations. Example from [1]:

Question: Who killed Lee Harvey Oswald?
Text: Jack Ruby, who killed John F. Kennedy assassin Lee Harvey Oswald.

The explanation is given in [1] immediately after the example: "Even if 'John F. Kennedy' is textually closer to the question terms 'killed' and 'Lee Harvey Oswald', the system will choose "Jack Ruby" because its logical subject relation to the verb matches that of the interrogative in the question."

The third type of knowledge relies on the use of paraphrases. Because the wording in a potentially correct answer is not always similar with the one in the question, immediate textual matching does not always work. The idea is to generate alternate formulations of the question (but preserve the meaning) in order to increase the matching chances for a good answer.

An example of reformulation of a question from [1]:

Question: How deep is Crater Lake?
Reformulation patterns:

- Crater Lake is <what distance> deep?
- depth of Crater Lake is <what distance>?
- Crater Lake has a depth of <what distance>?
- <what distance> deep Crater Lake?
- Crater Lake's depth is <what distance>?

OUR METHOD

ChatScript

ChatScript [3] is a chat-bot engine, a tool that helps build conversational agents that are based on rules. It uses a scripting language to build these agents and process natural language. ChatScript represents the state-of-the-art for conversational agents and helped with the transition "from matching patterns of words to matching patterns of meaning." [13]

A chat-bot is modeled through a set of script files that contain rules. A rule is formed from a pattern and a response. The response represents the output that a ChatScript bot will provide if the input matches the pattern. For example:

```
u: ( Where * you * born ) In the capital.
u: ( When * born  ) This century.
```

The elements in the parentheses constitute the pattern and the sentence after the pattern represents the answer. The star symbol is a wildcard that can match none, one or more words.

In our case, the input is represented by the question asked by the user, therefore we want to create the best patterns for each possible answer we have retrieved from a person's biographical text. Example for our generated rules:

```
u: (vb marry) I married Elsa Löwenthal on 2 June 1919 , after having had a relationship with her since 1912 .
u: (vb die in in) I died in Princeton Hospital early the next morning at the age of 76 , having continued to work until near the end .
```

In the examples above, in the parenthesis we find the part of speech (in our examples, this is a verb) returned by Stanford NLP and the lemma of the word from the expression found by the algorithm.

Because our program needs to support a large number of different historical figures, we needed to automate the creation of ChatScript specific files. The method of generating the scripts is described in the *Pattern generation* subsection.

Stanford CoreNLP

Stanford CoreNLP [7] [4] is a tool for analyzing and processing text. It integrates some useful modules that we used for Wikipedia articles. The part-of-speech tagger was used to differentiate between verbs, nouns, adjectives etc. The tokenizer was used to split paragraphs into sentences and sentences into words, while the lemmatization tool was employed to find the canonical form for a word. In the end, the co-reference resolution system was used to link the subject of sentences to the anaphoric proper name in the context of a paragraph, if there exists such an anaphoric element.

Methods

Expressing a Property

In order to understand what is trying to be expressed in a sentence, we start from the DBpedia properties of a large set of people. Subsequently, for every property, we try to determine how that property is expressed in the Wikipedia corpus. For the information extraction, we use Stanford CoreNLP.

To find the manner in which a property is expressed, we searched the value of that property in the sentences from Wikipedia where the person which the article is about appears in. Having the desired sentence identified, we annotated it using Stanford CoreNLP and obtained a syntactic parse tree. Analyzing this tree, we determined that the root is the verb directly connected with the subject. Having the parse tree, we considered that the best way to express the property is the path from that property to the root verb.

Applying this algorithm to a large set of people, we managed to build a big, but extensible knowledge base by introducing the most relevant output expressions in a knowledge base.

Examples

In Table 1 we present some entries in our knowledge base, where the property is extracted from DBpedia and the lexicalization represents an enumeration of ways the given property appears to be expressed in the Wikipedia articles.

