ComicKit: Acquiring Story Scripts Using Common Sense Feedback

Ryan Williams
MIT Media Lab
20 Ames St
Cambridge MA
breath@mit.edu

Barbara Barry
MIT Media Lab
20 Ames St
Cambridge MA
barbara@media.mit.edu

Push Singh
MIT Media Lab
20 Ames St
Cambridge MA
push@media.mit.edu

ABSTRACT

At the Media Lab we are developing a resource called StoryNet, a very-large database of story scripts that can be used for commonsense reasoning by computers. This paper introduces ComicKit, an interface for acquiring StoryNet scripts from casual internet users. The core element of the interface is its ability to dynamically make common-sense suggestions that guide user story construction. We describe the encouraging results of a preliminary user study, and discuss future directions for ComicKit.

Categories and Subject Descriptors

I.2.6 [**Learning**]: Knowledge Acquisition; H.5.2 [**User Interfaces**]: Interaction Styles

General Terms

Human Factors

Keywords

Common sense, Story Representation, Knowledge Acquisition, Case-Based Reasoning.

1. INTRODUCTION

At the Media Lab we are developing a resource called StoryNet, a very-large database of story-scripts that can be used for case-based commonsense reasoning by computers [7]. Our goal is to acquire a corpus of one million unique story-scripts of the breadth, depth and quantity of knowledge to understand real world situations.

This paper introduces ComicKit, a novel web-based interface for acquiring story knowledge from thousands of users over the internet. ComicKit uses the distributed knowledge capture method that was shown to be effective in the Open Mind Common Sense project (OMCS) [6][1]. OMCS succeeded at collecting about 700,000 simple commonsense facts from 15,000 contributors around the world. ComicKit takes the same approach – casual, non-expert internet users contribute commonsense knowledge via simple and engaging knowledge acquisition activities on a public website.

[1]`http://openmind.media.mit.edu/`

One challenge we face is that people are natural-born storytellers. How can we create a system that can guide the user to contribute stories that cohere to a machine representation of stories, and guides the collection of commonsense stories, rather than fantastical or unreal stories?

We wanted a design that allowed users to enter any desired story content, yet hinted them to input them in a format compatible with a script-like representation. It should be easy for the player to create the type of story that satisfies the script representation of StoryNet, and more difficult to create story scripts that defy easy entry into StoryNet. A comic-creation format satisfies our needs because it isn't free-form text entry, yet through a combination of graphical elements can represent a broad range of stories.

The suggestion aspect of the ComicKit is its core functionality, and serves three purposes: guiding the user into creating a usable story script; reducing the amount of work the user has to spend to create a story; and making the experience more fun through feedback from the inference engine. The suggestion engine bases its suggestions on the content of the story as the user creates it, and referencing a sizeable body of existing common sense knowledge. For example, the system knows that if a character were near a phone, a phone call would be a sensible next action and presents that suggestion to the user. In the future, we intend to study how often people take the suggestions and how often they create their own content.

