A Lightweight Story-Comprehension Approach to Game Dialogue

Robert van Leeuwen, Yun-Gyung Cheong, and Mark J. Nelson

Center for Computer Games Research
ITU Copenhagen
Copenhagen, Denmark
luivatra@gmail.com, yugc@itu.dk,mjas@itu.dk

Abstract. In this paper we describe Answery, a rule-based system that allows authors to specify game characters' background stories in natural language. The system parses these background stories, applies transformation rules to turn them into semantic content, and generates dialogue during gameplay by posing it as a question-answering problem. By the means of simple categorization combined with rule inference engine, our system can generate answers efficiently. Our initial pilot study shows that this approach is promising.

Keywords: story comprehension, question and answer, dialogue

1 Introduction

An important part of story-driven videogames is the dialogue. To fully participate in gameplay, dialogue should be dynamic, varying based on players' actions and the game state. Adding a complication to achieving this goal, most games are made by teams with diverse skills; for example, engineers write the code and gameplay systems, and designers and writers may be able to perform lightweight scripting but are typically not programmers.

A common way of factoring engineering versus design/writing is as follows. A programmer implements gameplay systems, and exposes certain attributes that can be modified by a designer via an interface. Character traits are particularly common to expose in this manner, such as a particular character's personality and skills. On the dialogue side, the programmer builds a dialogue system that, while simple enough to be used by non-programmers, can query attributes and the current game state in order to vary the dialogue. Common approaches include branching dialogue trees that use conditional tests, and templated dialogue that substitutes certain words or phrases according to the values of variables.

As character attributes and in-game interaction vary more, there is a content-authoring bottleneck, since authors must write dialogue for all permutations of possible situations [9]. In our work, we are motivated by the idea that part of a character's personality is formed by their backstory, which can take on very different characteristics for different characters, and impact the story in varying ways—so backstory cannot be boiled down to a small number of traits in fixed slots. Instead, our system allows the author to specify character backstories in natural language.

There have been several experiments with natural language as an authoring method for interactive stories [7, 11]. Nelson [11] argues "the natural language in which to write interactive fiction is natural language", and overhauled the popular Inform authoring system to use a natural-language-like syntax in Inform 7.[1] It remains only natural-language-*like*, because it uses a strict deterministic parser, in which the language's concrete syntax, despite looking like natural language, is really a sort of programming language. We wish to relax that restriction by using a broad-coverage off-the-shelf parser. This means that more input is accepted, but that the result of a parse is more ambiguous semantically than a programming language's abstract syntax tree.

Transformation rules supply the semantic content, specifying how grammatical components in a parse structure are to be mapped to character-backstory attributes. This approach has some similarities to the hybrid approach of Khoury [8], which also uses a statistical parser followed by a rule-processing step to achieve broad coverage with semantic specificity. In addition, we retain the original surface text to use in the question-answering phase, both to synthesize and to rank question answers, allowing linguistic features not captured in the semantic mapping step to still make their way into answers.

A particular feature of the videogame-dialogue setting is that, during the authoring phase, speed is not a major consideration, and authors are willing to do at least moderate iteration to correct errors; whereas during gameplay, the system must be efficient and lightweight. This differs from the situation with many query-answering systems, which use shallow NLP techniques to scan a large database of texts to find segments responsive to a query online, rather than doing full parses of all the source texts ahead of time. In the videogame setting, it's preferable to do heavyweight processing beforehand. In addition, some degree of errors in parsing author input are acceptable, since authors can notice the misparses of their character backstories, and reword them to avoid errors; but during gameplay, speed is critical and error rates should be low.

