Question-Answering with Logic Specific to Video Games

Corentin Dumont, Ran Tian, Kentaro Inui
Tohoku University,
6-6-05 Sendai, Miyagi 980-8579, Japan

{corentin-d, tianran, inui}@ecei.tohoku.ac.jp

Abstract

We present a corpus and a knowledge database aiming at Question-Answering in a new context, namely the open world video games. We choose a popular game called Minecraft, and created a QA corpus with knowledge database related to this game. We are interested in the logic rules specific to the game, which may not exist in the real world. The ultimate goal of this research is to build a QA system that can answer natural language questions from players with the ability of inference on these game-specific logic rules. The QA corpus is partially composed of online quizzes questions and partially composed of manually written variations of the most relevant ones. The knowledge database is extracted from several wiki-like websites about Minecraft. It is composed of unstructured data such as text and structured data such as infoboxes. A preliminary examination of the data shows that players are asking creative questions about the game, and the QA corpus can be used for clustering verbs and linking them to predefined actions in the game.

Keywords: question-answering, knowledge acquisition, structured database, word clustering, name entity recognition

1 Introduction

This paper presents a corpus and a knowledge database aiming at creating a Question-Answering system specific to a new context, namely the open world video games. Unlike many QA systems that are designed to answer real world questions [**?**, **?**], the goal of this research is to build a system that can answer questions using the logic specific to the game, which may not be identical to the logic in the real world. We choose a popular game called Minecraft, of which the openness provides a great liberty for players, thus it guarantees a large number of possible questions to ask about the game, and yet there is a specific logic that limits the actions of players (Section 2). We are interested in this problem setting because it could provide a testbed for combining Natural Language Processing with advanced logical inference techniques. The QA corpus is partially collected from quiz websites and partially written by human annotators (Section 3). The knowledge database is extracted from several wiki-like websites about Minecraft, which contains both unstructured text data and structured infoboxes (Section 4). A preliminary examination of the data suggests that the knowledge database can be used for answering most relevant questions, but it may not as simple as a keyword search (Section 5). Also, we show examples that the QA corpus can be used for clustering verbs and linking them to the actions in the game, suggesting its usefulness in encountering language variations in the QA task (Section 6). We conclude the paper in Section 7.

2 Minecraft

Minecraft (Figure 1) is a sandbox video game, which means that the player is free to choose the actions he wants to execute, and the order of these actions. However, as all video games, the number of possible actions is limited. The main occupation of the player in Minecraft is to survive in a world populated by monsters, by finding resources (e.g. mining minerals, growing plants, etc.), to create structures, items and weapons (i.e. crafting them with the collected resources by follow-

Mobs (monsters)		Items (objects)		Blocks/Structures (world elements)	
Subcategories :		Subcategories :		Subcategories :	
Hostile mobs	Attack the player.	Materials	Used to craft other items.	Minerals	Found in/under the ground.
Neutral mobs	Do not attack first.	Tools	Used to collect, built..	Plants	Found over the ground.
Passive mobs	Do not attack the player.	Weapons	Used to fight mobs.	Mechanisms/Utility blocks	Blocks that can be used.
Possible action of mobs :		Possible actions of items :		Possible actions of blocks	
Spawn, Appear		Be obtained, Be craft, Be made, Be bought		Be obtained, Be harvested	
Fight, Attack		Be dropped by a mob		Be placed	
Use an item		Be used		Be used	
Die, Disappear, Be killed		Be modified, Be enchanted		Be modified	
Drop items/ressources/experience		Be sold		Be thrown	
Be tamed, Be ridden, Be fed		Be thrown, Be put		Be destroyed, Be mined	
Possible actions of the player on mobs :		Possible actions of the player on items :		Possible actions of the player on blocks :	
Spawn, Make appear		Obtain, Craft, Make		Obtain, Harvest	
Fight, Attack		Use		Place	
Use an item on		Modify, Enchant		Use	
Beat, Kill		Sell		Throw	
Tame, Ride, Feed		Throw, Put		Destroy, Mine	
Other possible actions of the player (not related to an other entity) :					
Walk, Run, Swim, Travel		Restore Health, Sleep		Eat	Spawn
Earn experience		Lose health		Get hungry	Die

