An Ontology for Question-Answering on Minecraft

Corentin Dumont, Ran Tian, Kentaro Inui
Graduate School of Information Sciences, Tohoku University

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

1 Introduction

The ability to answer complicated questions by advanced logical inference is one of the ultimate goal of NLP. In this paper, we present a new corpus for QA on an open world video game called Minecraft. The game has a specific logic, which is more restricted than the real world and supposedly easier to be handled by an artificial intelligence. Yet, the openness of the game guarantees a large number of possible questions to be asked, and the popularity has attracted players to creating plenty of resource on the web. We have (i) created a QA corpus with 1684 questions, (ii) extracted and organized a knowledge base of 1222 documents about entities in the game, and (iii) manually constructed an ontology scheme for the QA task. We believe the data can provide a testbed for combining NLP with advanced logical inference techniques.

2 Minecraft

Minecraft is a sandbox video game, which means that the player is free to choose the actions he wants to execute at any time. 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.

Though the number of possible actions is limited, there are a large variety of structures, items and mobs in the game and complicated interactions among them. This can elicit various questions of interest from players, such as "*How to tame a Wolf?*" and "*Are Spiders hostile?*". Our goal is to build a QA system that can automatically answer the questions. For this purpose, we need (i) a QA corpus to analyze the needs of players and evaluate the system, and (ii) a set of documents about the game that the system can extract information from. Fortunately, the popularity of the game has made such data available on the web, as we explain in Section 3.

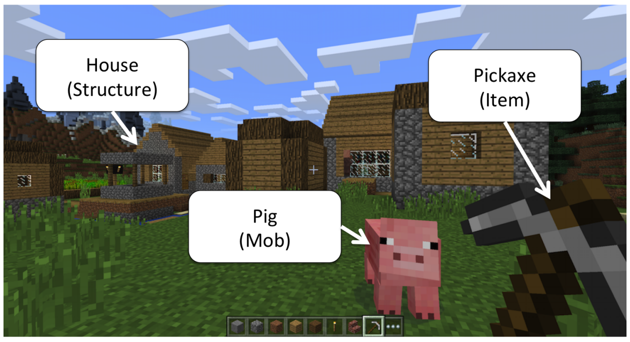


Figure 1: A snapshot of Minecraft

3 Corpus

We have extracted a corpus of 754 questions about Minecraft from three online quiz sites[1], in which 544 questions come with an answer. From these questions, we selected the 100 most relevant ones (the ones that cover the major types of questions we expect our system to answer), and we manually wrote about 10 similar questions for each of the selected questions, adding a total of 930 to the corpus. As a result, we obtained a corpus of 1684 questions, among which 1474 have an

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

answer.

As a preliminary analysis on the corpus of questions, we divide the questions into factoid and non-factoid questions. Factoid questions usually ask about yes/no facts, numbers, or entities regarding a specific piece of information, such as "*Are Spiders hostile?*" and "*How many hearts does a Giant have?*". These questions can be "easy" to answer assuming that we can locate the piece of information precisely. On the other hand, there are also non-factoid questions such as "*How to spawn the two different types of Golem?*" and "*Is it interesting to kill the Ender Dragon?*". Answering such questions usually need to combine different pieces of information and make inference. In our corpus, we find that non-factoid questions are quite common (about 20%), which motivates the development of advanced semantic processing techniques for this task.

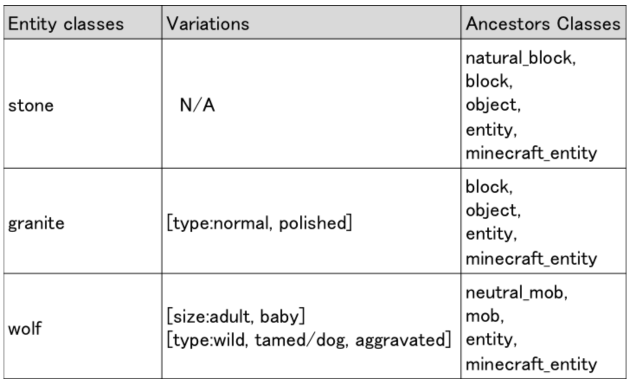
In parallel, we have extracted a set of documents about 'Minecraft', from three Wikipedia-like websites[2]. Totally there are 1222 documents, each one describing a concept of the game, such as "*Abandoned Mine Shaft*" (an item that can be used by players), "*Bat*" (a mob/monster), and "*Stone*" (a block that can be used to build structures). Then, in order to organize the information in these documents such that inference can be made for answering complicated questions, we manually constructed an ontology as described in Section 4.

4 Ontology

Our ontology uses two types of classes to represent the concepts in Minecraft. These are Minecraft Entities (Figure 2) and Events (Figure 3).

