The University of Michigan Robot Language Benchmark #1

Peter Lindes 2 May 2017

Overview

This document describes a set of files that are intended for use in evaluating and comparing different language comprehension systems that produce grounded meaning representations within an autonomous robotic system. This material is submitted as supplementary material for a paper titled *Grounding Language for Interactive Task Learning* submitted to the Language Grounding for Robotics workshop at ACL 2017. Refer to the main paper for more context on how the sentences are to be used and about how language grounding works.

File Name	Description
UMRLB-1_v0.1.pdf	This document describing the files in the benchmark and their meanings.
Grounding_v0.1.pdf	The paper submitted for which all these files are the supplemental material. This gives a more theoretical explanation of how the grounding works, with references.
Sentences.txt	The corpus of sentences that can be used to evaluate a comprehension system.
World.json	A definition of a particular snapshot of the world perceived by a robot that can be used to ground linguistic expressions.
Ontology.json	A simple ontology defining properties of perceived objects and robot actions.
GoldStandard.json	A file giving the gold-standard meaning for each sentence in the corpus, along with other metadata.

Table 1: Files Included in this Benchmark

The benchmark includes a corpus of sentences to be understood, a world model describing a snapshot of the agent's view of the world, and an ontology describing object properties derived from the agent's perception and actions the agent knows how to perform. In addition it includes a set of gold-standard output messages, one for each sentence in the input corpus.

Sentences

The input corpus consists of 200 sentences contained in the file Sentences.txt, which is a simple text file with one sentence per line. These sentences were chosen manually to provide ways of communicating important messages to a robot and to show a reasonable range of linguistic issues. The fall into the following general categories:

- Simple declarative sentences that describe objects

- Commands to tell the robot to manipulate objects or navigate in a simple building environment
- Questions, both yes/no questions and WH-questions
- Sentences using the pronouns *it*, *this*, *that*, and *you*
- Sentences with prepositional phrases and relative clauses that must be attached properly to other parts of the sentence
- Conditional sentences providing a condition that must be satisfied before taking an action
- Sentences defining an *until* condition to terminate an action

Most of the sentences contain referring expressions that refer to objects in the environment. It is expected that a system to understand them will ground the meanings of these linguistic expressions to objects described in the world model. In the following paragraphs we give examples of the various kinds of sentences.

Declarative sentences

(1)

a. `The medium block is green.`
b. `This is a big triangle.`
c. `The soda is in the kitchen.`

(2) Object manipulation commands

a. `Pick up the stapler.`
b. `Put it on this.`
c. `Move the green rectangle to the left of the large green rectangle to the pantry.`
d. `Pick a green block that is on the stove.`

The sentences in (1) and (2) illustrate a number of cases of nouns, adjectives, determiners, and verbs that must be grounded as individual words and then combined into grounded phrases and sentences. The pronouns *this* and *it* must also be dealt with. Prepositions introduce prepositional phrases, and in cases like (2c) the attachment of these phrases must be resolved in order to ground the actions and referring expressions properly. Sentence (2d)[1] shows a case of a relative clause modifying a noun phrase, and that sometimes these interact with prepositional phrases.

The sentences in (3) show several simple commands for navigation in the indoor environment. These include various primitive motion verbs such as *go*, *drive*, *turn*, and *face*. In most cases these verbs take a location or a direction which indicates how far to go or how far to turn. Sentence (3c) differs somewhat from (3a) in that the terminating location is an obstacle to be reached rather than a room to be entered. In (3d) the termination condition is a distance to be travelled rather than a location.

(3) Indoor navigation commands

a. `Go to the kitchen.`
b. `Turn left.`
c. `Drive to the wall.`

[1] Sentence (2a) illustrates the verb *pick up*. Sentence (2d) shows a case where the verb *pick* alone is used as a synonym for *pick up*. This usage of *pick* appears in a number of the sentences in this corpus.

d. `Drive forward one meter.`
e. `Face east.`

Sometimes sentences like those in (4) are needed to tell the robot to perform a certain command only when some condition is met. Sentences (4a) and (4b) could refer to a condition in the immediate situation and the command could be executed immediately. All of the sentences in (4), however, can usually be interpreted to set up a condition which will not be fulfilled until some point in the future as the robot is driving around. This means that the conditions will be grounded only as unseen or unknown objects until such time as the condition is met.

(4) Conditional commands
a. `If you see the soda then pick it up.`
b. `If the green box is large then go forward.`
c. `If the lights in an empty room are lit then turn off the lights.`
d. `If you see some trash then throw it away.`

Another kind of deferred grounding is illustrated in the sentences in (5) involving *until* clauses.