Table 1: Entity types and actions in Minecraft

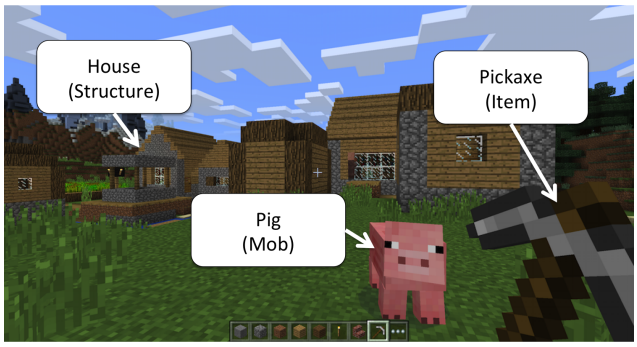

Figure 1: A snapshot of Minecraft

ing recipes) and beating monsters using crafted weapons and items to protect the created structures and earn experience and new items, in order to continue to develop. The different entities and objects of the game can be divided in 3 groups interacting with the player: Mobs (monsters), Items and Blocks/Structures. The player can interact with these entities, or make independent actions. These interactions are summarized in the Table 1. The liberty of the player guarantees a large number of possible questions to ask about the game, but the game follows nonetheless a logic that can be learned by a QA system to increase its ability to understand the meaning of the questions. In other words, our goal is to build a QA system that can translate a question asked by players to a series of queries about the entities and actions in the game. In addition, since Minecraft is a popular game, we expect that we can find abundant data from the Web.

3 Constitution of the QA corpus

The corpus of questions and answers is based on posts extracted from quiz websites. 754 questions have been collected from different websites[1]. Then, we manually selected 100 relevant questions. For our purpose, it is important that the questions deal with facts inside the game (see Table 2), or at least closely related to the game (e.g. questions about the creator of the game, the programming language used, etc.).

We selected both factoid and non-factoid questions, and tried to include the most possible variety of concept of the game in the selected questions. For each selected question, we wrote about nine questions with the same meaning but asked differently, or with a close or related meaning (Table 3).

This way, we obtain a corpus that can be used for encountering language variations in the QA

[1] www.quizlet.com
www.allthetests.com
www.gamefaqs.com

Factoid questions:
What Item should I use to tame a Wolf?
Are Spiders Hostile?
Non-factoid questions:
What is the best way to spawn the two different types of Golem?
Is it interesting to kill the Ender Dragon?

Table 2: Examples of questions

0	*Where do you find a Mushroom?*
1	*How do you obtain a Mushroom?*
2	*How do I get a Mushroom?*
3	*Where can I get a Mushroom?*
4	*Where can I obtain a Mushroom?*
5	*What is a way to get a Mushroom?*
6	*How to get Mushrooms?*
7	*Where do I find Mushrooms?*
8	*Where are Mushrooms located?*
9	*Where can Mushrooms be found?*

Table 3: Examples of questions

Breaking time[note 1]	
Hand	25
Wooden	12.5
Stone	6.25
Iron	1.25
Diamond	0.95
Golden	2.1

<myTable type="wikitable">
{Breaking time: Hand; 25}
{Breaking time: Wooden; 12.5}
{Breaking time: Stone; 6.25}
{Breaking time: Iron; 1.25}
{Breaking time: Diamond; 0.95}
{Breaking time: Golden; 2.1}
< /myTable>

Figure 2: Table contents (Upper) are converted to tuples (Lower), with information such as headers("Breaking time") added to the tuples.

task. We also annotated the words of entities specific to the game, so this corpus can be used for training a Named Entity Recognition system adapted to our context. We obtain a corpus of 1684 questions, among which 928 has been written on the basis of 100 relevant questions.

4 Constitution of the knowledge database

The knowledge database is extracted from three different websites[2]. These websites are constituted of pages describing an object, an entity or a fact of the game. Similar to Wikipedia, the webpages can be divided into two parts, namely the structured data such as infoboxes and tables, and unstructured data such as natural language texts. We preserve the structures of infoboxes and tables in our extraction. As a result, we obtain a database composed of 1222 text files, organized in 51 folders and sub-folders to regroup related objects (see Table 4 for details).

Each file describes an entity/object of the game or a fact about the game, in which we can find text and links as unstructured data and/or the content of tables written in the form of tuples (see Figure 2).

One technical issue here is that the way of arranging information differs among tables. We use hand-written rules to recognize headers, categories and values in the tables, and rearrange the information in the extracted tuples.

5 Can the questions be answered by the database

From the questions that have been extracted from quizzes websites, we can distinguish 3 types of questions. Some questions are not relevant, because they deal with some facts extern to the game itself, or because they contain a mistake:

What is the name of the famous yellow duck who plays Minecraft on YouTube? (YouTube is extern to the game.)