2 System architecture

Answery takes a story background containing facts about story characters written in natural language, and outputs a relevant answer to a question about the backstory. As shown in Figure 1, Answery consists of four components: parser, post-parse processing, reasoning, and answer ranker. The first step is to parse the input story text into a grammatical functional structure, for which we use an off-the-shelf statistical parser that outputs word dependencies. Since parsing is the focus of this work, we constrain the input text to simple sentences. In post-parse processing, the sentence is split into *core* parts of the sentence (nouns and verbs) and *decorative* elements such as adjective and adverb. The

[1] `http://www.inform7.com`

core facts are given to the reasoning engine to infer high-level knowledge using simple rule-based schemas. In the final step, candidate answers are ranked based on their similarity to the given question.

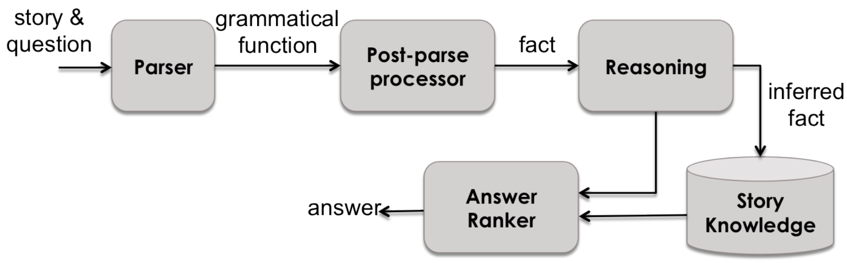

Fig. 1. The system architecture

2.1 Parser

We employ RASP (Robust Accurate Statistical Parser) [1], a corpus-trained statistical parser. The parser labels each word part-of-speech tag (POS), indicating the grammatical function. RASP uses a hidden Markov model (HMM) to determine POS as the most likely sentence structure. The output of the parser consists of a list of Grammar Relations (GR), where the format is represented as a tuple of *(type sub-type head dependent initial)*. In the tuple, type, head, and dependent are mandatory parameters while the rest are optional. For instance, when "I am Frodo" is given as input, the parser returns two GRs—*(ncsubj be I), (xcomp be Frodo)*—representing a non-clausal subject and a complement, respectively.

2.2 Post-parse processing component

The post-parsing step extracts facts based on patterns in the grammar relations frequently found in the training set.

(ncsubj,1,2)p + (xcomp,3,4)p → (2,1,4)
where 1 == 3
(4.type = NP1 OR 4.type = NN1 OR 4.type = JJ)

In the example pattern above, the first line consists of a head and a tail, separated by '→'. The rest of the lines are conditions for the rule application. This pattern will extract a fact of (2,1,4) if 2 GR-elements of type *ncsubj* and *xcomp* appear where the values for 1 and 3 are identical, and the type of 4 falls in one of the options *(NP1, NN1, or JJ)* in the second rule. Following the CLAWS4 Tagset [6], NP1 means singular proper noun, NN1 denotes singular

Table 1. Fact categorization rule examples

Category	Condition Rule
location	If a pronoun is preceded by a preposition indicating a position
property	If a fact is an adjective, adverb or is positioned after a determiner
timeproperty	If a fact is a general adverb or a nominal adverb of time
reason	If a conjunction is preceded by a verb in a 3-tuple fact
time	If a fact is a time-related adverb, preposition conjunction, or temporal noun

common noun, and JJ denotes general (comparative) adjective. When the input example is *(ncsubj be I)* and *(xcomp be Frodo)*, the system can infer the core fact *(I,be,Frodo).* This pattern language also converts the query into an easier sentence for answer retrieval. For instance, the question "Are you Frodo?" can be converted into "Am I Frodo?" (interpreting from the character's perspective) then checked with the fact in the story knowledge base, *(I, be, Frodo).* When it matches with a fact in the story, the answer is yes.