(5) Sentences with *until* clauses
a. `Go until there is a doorway.`
b. `Go forward until you see an intersection.`
c. `Follow the right wall until there is a doorway.`
d. `Drive until you sense a wall.`

Many of the tasks a robot must perform involve learning new verbs that may require the agent to learn several steps and make a plan to reach some goal. Since these tasks are learned using interaction with a human instructor, several interactions with the instructor may be needed to learn the whole task. For our purposes here we will simply describe some sentences that are used to teach the agent some of these tasks.

(6) Sentences for task instruction
a. `Throw away the trash.`
b. `The goal is that the trash is in the garbage.`
c. `Deliver the box to the office.`
d. `The goal is that the box is in the office.`

When an agent gets sentence (6a) and it does not know what *throw away* means, it tells the instructor that it needs help. Sentence (6b) defines the goal of the task, and with this information the agent can make a plan and, remembering what it was told in (6a) carry out the new task. A similar process happens with (6c) and (6d). Once such a task has been performed once, the agent can remember what it has learned for the next time it sees a similar command.

Another part of interaction with a robot involves asking it questions about things it knows about. In (7) we see examples of both yes/no questions and WH questions that the agent should be able to understand and answer.

(7) Questions
a. `Is the large sphere green?`
b. `Is the small orange triangle behind the green sphere?`

c. `What is inside the pantry?`
d. `Where is the red triangle?`
e. `What is this?`
f. `What color is the large sphere?`

The sentences we have shown in these examples constitute less than 13% of the entire corpus. The entire set of sentences and their gold standard meanings are in the data files.

The World Model

The world model is a data structure meant to represent the robot's current perceived view of the world at a certain fixed point in time. It contains a list of movable objects which the robot can manipulate, an indicator of which of these objects is currently being pointed to by the instructor, a list of locations in which objects can be placed or that can be navigated to, and a list of binary relations between objects and other objects or locations. The particular state of the world model used for this benchmark is contained in the World.json file.

Table 2 lists the objects in this current world. Each object has a unique "id" by which it can be identified in messages produced by the comprehension system. Each object also has a "handle" that gives a unique human-readable identifiers that is helpful for debugging.

Id	Handle	Category	Color	Shape	Size
obj-001	self				
obj-002	large-orange-triangle1	block	orange1	triangle1	large1
obj-003	small-red-triangle1	block	red1	triangle1	small1
obj-004	medium-green-block1	block	green1	rectangle1	medium1
obj-005	large-green-block1	block	green1	rectangle1	large1
obj-006	large-green-sphere1	block	green1	sphere1	large1
obj-007	small-orange-triangle1	block	orange1	triangle1	small1
obj-008	medium-purple-triangle1	block	purple1	triangle1	medium1
obj-009	small-green-box1	block	green1	box1	small1
obj-010	conference-room-lights1	fixture		lights1	

Table 2: Objects in the World Model

Each object is defined by its category, color, shape, and size properties. The values of these properties are unique strings similar to handles in that they are human-readable. However, these names of property values are not intended to be meaningful to the language comprehension system and could be any arbitrary strings. The comprehension system must contain knowledge to map words in the input language to these property values so that it can then ground a referring expression to a particular object or set of objects that fit the linguistic description. The world model does not include specific positions of the objects, but it does contain both a set of locations where objects can be located. The following shows what the objects look like in the World.json file.

(8) Some objects in the benchmark world model

a. {"id":"obj-003","handle":"small-red-triangle1",
"item_type":"object",
"properties":{"category":"block","color":"red1",
"shape":"triangle1","size":"small1"}}

b. {"id":"obj-004","handle":"medium-green-block1",
"item_type":"object",
"properties":{"category":"block","color":"green1",
"shape":"rectangle1","size":"medium1"}}

c. {"id":"obj-005","handle":"large-green-block1",
"item_type":"object",
"properties":{"category":"block","color":"green1",
"shape":"rectangle1","size":"large1"}}

d. {"id":"obj-006","handle":"large-green-sphere1",
"item_type":"object",
"properties":{"category":"block","color":"green1",
"shape":"sphere1","size":"large1"}}

e. {"id":"obj-009","handle":"small-green-box1",
"item_type":"object",
"properties":{"category":"block","color":"green1",
"shape":"box1","size":"small1"}}

f. {"id":"obj-010","handle":"conference-room-lights1",
"item_type":"object",
"properties":{"category":"fixture","shape":"lights1"}}

g. {"id":"obj-011","handle":"1",
"item_type":"object",
"properties":{"category":"location","name":"pantry"}}

h. {"id":"obj-018","handle":"office1",
"item_type":"object",
"properties":{"category":"location","name":"office"}}

The notation used here requires some explanation. The basic format used a language called JSON, an industry standard. This is used for the benchmark rather that internal Soar notation so that it can be easily processed by others using other systems. Each object has both an *id* and a *handle*. The ids are essential for the internal workings of the system, and are unique over all the data structures of interest. The handles are part of the world model used for testing only. They are also unique, but are also easily human readable to facilitate debugging. A real running robot will not have these handles.