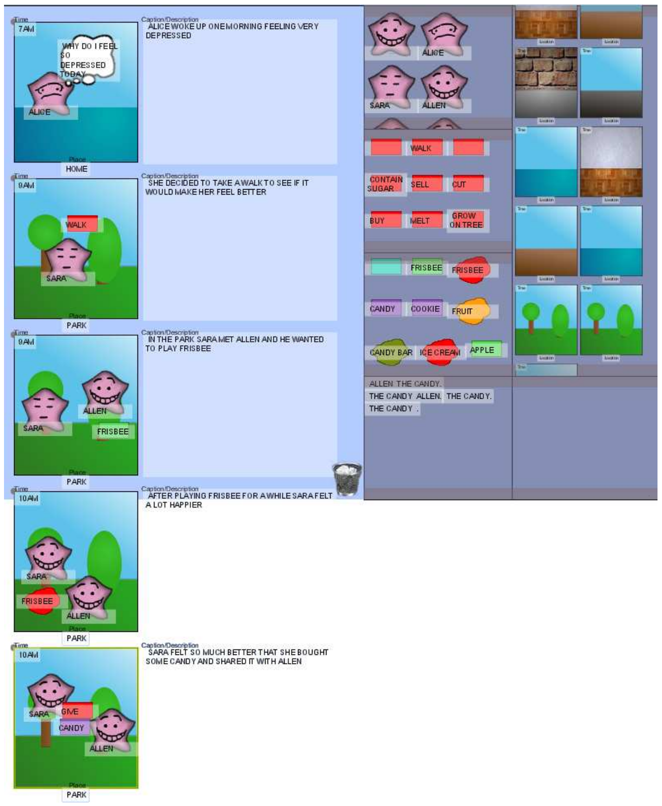
The suggestion engine uses an existing commonsense reasoning system, ConceptNet, whose knowledge base consists of a large semantic network mined from OMCS data, to generate educated guesses about what the user could add to their story [5]. Educated guesses are cited as principle in human teaching and learning that can be used as an effective knowledge acquisition method [3], and in addition, gives the system the quality of a playful storytelling partner.[4]

2. COMICKIT SYSTEM

Over the summer, we wrote a proof-of-concept mockup of the ComicKit in Java. An example of the interface is shown in Figure 2. In the example, the entire comic is shown to illustrate one of the comics created during our user test. On screen the panels below the blue region would be cropped, and the user would scroll the workspace to view them.

The left half of the interface is the workspace, where panels are composed into a story. The right half contains several palettes, each of which contains a different type of scene component. From the top left down these elements are people, actions, things, captions, panels, and thoughts/speech.

In the example (Figure 2), the second visible frame contains a person "Sara", and a "walk" action. Note that each palette contains a blank component in addition to several suggested components. When playing, a player will drag a panel to the workspace, then drag the other elements (things, people, actions, thoughts) onto the panel. By hovering the mouse over a component, the player is given the ability to edit the text in that component.



ALICE WOKE UP ONE MORNING FEELING VERY DEPRESSED

SHE DECIDED TO TAKE A WALK TO SEE IF IT WOULD MAKE HER FEEL BETTER

IN THE PARK SARA MET ALLEN AND HE WANTED TO PLAY FRISBEE

AFTER PLAYING FRISBEE FOR A WHILE SARA FELT A LOT HAPPIER

SARA FELT SO MUCH BETTER THAT SHE BOUGHT SOME CANDY AND SHARED IT WITH ALLEN

ALLEN THE CANDY.
THE CANDY ALLEN. THE CANDY.
THE CANDY .

Figure 1: ComicKit interface

The ComicKit is a Java applet that acts as a client to a centralized XML-RPC server. The server, written in Python, is connected to ConceptNet. Every time the player makes a change to some element of a panel, the client sends its state to the server, and receives in return a list of suggestions for the palettes. The next version of ComicKit will enable users to save and retrieve their comics, to and from the server. The future version of the ComicKit interface will be written in Flash, but the server as a back-end will not require changes to support the new front-end.

In order to make suggestions in the palettes, the back-end parses the contents of the panels on the workspace, and makes three types of queries to ConceptNet, the results of which are used to populate the palettes. The first type of query retrieves objects that are similar to the objects in the workspace. The second type of query returns the actions that are likely to be performed by or on the objects in the workspace. The third type of query generates sentences for the caption by combinatorially guessing the subject-object relationships and letting the user select the best caption.

When the suggestions fail to suit the player, it is a very simple matter for the player to relabel any element or type a new caption. Every text field is editable, and empty-label elements are always available for relabeling. The only elements that are never suggested are thoughts/speech.