DBpedia property	Lexicalizations
birthDate	born; born in
almaMater	receive in; graduate as
award	award; receive
college	graduate from; attend
deathPlace	die in
profession	serve in; become
spouse	marry; marry to

Table 1: Examples of how specific DBpedia properties are most often expressed.

[3] ChatScript, online at http://chatscript.sourceforge.net/.

[4] CoreNLP, online at http://nlp.stanford.edu/software/corenlp.shtml

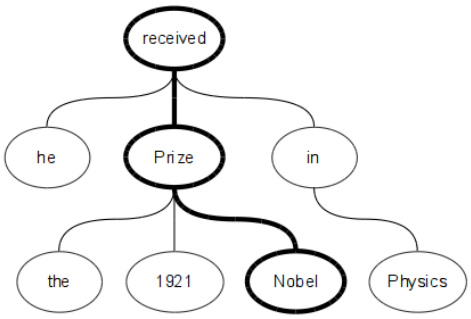

Figure 1: Parse tree for the phrase "*He received the 1921 Nobel Prize in Physics*"

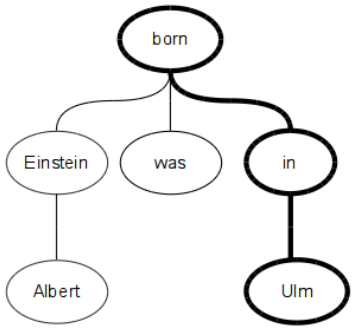

Figure 2: Parse tree for the phrase "Albert Einstein was born in Ulm."

In Figure 1 is shown the syntactic parse tree for the sentence "He received the 1921 Nobel Prize in Physics" without the corresponding syntactic elements. The DBpedia property that connects "Albert Einstein" and "The Nobel Prize" and for which we want to identify a lexicalization is the "award" property. These are the characteristics of the search:

Subject: Albert Einstein
Object: Nobel Prize
Property: Award
Output: receive

In Figure 2 is presented a similar parse tree from which we can extract the following information:

Subject: Albert Einstein
Object: Ulm
Property: BirthPlace
Output: born in

We can see that in the second example the found expression of a property is the path from the verb to the value of the DBpedia property (in this case "BirthPlace") excluding its value.

Pattern Generation

In order to generate ChatScript files for a specific person, we fetch that person's Wikipedia page, split it into phrases and keep only those that have the person as a subject. This filtering was done using the Stanford Deterministic Co-reference Resolution System. After extracting all the phrases referring to the current historical figure, we select only those that express a property from DBpedia matching the expression against the knowledge base. We then create a rule-answer entry to add to the ChatScript files. The rule is represented as an expression of a property that appears both in the analyzed sentence and the knowledge base. The answer is the analyzed sentence from Wikipedia which is converted to be expressed in the first person. The conversion from the third person to the first person of the sentence is accomplished with Stanford's Part-of-Speech Tagger and CoreNLP. All these patterns are written in ChatScript's file hierarchy. For a fast and easier way to find the answer, we arranged ChatScript's files by the properties of the person.

INTEGRATION OF ANSWER SENTENCE SELECTION

Because it is impossible to build an exhaustive rule-based system, a secondary approach to this problem has to be taken into consideration in order to give a good answer. In addition, ChatScript has its own limitations coming from the fact that it ignores a rule after it first matches it. Therefore, a fallback option is needed in case the former approach fails to provide an answer.

Considering the fact that the former approach gives better answers the simpler and more common questions are asked, we observe that either it fails to match questions that are more complex or it has too many matches for a question that uses a common verb (like "to be" or "to have") and the results will be inaccurate or noisy. The solution to avoid this is to try and find the answer directly from the source of the previously described knowledge base with an ad hoc approach considering every sentence from that respective source.

Answer sentence selection

Following the goal of having to answer a question for a certain historical figure, the set of possible answers is reduced to a set of sentences from that person's biography. This leaves us with the task of identifying a sentence from a biography that has the highest probability of correctly answering the question at hand.