2.3 Reasoning component

The reasoning component performs two roles: fact categorization and inference. The fact tuples are asserted into a reasoning engine (Jess [5], a Rete-based forward-chaining engine [4]) and used to infer high-level knowledge based on rules and relationships between types. A categorization inference rule is defined as a pair, of condition and category, as shown in Table 1. A fact is categorized into the category in the left-hand column when it meets the condition defined in the right-hand column. When a fact fails to match any conditions, its default category is noun in this study. When a question is given, the system determines candidate fact categories based on Table 2. For instance, if the question contains "when", the system returns the facts in the categories of Time or TimeProperty.

2.4 Answer ranker

When multiple answers are available for a given query, the ranker estimates the relevance of each answer, based on syntactic and word similarity between the query and the answer. Word similarity is simply percentage of words that appear in both the question and answer. Syntactic similarity considers the parts of

Table 2. Question type and candidate answer types

Question type	Candidate answer types
Who	Person
What	Noun,Property,TimeProperty
When	Time,TimeProperty
Where	Location
Why	Reason

speech words appear as in both the question and answer. Take an example of a background story containing two sentences: ”John killed George” and ”George killed John”. With the query ”Who killed John?”, word similarity rates the two sentences equally relevant. However, the second sentence, ”George killed John”, has higher syntactic similarity, since the part of the speech of *John* is the object, the same as in the query sentence. The weighted sum of these similarities ranges from 0.0 and 1.0. Four combinations of weights for syntactic similarity/word similarity (i.e., 0.5/0.5, 0.75/0.25, 0.25/0.75, 0.0/1.0) were tested to find the optimal setting. In addition, a predefined value is used as threshold to determine if any potential answer is sufficiently good to be returned at all. When no answers are over the threshold, Answery returns ”I do not know.” On the other hand, it returns ”I do not understand” if no matching patterns are found. We give distinct importance to different word types in ranking answers. Nouns are weighted highly (1.0 in this study) as they are key to sentence comprehension, while common verbs (*be*, *have*, *do*) are given low weights (0.05 in this study).

3 Evaluation and Discussion

In order to evaluate the effectiveness of the system we collected 9 different story backgrounds from different authors. These stories are divided into two sets: 5 stories and 67 questions for each story in the training set and 4 stories and 40 questions for each story in the test set. We re-wrote these stories in simple sentence forms to make the text input to Answery conform to the constraints in parsing. One story example used in the study is shown in 2. The training set is used to extract 23 patterns for the post-parse processing phase and 15 rules for the fact categorization.

I am Frodo. I am a hobbit and am born in Bag End. Bag End is a town in The Shire. Hobbits like tobacco and beer. I love walking. I am very interested in Elves. Elves live in Rivendel, far to the east of Bag End. I have an uncle called Bilbo. Bilbo is very rich so Bilbo and I live in a big house and don't have to work for a living. The gardener's son, Sam, is my best friend. Sam and I love to go out and drink a lot of beer. Gandalf is a wizard and is an old friend of Bilbo. Gandalf makes amazing fireworks. Lately Bilbo has been acting strange. Gandalf thinks Bilbo is acting strange because the ring he carries around is evil. Bilbo decides to move to Rivendel so Bilbo can live in peace amongst the elves. Bilbo gives the ring to me before he leaves. After some months Gandalf tells me the ring is very evil and has to be destroyed. To destroy the ring I have to bring the ring to Rivendel. Sam and I walk to Bree, a town between Bag End and Rivendel. In Bree we stay at an inn and meet Strider. Strider is a friend of Gandalf and helps us avoid evil creatures. The evil creatures are after the ring. After many hardships we reach Rivendel.