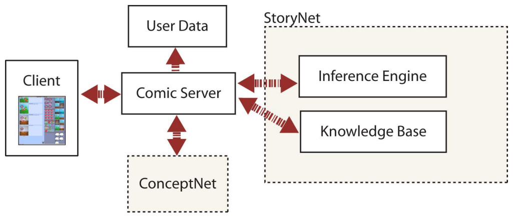


Figure 2: Dataflow in the ComicKit system

The existing interface should be thought of as a platform for story script acquisition games, rather than an end in itself. The platform allows the user to create stories by dragging icons into panels and occasionally typing new labels. Around this platform could be built a variety of future activities that aim to increase player engagement. For example, we could distribute a small Flash application that displays a stored comic, which players could download and use on their own websites to display comics they have created.

3. RESULTS

In evaluating the game, we wanted to see whether the comic interface was more effective than a plain text interface, and if the intelligent suggestions were helpful. We ran an experiment that pitted three story-entry applications against each other.

1. **SN-Text** The first was a Flash-based story generator that had users dragging sentences into order. The sentences originated in a related common sense resource called LifeNet [8], and were not modifiable. In order to tell their story, users had to repeatedly search the LifeNet database for the existing sentence that most closely matched their intent.

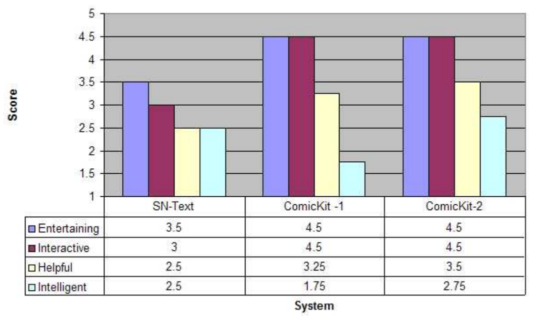
2. **ComicKit-1** The second tested application was the Comic Kit's interface as describe, but with no suggestions of elements from the server side.

3. **ComicKit-2** The last tested application was the fully intelligent ComicKit with suggestions for elements and captions.

For each application, subjects were asked to create three stories, then rate the application on three scales: entertainment value, helpfulness, and intelligence. We collected data from five grad students/professors at the MIT Media Lab for the preliminary study, and their average subjective responses are charted in Figure 3. We believe that for this test at least one user was not able to connect to the server and thus her experience for parts 2 and 3 were identical.

The evaluators found the ComicKit interface to be more rewarding than ordering sentences into a list to tell a story. They also found that the ComicKit with suggestions was more intelligent, and more helpful.

It was fairly clear even during the experiment that the evaluators were using the three applications in different ways.



	SN-Text	ComicKit -1	ComicKit-2
▨ Entertaining	3.5	4.5	4.5
■ Interactive	3	4.5	4.5
□ Helpful	2.5	3.25	3.5
□ Intelligent	2.5	1.75	2.75

Figure 3: User Evaluation Subjective Results

They spent about ten minutes total creating stories on the text interface, and about two hours on the two versions of ComicKit, much more than we expected or asked. Some of them needed to be reminded of the time before they stopped. It is possible that the comic interface is very inefficient, and thus it took the users much longer to create an equivalent story. The other possibility is that the users found the comic interface more engrossing, and created longer and more involved stories with it.

It is hard to compare the experience of creating a story with the ComicKit with the text story creation utility. Users commented that the comic medium was richer, allowing them to tell the story both graphically in the panels, and in the text captions. They felt limited by the suggestions provided, and responded to this limitation in two ways. Some let the suggestion engine inspire them when their own ideas were running low, and sometimes they made choices they wouldn't have made on their own. Others felt challenged by the limitations, and went out of their way to overcome them. Because each label was completely editable, the only real limitation was in the amount of typing the user was willing to do in order to tell their story.

Most people used the thought and speech bubbles much more than we had anticipated – most people had at least one thought bubble in every panel. They were the primary means for people to get around the simplistic subject-object-verb model created by the objects in the interface. Thought bubbles provide an important insight into the mental state of story script characters, but unfortunately they can take on difficult-to-understand forms due to their freeform nature.