Considering what was previously stated, that this approach tries to find the answer to a more complex question, we can assume that, at least for now, there is a great deal of semantic information embedded in the form of the question (lexically and semantically) so that the chances are a part of the answer textually lies in the question. Therefore, what we can do is actually search for the question (or paraphrases of the initial question) in the reference text.

To get the best results out of this approach, we need to follow a number of steps.

First, we need to remove unnecessary words, including stop words, the interrogative words (what, when, where, who, why and how) and irrelevant verbs ("to be", "to have" and

other similar verbs as described above) in order to remain only with meaningful words, i.e. the kernel of the question.

Second, we want to use the Stanford CoreNLP software to lemmatize the question (i.e. to convert every word to its appropriate canonical form) because, as described later on, the corpus used for a historical figure will be lemmatized too. This will help in the search step because words will more likely match if they are in their base form.

Third, we try to find alternative ways of expressing the input question and attempt to search for all these variants in the biographical text. We do this by trying different synonyms for the words in the question so that we can get more results, even if the initial question is formulated in such a way that it does not contain the exact words that might appear in the sentence representing the correct answer.

Next, the top paragraphs from the Wikipedia article are filtered based on the textual matching score between the question and the respective paragraph given by Apache Lucene[5], a specialized text indexing and searching tool. Then, we apply the same algorithm at the level of sentences instead of paragraphs. In short, to get the best answer the corpus is divided in separate paragraphs and a small set of paragraphs where the answer sentence might be part of are selected. Next, we attempt to find an even smaller set, made of sentences that are the best candidates to answer the question.

After we have a set of sentences that passed the lexical filtering, we want to eliminate those in which the subject of the sentence does not match the subject of the question. This mechanism is similar to maintaining the semantic relations as described in [1], and presented above in the Related Work section. To achieve that, we want to use Stanford CoreNLP, and in particular the Stanford Deterministic Co-reference Resolution System, to determine who is the subject of a given sentence.

After the syntactic filtering we are left only with the semantic filtering. This means we want to filter out all the sentences that do not have the type as the one expected by the question. For example, questions starting with "When" expect an answer sentence that contains a numerical value.

Finally, we choose the first sentence in order of the previously gathered relevance scores. Using the aforementioned approach, we manage to answer more complex questions.

Algorithm

1. fetch the question for a historical figure
2. fetch the biography text for that person
3. split the text into paragraphs
4. lemmatize the question and all the paragraphs
5. find a set of top paragraphs against which the lemmatized question matches
6. reduce this set to a set of sentences from these paragraphs against which the lemmatized question match
7. eliminate the sentences that do not refer to the subject of the question
8. filter out the sentences that don't respect the expected answer type
9. return the best answer found

RESULTS

Knowledge base approach

Because a conversational agent is an interactive system, verifying and validating it is not an easy task, therefore the only way of testing the chat-bot was by asking questions and analyzing the returned results. Most of the test questions were conceived by the development team. The first time a historical person is chosen to interact with, we generate all the rules (questions and answers) that can be extracted from the knowledge base which are written in ChatScript format. After this step, each time this historical person is selected, we use the already generated data from the ChatScript. Because of this the response time is less than 1 ms.

So far, the conversational agent provides answers to simple questions that express properties from DBpedia. Some examples follow below.

Examples for Albert Einstein bot

Q: Where were you born?
A: I was born in Ulm, in the Kingdom of Wurttemberg in the German Empire on 14 March 1879.
Q: When did you die?
A: I died in Princeton Hospital early the next morning at the age of 76, having continued to work until near the end.
Q: What award did you receive?
A: I received numerous awards and honors, including the Nobel Prize in Physics.
Q: What schools did you attend as a child?
A: I attended a Catholic elementary school from the age of 5 for three years.
Q: Who were you married with?
A: I married Elsa Löwenthal on 2 June 1919, after having had a relationship with her since 1912.

Personality	Different answer
John F. Kennedy	146
Abraham Lincoln	141
Ronald Regan	131
Adolf Hitler	129
George W. Bush	125

Table 2: The number of different answers given by the bot.

[5] Apache Lucene, online at https://lucene.apache.org/.

