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 developing Question-Answering in a new context, the open world of a video game. We chose a popular game called 'Minecraft', and created a QA corpus with a knowledge database related to this game and the ontology of a meaning representation that will be used to structure this database. 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 by using inference on these game-specific logic rules. The QA corpus is partially composed of online quiz 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, that will be structured using the meaning representation we defined, and already structured data such as infoboxes. A preliminary examination of the data shows that players are asking creative questions about the game, and that the QA corpus can be used for clustering verbs and linking them to predefined actions in the game.

Keywords: question-answering, knowledge acquisition, meaning representation

1. Introduction

This paper presents a corpus and a knowledge database aiming at creating a Question-Answering system specific to a new context, the open world of a video game. Unlike many QA systems that are designed to answer real world questions (Berant and Liang, 2014; Yao, 2015), 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, whose openness provides a great liberty for players, which guarantees a large number of possible questions to ask about the game, and yet the presence of 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 contain both unstructured text data and structured tables and infoboxes (Section 4.). A preliminary examination of the data suggests that the knowledge database can be used for answering most of the relevant questions, but it may not be as simple as a keyword search (Section 5.), and may require to structure the unstructured part of the knowledge database, so that inference can be done by the QA system on several pieces of information in order to answer complex questions (Section 6.). We also show that the QA corpus can be used for clustering words and for linking them to the entities and actions of the game, in order to face the language variations that are encountered in the QA tasks (Section 8.). We conclude the paper in Section 9.. Our resource is publicly released at: `github.com/CorentinDumont/QA_Minecraft`

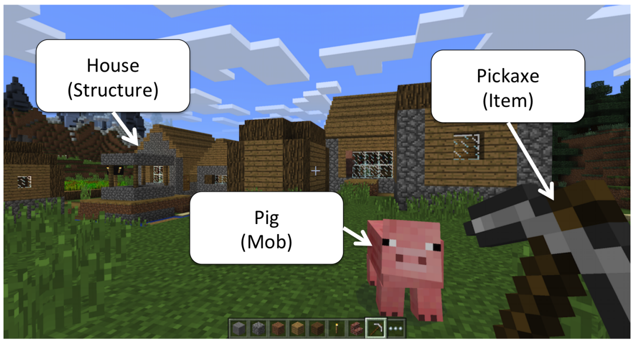


Figure 1: A snapshot of Minecraft

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 following 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 2 groups interacting with the player: Mobs (monsters), and Objects, which are divided in Items (for example used to fight) and Blocks (used to build Structures). The player can interact with these entities, or make independent actions. These actions and 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

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

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

players to a series of queries about the entities and actions in the game, and find a relevant answer in the knowledge database among all the information about these entities and actions. 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 as much language variety (vocabulary and grammar) as possible in the reference to the concepts of the game. 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 obtained a corpus that can be used for handling language variations in the QA task. We also annotated the words designating entities specific to the game, so this corpus can be used for training a Named Entity Recognition system adapted to our context. We obtained a corpus of 1684 questions, among which 928 has been written on the basis of 100 relevant questions.

4. Constitution of the knowledge database

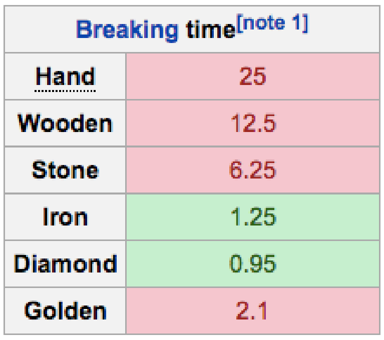
The knowledge database is extracted from three different websites[2]. These websites are constituted of pages describing an entity or a concept of the game. Similar to those from Wikipedia, the webpages can be divided into two parts, namely the structured data such as infoboxes and tables, and unstructured 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).