We tracked the number of scene elements that each person composed into their comic strip, as well as the number of steps they took to generate each strip. The average values of these are presented below for comparison.

	ComicKit-1	**ComicKit-2**
Elements Used	13	17
Modifications	50	76

These values show that the users created comics with more elements when the interface included suggestions, and they spent more time trying out combinations. The interface afforded "playing", since virtually everything in it was interactive.

4. RELATED WORK

ComicKit is superficially similar to other story construction systems, but departs significantly in its purpose. The ComicKit can be thought of as a graphical knowledge representation and generation tool, whereas other projects have focused on the learning of the user [9], fostering communication skills [2], and story authoring[1].

Comic Kit as a story creation tool is similar to StoryWriter because users construct a story graphically using a palette of story elements and captions. ComicKit is distinguished by its use of story suggestions and abstract appearance of story elements. ComicKit does not have representational objects like StoryWriter[9] does – a ComicKit pear does not bear any visual resemblance to a real pear, whereas in StoryWriter pears are identified and used based on their appearance. By deliberately dissociating the appearance of the object from its meaning, we allow the scope of the stories to be limited only by the user's imagination. ComicKit makes suggestions to the user in the manner of an intelligent story agent[1].

5. IMPLICATIONS

We believe that this study showed that ComicKit is likely to achieve our goals as a knowledge acquisition interface for StoryNet. As we acquire story script knowledge, we gain the ability to feed this knowledge back into the ComicKit through suggestions. This feedback loop will improve the sophistication of the system, and encourage users to play with it more. While we have yet to test the collected knowledge in an actual case-based commonsense reasoning system, ComicKit gets us closer to our goal of acquiring the very-large corpus of story-scripts that such a system would need to reason robustly about a wide range of commonsense scenarios.

6. ACKNOWLEDGMENTS

This research is being supported by the sponsors of the MIT Media Lab, in particular Warner Brothers Interactive Entertainment. Thanks to Glorianna Davenport and Marvin Minsky for their guidance.

7. REFERENCES

[1] K. M. Brooks. Do story agents use rocking chairs? the theory and implementation of one model for computational narrative. In *Proceedings of the fourth ACM international conference on Multimedia*, pages 317–328. ACM Press, 1996.

[2] A. Druin, J. Stewart, D. Proft, B. Bederson, and J. Hollan. Kidpad: a design collaboration between children, technologists, and educators. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 463–470. ACM Press, 1997.

[3] Y. Gil and H. Kim. Interactive knowledge acquisition tools: A tutoring perspective. In *Proceedings of the 24th Annual Meeting of the Cognitive Science Society (CogSci), Fairfax, VA, August 8-10*, 2002.

[4] H. Lieberman, H. Liu, P. Singh, and B. Barry. Beating some common sense into interactive applications. *AI Magazine, To Appear Fall 2004. AAAI Press*, 2004.

[5] H. Liu and P. Singh. Conceptnet: a practical commonsense reasoning toolkit. *BT Technology Journal*, 22(4):201–210, 2004.

[6] P. Singh. The public acquisition of commonsense knowledge. In *AAAI Spring Symposium: Acquiring (and Using) Linguistic (and World) Knowledge for Information Access*, Palo Alto, CA, 2002. AAAI, AAAI.

[7] P. Singh, B. Barry, and H. Liu. Teaching machines about everyday life. *BT Technology Journal*, 22(4):227–240, 2004.

[8] P. Singh and W. Williams. Lifenet: a propositional model of ordinary human activity. In *Proceedings of the Workshop on Distributed and Collaborative Knowledge Capture (DC-KCAP)*, 2003.

[9] K. E. Steiner and T. G. Moher. Graphic storywriter: an interactive environment for emergent storytelling. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 357–364. ACM Press, 1992.