A way of validating this method is done through the number of different answers we can provide, which are presented in Table 2.

Answer sentence selection method

The testing of the answer selection methods was done for a set of 5 personalities and about 20 questions for each personality, some of them general, others more specific. What was observed during the testing is that, in the majority of cases, a correct answer was somewhere in the top 10 sentences outputted by the lexical pipeline.

Personality	P@1	P@2	P@3
John Lennon	40%	60%	80%
Albert Einstein	30%	60%	80%
Napoleon	30%	40%	40%
Charlie Chaplin	42%	57%	57%
Adolf Hitler	50%	83%	83%

Table 3: The percentage of correct answers for various personalities in case of the first (P@1), the first two answers (P@2) and the first three answers (P@3).

From the tests done, it is noticeable that all the answers took less than 700 ms. Here we present several examples from a discussion with the conversational agent using the answer sentence selection method. The chat-bot impersonates John Lennon:

Q: What was your debut album?
A: My emotional debut solo album, John Lennon/Plastic Ono Band (1970), was received with high praise.
Q: Who shot you?
A: At around 10:50 pm on 8 December 1980, as me and Ono returned to their New York apartment in the Dakota, Mark David Chapman shot me in the back four times at the entrance to the building.

CONCLUSIONS

This paper presented the two methods used together to implement a conversational agent that models a historical figure: the first method of generating ChatScript files using ontologies extracted from DBpedia and the fallback method using answer sentence selection with textual matching. The advantage of implementing a chat-bot that answers trivia questions from the perspective of a historical personality is that for each person, there is a small amount of information to be processed. In addition, the questions are easy to predict, unlike questions in a general purpose open-domain question answering system.

ACKNOWLEDGMENTS

This work has been partly funded by the Sectorial Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of European Funds through the Financial Agreements POSDRU/159/1.5/S/132397 and by POSDRU/155420 – PROSCIENCE.

REFERENCES

1. A. Echihabi, U. Hermjakob, E. Hovy, D. Marcu, E. Melz, and D. Ravichandran, "How to Select an Answer String?," *Adv. open domain Quest. answering*, pp. 383–406, 2006.

2. D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty, "Building Watson: An Overview of the DeepQA Project," AI Magazine, vol. 31, no. 3. pp. 59–79, 2010.

3. F. Giunchiglia, M. Yatskevich, and P. Shvaiko, "Semantic matching: Algorithms and implementation," *J. Data Semant.*, 2007.

4. F. Giunchiglia and P. Shvaiko, "Semantic Matching," *Knowl. Eng. Rev. J.*, vol. 18, no. 3, pp. 265–280, 2004.

5. T. R. Gruber, "A translation approach to portable ontology specifications," *Knowl. Acquis.*, vol. 5, no. 2, pp. 199–220, 1993.

6. B. Heller, M. Procter, and D. Mah, "Freudbot: An investigation of chatbot technology in distance education," *Proc. World Conf. Educ. Multimedia, Hypermedia Telecommun.*, pp. 3913–3918, 2005.

7. C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard and David McClosky, "The Stanford CoreNLP Natural Language Processing Toolkit", *ACL 2014*, 2014.

8. M. Mehta and A. Corradini, "Developing a Conversational Agent Using Ontologies," *HCI'07 Proc. 12th Int. Conf. Human-computer Interact. Intell. multimodal Interact. Environ.*, pp. 154–164, 2007.

9. K. Taylor and S. Moore, "Adding question answering to an e-tutor for programming languages," in *Applications and Innovations in Intelligent Systems XIV*, 2007, pp. 193–206.

10. A. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

11. R. Wallace, "The elements of AIML style," Alice AI Found., 2003.

12. J. Weizenbaum, "ELIZA — A Computer Program For the Study of Natural Language Communication Between Man And Machine," *Commun. ACM*, vol. 9, no. 1, pp. 36–45, 1966.

13. B. Wilcox, "Beyond Façade: Pattern Matching for Natural Language Applications ", Telltale Games, 2011.