[2] www.minecraft.gamepedia.com
www.minecraft.wikia.com
www.minecraftguides.org

Folders		Number of Files	Description
DB-Gamepedia (from minecraft.gamepedia.com)	Blocks	154	Environment's blocks
	Entity	72	Mobs (Monsters)
	Items	161	Objects used by the player
	Others	183	Gameplay, History, etc.
	=> Total	530	Different files
DB-Wikia (from minecraft.wikia.com)	Blocks	200	
	Items	167	
	Main	23	Important objects/entities
	Mobs	56	
	=> Total	392	Different files
DB-Guides (from minecraftguides.org)	Blocks	101	Minerals, Plants, etc.
	Brewing	34	Recipes of potions
	Building	7	
	Farming	7	
	Items	77	Food, Tools, Weapons, etc.
	Main	10	Summaries of sub-folders
	Mini-Games	25	
	Mobs	27	
	Tutorials	30	
	=> Total	300	Different files
=> Total		1222	

Table 4: A summary of the knowledge database

What are the 5 types of wood? (There are actually 6 types of wood.)

Some questions can be 'easily' answered with the knowledge database. This is the case when the answer is clearly written in the database (for example a numeric value in a table). The questions that can be 'easily' answered are often factoid questions:

How many hearts does a Giant have? (The answer, 50 hearts, is written in the infobox of the Giant.)

This kind of question can be answered simply by locating the place where the answer is written. However, some questions can be answered only by computing the answer using crossed information. This is the case of non-factoid questions, which can be considered as 'difficult' to answer:

What is the best strategy for finding diamonds? (To answer this question, the system has to find all the different ways to find diamonds and evaluate the efficiency of each method. This evaluation is challenging because the criteria for a good strategy are not stated in the question.)

In our data, the not relevant questions are rare (about 2%), whereas the non-factoid questions are quite common (about 20%), which provides a good motivation for a QA system to handle complicated questions.

6 Using the QA corpus to cluster verbs

We use the Stanford POS-tagger to extract nouns, verbs, adjectives and adverbs in the questions of the QA corpus. This results in a list of 465 different words, with 235 nouns, 126 verbs, 81 adjectives and 23 adverbs. Then, by calculating the co-occurrence of words in questions, we extracted the context of each word in a vector, and compared the nouns, verbs, adjectives and adverbs side-by-

rank	verb	distance to the verb 'craft'
1	make	0.0929
2	use	0.0930
3	get	0.0935
4	create	0.0939
5	obtain	0.0964
	...	
	pay	0.3302
	contain	0.3594

Table 5: Cluster of the verb 'craft'

side to extract clusters. An example of the results is given in Table 5.

We see that by only using the questions, the system is able to link the meaning of several verbs into one type of action in the game. Understanding this kind of link is essential for the system to be efficient when the player ask a question with its own words, and not those used by the knowledge database.

7 Conclusion and discussion

We have described a Question-Answering corpus and a knowledge database related to the video game Minecraft. Our goal is to build a system that can answer questions using the logic specific to the game. A lot of research has been done on the answering of real world questions using Freebase [**?**, **?**] or Wikipedia [**?**]. Datasets for these tasks usually favour systems that do simple queries of facts on the knowledge database [**?**]. As the complexity of the questions increase, answering the questions usually becomes considerably difficult [**?**], due to the vast complexity of the real world. There are efforts to restrict the domain of the task and pursue some advanced reasoning. The Todai Robot Project [**?**] restricts the domain to university entrance exam questions. Other research includes solving algebra word problems [**?**] and instructing robots [**?**]. As a complement to these previous works, we believe the using of an open world video game as the domain has several merits. Firstly, the logic in a video game is simpler than the real world, which means that it can be handled readily. Therefore, this domain may provide a convenient testbed for integrating logical inference techniques into NLP systems, such as the logical inference using dependency-based compositional semantics [**?**]. Secondly, despite the rather simple rules, open world video games provide enough liberty for players, and their popularity attracts people to ask many questions about them, including creative and fun questions that can be solved only by completely understanding the rules and logically combining them. Therefore, we expect the domain to be interesting and challenging as well.

References

[1] J. Berant and P. Liang. Semantic parsing via paraphrasing. In *ACL*, 2014.

[2] A. Fujita, A. Kameda, A. Kawazoe, and Y. Miyao. Overview of todai robot project and evaluation framework of its nlp-based problem solving. In *LREC*, 2014.

[3] N. Kushman, Y. Artzi, L. Zettlemoyer, and R. Barzilay. Learning to automatically solve algebra word problems. In *ACL*, 2014.

[4] D. K. Misra, K. Tao, P. Liang, and A. Saxena. Environment-driven lexicon induction for high-level instructions. In *ACL*, 2015.

[5] P. Pasupat and P. Liang. Compositional semantic parsing on semi-structured tables. In *ACL*, 2015.

[6] R. Tian, Y. Miyao, and T. Matsuzaki. Logical inference on dependency-based compositional semantics. In *ACL*, 2014.

[7] X. Yao. Lean question answering over freebase from scratch. In *NAACL*, 2015.