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

[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 files		
DB-Wikia (from minecraft.wikia.com)	Blocks	200	
	Items	167	
	Main	23	Important objects/entities
	Mobs	56	
	Total: 392 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 files		
Total: 1222 files			

Table 4: A summary of the knowledge database



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

{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}

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

The content of tables has been saved 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 external 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 external to the game.)

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 in the database. 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 non-relevant questions are rare (about

Event classes	Description
change_value	One of the value (health, defense,...) of one entity changes
become	An entity becomes an other entity
mine	The player or a mob destroys a block
craft	The player crafts an object
use	The player or a mob uses an object
spawn	A mob appears
encounter	The player encounters an entity
kill	The player or a mob kills a mob
move	An entity moves
face	The player or a mob looks at something
get	The player gets an object
mix	Several objects become one
collide	Several entities collide
fight	The player or a mob fights an other mob
place	The player puts a block
despawn	An entity disappears
drop	A mob drops an item
ride	The player or a mob rides an other mob

Figure 4: Events with associated predicates

2%), whereas the non-factoid questions are quite common (about 20%), which provides a good motivation for a QA system to handle complicated questions.
However, answering non-factoid questions implies reasoning, i.e. logic inference on different pieces of information contained in the knowledge database, which can only be done on a structured database. This is why we defined the ontology of a meaning representation which is used to structure the natural sentences of the database to their logical form. The challenge is to balance the complexity of the meaning representation, as it must be simple to take advantage of the simple logic of the game, but must be expressive enough so that it can be used to structure all the useful information.

6. Ontology of a meaning representation

Our ontology uses two types of classes to represent the concepts in Minecraft. These are Minecraft Entities (Figure 3) and Events (Figure 4).
Minecraft Entities include all Objects (e.g. Blocks, Structures, Items, *etc.*) and Mobs that the player can interact with. We manually listed a total of 444 Minecraft Entities arranged in a class hierarchy (e.g. the Minecraft Entity "*Stone*" is a subclass of "*Natural Block*"), as shown in Figure 3. The list is constructed by checking named entities appeared in our QA corpus and document set, and is supposed to have high coverage. We have regrouped some concepts that are usually used by players as different ones but are actually the same objects in the game. For example, both "*chicken*" and "*chick*" are represented by the same Minecraft Entity class *Chicken*, but with variations in the attribute *size* set as *adult* and *baby* respectively. The regrouping is done because these entities have similar interactions with the player and other entities.

Gold is a type of ore.
$gold(x_1), ore(x_2)$ $type\_of(f_1), subject(f_1, x_1), type\_of(f_1, x_2)$ $Holds(f_1)$

Figure 5: A fact (the first type) representing a subsumption between classes.

Each Event is represented by linking an action (Event class) with some participant entities in the event. There are only 18 possible actions, each one associated with a set of particular predicates indicating the participants. Figure 4 shows a complete list of the actions that are used in our meaning representation. We use a Davidsonian style representation for events; for example, an entity x_1 dropping an item x_2 is represented as

$$drop(e), dropper(e, x_1), dropped(e, x_2).$$

We make the list of Event classes by considering possible operations by the player and checking questions asked in our QA corpus. We tried to minimized the number of Event classes by regrouping some events that can be expressed as the same actions linked to different Minecraft Entities. For example, both the actions *sleep* and *eat* are regrouped into the event *use*, because "*sleeping*" and "*eating*" are equivalent to "*using*" the Minecraft Entities *bed* and *food*, respectively.
A piece of information is represented by a Fact in our ontology. There are three types of Facts. The first type describes Minecraft Entities in the game but does not involve actions (Figure 5), such as the existence of an entity or subsumptions between classes. We defined 10 such facts.
The second type regards properties of a single event, such as possibility and frequency (Figure 6). We defined 3 such facts.
The third type represents relations between multiple events, such as condition and effect (Figure 7). We defined 3 such facts.

As an example of possible logical inference, the following piece of information is written in our document set:

If a chicken dies while on fire, it drops cooked chicken instead of raw chicken.

Then, assuming the system has the following common sense knowledge (axiom):

If something is dropped, the player gets it.