Fig. 2. A story background example

Table 3. Evaluation Results

Set	Correct answer	False positive	False negative
Training	329 (98.2%)	2 (0.6%)	4 (1.2%)
Test	147 (91.0%)	11 (6.9%)	2 (1.3%)

Table 3 shows the results of a pilot evaluation, which found high precision (91%) in generating correct answers. The system generated correct answers 98.2% in the training set and 91% in the test set. False negatives are when the system returned no answer to the query despite correct answers existing. False positives are when the system returned an answer even though no relevant information existed in the backstory. For instance, the correct answer for the question "Where can I get a lot of money?" is "I don't know", but the system returned the incorrect answer "Sam and I drink a lot of beer" due to the phrase "a lot of" appearing in both the query and answer. Note that the system achieves high precision in answering to questions with its relatively poor performance of fact categorization. Among the 239 facts, 27 facts were incorrectly classified, resulting in 11.3% error. Location and Time classes account for the majority of these errors, because Locations are often classified as Noun or Person, and Time keywords are often recognized as a Property.

Situating these numbers is complicated by the lack of an agreed-upon benchmark problem, however. Chen *et al.* [3] reported 53% accuracy when questions are available, and 43% of total questions, but their results are not directly comparable. Their system is fully automated, extracting domain knowledge (analogous to the inference schemas in our system) from text without manual processing, and their characters answer players' questions based on factual information derived from Wikipedia, rather than fictional backstories.

4 Conclusions and Future Work

This paper describes a lightweight ontology-based story comprehension system. Answery takes a story background as input and returns a relevant answer to a question, by parsing the story background, mapping it to core character traits, and then evaluating grammatical and semantic similarities between question and answer sentences. In a pilot study we carried out the system showed high precision in generating answers to W-questions and yes/no questions. We acknowledge that the high performance achieved in this pilot study is not directly comparable to the state of the art in the question and answer community, due to our syntactic restrictions on the input text. However, we believe that this project is a practical solution to alleviate the game designer's burden in authoring game dialogue.

Our future work is two-fold: system improvement and user experience evaluation. First, we plan to use a lexical database such as WordNet [10] to enhance categorization accuracy, since it can be used to build useful semantic-distance metrics [2]. Second, we will look into how authors and players perceive the system. For authors, it can be compared to more traditional slot-based authoring approaches; for players, it will be interesting to observe player behavior when incorrect answers are presented, as an attempt to understand the impact of different kinds of dialogue errors in games.

Acknowledgements

This work has been supported in part by the EU FP7 ICT project SIREN (project no: 258453).

References

1. Briscoe, T., Carroll, J., Watson, R.: The second release of the RASP system. In: COLING/ACL 2006 Interactive Presentation Sessions. pp. 77–80 (2006)
2. Budanitsky, A., Hirst, G.: Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In: Workshop on WordNet and other Lexical Resources (2001)
3. Chen, G., Tosch, E., Artstein, R., Leuski, A., Traum, D.R.: Evaluating conversational characters created through question generation. In: Proceedings of FLAIRS 2011. pp. 343–344 (2011)
4. Forgy, C.: Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence 19, 17–37 (1982)
5. Friedman-Hill, E.: Jess in Action. Person Education (2003)
6. Garside, R., Smith, N.: A hybrid grammatical tagger: CLAWS4. In: Garside, R., Leech, G., McEnery, A. (eds.) Corpus Annotation: Linguistic Information from Computer Text Corpora, pp. 102–121. Longman (1997)
7. Kacmarcik, G.: Using natural language to manage NPC dialog. In: Proceedings of the 2nd Artificial Intelligence and Interactive Digital Entertainment Conference. pp. 115–117 (2006)
8. Khoury, R., Karray, F., Kame, M.S.: Keyword extraction rules based on a part-of-speech hierarchy. International Journal of Advanced Media and Communication 2(2), 138–153 (2008)
9. Mateas, M.: The authoring bottleneck in creating AI-based interactive stories. In: Proceedings of the AAAI 2007 Fall Symposium on Intelligent Narrative Technologies (2007)
10. Miller, G.A.: WordNet: A lexical database for English. Communications of the ACM 38(11), 39–41 (1995)
11. Nelson, G.: Natural language, semantic analysis, and interactive fiction. Online whitepaper. `http://www.inform-fiction.org/I7Downloads/Documents/WhitePaper.pdf` (2006)