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*"). The list is constructed by checking named entities appeared in our QA corpus and document set, and is sup-

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



Entity classes	Variations	Ancestors Classes
stone	N/A	natural_block, block, object, entity, minecraft_entity
granite	[type:normal, polished]	block, object, entity, minecraft_entity
wolf	[size:adult, baby] [type:wild, tamed/dog, aggravated]	neutral_mob, mob, entity, minecraft_entity

Figure 2: Examples of Minecraft Entities

posed 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 (Figure 2). The regrouping is done because these entities have similar interactions with the player and other entities.

Each Event is represented by linking an action (Event class) with some participant entities in the event. There are only 18 possible actions, each associated with a set of particular predicates indicating the participants. Figure 3 shows a complete list of all actions with their associated predicates. 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 event linked to different Minecraft Entities. For example, both the actions *sleep* and *eat* are regrouped to the event *use*, because they 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 not involving actions (Figure 4), such as the existence of an entity or subsumptions between classes. There are 15 predicates related to

Event classes	Related predicates

change_value	target(change_value,entity)
	increase(change_value,gameplay_quantity)
	decrease(change_value,gameplay_quantity)
	new_value(change_value,gameplay_quantity)

become	old(become,minecraft_entity)
	new(become,minecraft_entity)

mine	miner(mine,mob)
	mined(mine,block|structure)
	instrument(mine,tool|weapon)

craft	crafted(craft,block|item)
	ingredient(craft,block|item)

use	user(use,mob)
	used(use,block|item)
	target(use,entity)

spawn	spawned(spawn,entity)
	manner(move|spawn,{close,far})

encounter	encountered(encounter,entity)

kill	killer(kill,mob)
	killed(kill,mob)
	instrument(kill,tool|weapon)

move	subject(move,entity)
	destination(move,entity|place)
	manner(move|spawn,{close,far})

face	subject(face,entity)
	target(face,entity)

get	obtained(get,block|item)
mix	ingredient(mix,block|item)
collide	collider(collide,entity)

fight	fighter(fight,mob)
	target(fight,mob)
	instrument(fight,tool|weapon)

place	placed(place,block)
despawn	despawned(despawn,entity)

drop	dropper(drop,entity)
	dropped(drop,entity)

ride	ride_on(ride,mob)

*	at_time(*,time)
	in_place(*,place|structure|block)
	in_version(*,game_version)

Figure 3: Events with associated predicates

this type of facts. The second type regards properties of a single event, such as possibility and frequency (Figure 5). We have defined 6 predicates related to this type. The third type represents relations between multiple events, such as condition and effect (Figure 6). We defined 2 predicates to represent such relations.

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 chiken instead of raw chicken.*

Gold is a type of ore.
$gold(x_1), ore(x_2)$ $definition(f_1), subject(f_1, x_1), group(f_1, x_2)$ $assert(f_1)$

Figure 4: A fact (the first type) representing a subsumption between classes.

Stone can be mined with a pickaxe.
$stone(x_1), mine(e_1), pickaxe(x_2)$ $mined(e_1, x_1), instrument(e_1, x_2)$ $possible(e_1)$

Bats usually spawn in caves.
$spawn(e_1), bat(x_1), cavern(x_2),$ $spawned(e_1, x_1), in\_place(e_1, x_2),$ $frequent(e_1)$

Figure 5: Facts (the second type) on properties of single events.

Then, assuming the system has the following common sense knowledge:

> *If something is dropped, the player gets it.*

So 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.*

5 Future work

In the future, we plan to use the ontology to annotate the meaning of a small amount of documents, so as to check its consistency and estimate the coverage. Semi-supervised methods can be used for linking natural language relations to structural databases, such as by matrix factorization [7]. Thus, the annotation data can be used for evaluation of such methods. After the ontology is populated, we plan to combine the knowledge with logical inference.

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

Figure 6: A fact (the third type) about conditions between events.

6 Discussion

We have described a dataset for Question-Answering on the video game Minecraft. A lot of research has been devoted to answering real world questions using structural knowledge, such as Freebase [1, 9] or Wikipedia [6]. It is considerably difficult to answer complicated questions in these tasks because of the complexity of real world. Therefore, we expect the problem to become simpler by restricting to a virtual world.

There are efforts to restrict the domain and pursue advanced reasoning in QA. The Todai Robot Project [2] and especially the world history ontology [3] is such an effort. Other research includes solving algebra word problems [4] and instructing robots [5]. We believe our data as a complement to these previous works can bring interesting and challenging topics to the field. [8]

Acknowledgment This work was supported by CREST (Establishment of Knowledge-Intensive Structural NLP and Construction of Knowledge Infrastructure), JST.

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] A. Kawazoe, Y. Miyao, T. Matsuzaki, H. Yokono, and N. Arai. World history ontology for reasoning truth/falsehood of sentences: Event classification to fill in the gaps between knowledge resources and natural language texts. In *LENLS*, 2014.

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

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

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

[7] S. Riedel, L. Yao, A. McCallum, and B. M. Marlin. Relation extraction with matrix factorization and universal schemas. In *NAACL*, 2013.

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

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