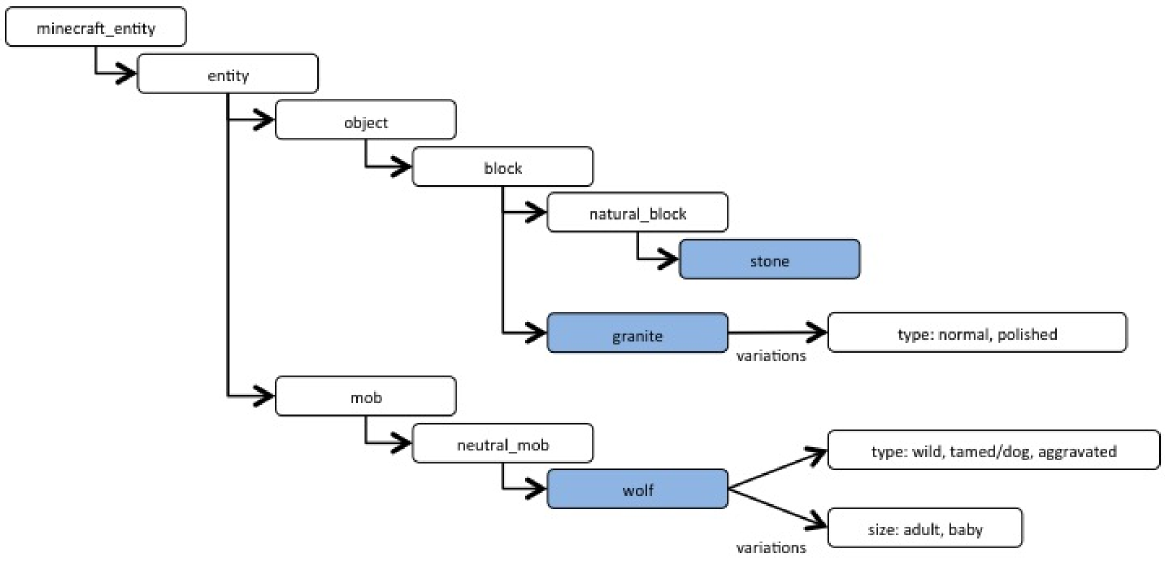


Figure 3: Examples of Minecraft Entities (stone, granite and wolf)

Stone can be mined with a pickaxe.
$stone(x_1), mine(e_1), pickaxe(x_2)$ $mined(e_1, x_1), instrument(e_1, x_2)$ $event\_property(f_1)$ $has\_property(f1, e1, possible)$ $Holds(f_1)$

Bats usually spawn in caves.
$bat(x_1), spawn(e_1), cavern(x_2),$ $spawned(e_1, x_1), in\_place(e_1, x_2),$ $event\_property(f_1)$ $has\_property(f1, e1, frequent)$ $Holds(f_1)$

Figure 6: Facts (the second type) on properties of single events.

If a chicken dies while on fire, it drops cooked chiken instead of raw chicken.
$chicken(x_1), kill(e_1), fire(x_2), drop(e_2),$ $chicken\_food[cooked](x_3),$ $chicken\_food[raw](x_4),$ $killed(e_1, x_1), instrument(e_1, x_2),$ $dropper(e_2, x_1), dropped(e_2, x_3),$ $!dropped(e_2, x_4)$ $has\_effect(f_1)$ $effect(f1, e1, e2)$ $Holds(f_1)$

Figure 7: A fact (the third type) about conditions between events.

we can deduce the following:

> *If a chicken is killed by fire, the player gets cooked chicken.*

A system equipped with logical inference ability can thus answer a question such as

> *How to obtain cooked chicken?*

by the inference process described above and responds:

> *You should kill a chicken with fire.*

7. Can our meaning representation express enough information?

A question will be answerable if the answer is present in the knowledge database (the question is theoretically answerable), and if all the pieces of information that are needed to answer can be represented with our meaning representation. We expect the contents extracted from the 3 complete websites to have a high coverage, so most of the relevant questions are theoretically answerable. The quality of our QA system will then directly depend on the quantity of relevant pieces of information (that can be used to answer players' questions) that can be represented with the meaning representation that we defined.

We manually answered 10 questions of our training corpus by locating all the related pieces of information (35 different pieces of information in the database were relevant to answer the questions), and by evaluating the difficulty to answer them using our meaning representation. The preliminary analysis of the results allows us to draw some conclusions.

Firstly, not all the pieces of information that are related to a question are necessary to construct a satisfying answer. In our annotations, only 40% of the pieces of information were necessary. The main reason for that is that most of these pieces of information bring details that are not compulsory to construct a relevant answer.

Secondly, in its entirety, the knowledge database is not highly redundant (we have chosen websites with complementary information). In our annotations, more than 60% of the pieces of information were written only once. However, if we only consider the pieces of information

that were absolutely necessary to construct relevant answers, about 70% of them were redundant and sometimes appeared in both text and tables (already structured data), what should be an advantage in the structuring process of the knowledge database. Nonetheless, we will have to care about the recall of the translation process not to loose essential information.
Thirdly, for 30% of the annotated questions, information contained in natural sentences and information contained in tables had to be combined in order to construct a relevant answer. So our QA system will have to be able to combine different types of information together.
Fourthly, the meaning representation can only represent the information in a single sentence, and even by solving the co-reference problem with the Stanford core NLP tool, we sometimes loose or misunderstand important information by removing the context of the sentence. In particular there is a risk to generalize some facts that are only true in a specific context that is not specified in the sentence. As the pages of the websites we used for the knowledge database are divided into sections, we believe that this problem can be at least partially solved by using the names of these sections to solve some further co-references and lacks of context.
Eventually, 30% of the annotated questions were not answerable with the knowledge database alone because some pieces of information, necessary for the inference process that lead to an answer, were not written in the database. These pieces of information are axioms, that are obvious for human readers but that have to be taught to the QA system. For example, the question

How do I obtain an Enchantment Table?

can not be answered with the knowledge database, unless the QA system is told that

When the player craft an object, the object is obtained by the player.

Indeed, if the crafting recipe of the Enchantment Table is written in the database, it is not explicitly written that the player will obtain this object by following this recipe. These axioms will probably have to be taught manually, but fortunately, the simple logic of Minecraft should restrict a lot the number of such axioms. We estimate that about 20 axioms could be sufficient to solve this problem.

8. Using the QA corpus to cluster words

We took advantage of the simple logic of Minecraft to simplify the meaning representation, in particular by using a very restricted list of possible entities and events. It implies that our system has to be able to handle the diversity of the natural language, as many different words can be used in both the knowledge database and the questions to describe the same concept in the ontology we defined. This is why we must cluster words.

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 the co-occurrence bag of words method, we calculated the vectors of all the words present in the questions, and compared the nouns, verbs, adjectives and adverbs side-by-side with the Euclidean distance to extract clusters. An example of the results is given in Table 5.

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’

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.

9. 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 (Berant and Liang, 2014; Yao, 2015) or Wikipedia (Pasupat and Liang, 2015). Datasets for these tasks usually favour systems that do simple queries of facts on the knowledge database (Yao, 2015). As the complexity of the questions increases, answering the questions usually becomes considerably difficult (Pasupat and Liang, 2015), 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 (Fujita et al., 2014) restricts the domain to university entrance exam questions. Other research includes solving algebra word problems (Kushman et al., 2014) and instructing robots (Misra et al., 2015). 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 (Tian et al., 2014). 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.

10. Bibliographical References

Berant, J. and Liang, P. (2014). Semantic parsing via paraphrasing. In *ACL*.

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

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

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

Pasupat, P. and Liang, P. (2015). Compositional semantic parsing on semi-structured tables. In *ACL*.

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

Yao, X. (2015). Lean question answering over freebase from scratch. In *NAACL*.