The locations in the world model are listed in Table 3 and in the last two items in (8) above. Locations do not have color, shape, or size, but each does has a "name" that can be used in an input sentence to identify that location.

Id	Handle	Category	Name
obj-011	1	location	pantry
obj-012	2	location	stove
obj-013	3	location	garbage
obj-014	4	location	sink
obj-015	6	location	table
obj-016	7	location	waypoint

obj-017	8	location	conference1
obj-018	office1	location	office

Table 3: Locations in the World Model

In addition to objects, the world model contains some information about spatial relations between objects, like these:

(9) Some relations in the benchmark world model

a.
```
{"id":"rel-001","handle":"in1","item_type":"relation",
 "instances":[
   {"arg1":"obj-003","arg2":"obj-011"},
   {"arg1":"obj-010","arg2":"obj-017"}]}
```

b.
```
{"id":"rel-002","handle":"left-of1","item_type":"relation",
 "instances":[
   {"arg1":"obj-004","arg2":"obj-005"}]}
```

c.
```
{"id":"rel-004","handle":"on1","item_type":"relation",
 "instances":[
   {"arg1":"obj-004","arg2":"obj-012"},
   {"arg1":"obj-002","arg2":"obj-003"}]}
```

Each relation is represented as a class whose handle suggests its English representation and which contains one of more instances. In general these relation classes are used to ground prepositions. Each instance contains two arguments, the two objects related by this instance of the relation. From these examples we can see that the object `obj-003` is in an instance of relation `rel-001` with `obj-011`. This means that the object with handle `small-red-triangle1` in (8a) is *in the pantry*. Similarly, from `rel-002` we see that `medium-green-block1` is *to the left of* `large-green-block1`.

There are two additional pieces of knowledge in the world model, as shown in here:

(10) Additional items in the benchmark world model

a.
```
"pointed-to":"obj-002"
```

b.
```
"robot":{"id":"robot-001","handle":"rosie",
         "arm":{"id":"arm-001","action":"wait"}}
```

The deictic pronouns *this* and *that* require that the system know what object is currently being pointed to by the instructor, and the `pointed-to` item gives this information. In this case the indicated object is `obj-002`, which happens to be `large-orange-triangle1`. This feature provides a crude form of what is often called "joint attention." The `robot` item gives the agent a representation of itself, which is used to ground the pronoun *you* when spoken to the robot by the instructor.

The Ontology

This section defines properties and actions, grouping them in classes. Table 4 shows the set of properties. The English column does not appear in the Ontology.json file, it is just shown here for clarification. Again it is important to remember that the handles shown look a lot like the corresponding English words, but that has no significance for the grounding of language. The language system itself must provide a

mapping from linguistic items to these handles. A few items are plurals, and those have their "multiple" property set.

Type	Handle	Multiple	English
category	block		block
	location		location
	object		object
	block	true	blocks
	location	true	locations
	object	true	objects
color	red1		red
	orange1		orange
	yellow1		yellow
	green1		green
	blue1		blue
	purple1		purple
	white1		white
	black1		black
	brown1		brown
	gray1		gray
shape	triangle1		triangle
	arch1		arch
	square1		square
	l-block1		l-block
	t-block1		t-block
	sphere1		sphere
	chicken1		chicken
	rectangle1		rectangle
	soda1		soda
	box1		box
	steak1		steak
	lights1		lights
	package1		package
	papers1		papers
	kinect1		Kinect
	trash1		trash
	line1		line
	triangle1	true	triangles
	stapler1		stapler
size	small1		small
	medium1		medium
	large1		large

Table 4: Properties and their values

In addition to the properties described in Table 4, Ontology.json has definitions of a large set of actions. Each action has the id that it is referred to by in the gold standard messages, and a handle that can be used to retrieve the agent's internal knowledge of this action. As before, the handles *NEVER* correspond to real English words, even though they may appear to.

Gold-Standard Output Messages

Once the comprehension process gets to the point where it has recognized a complete sentence as a single construction, the meaning of this top-level construction is interpreted to produce a grounded, actionable message for the agent to act on. Every message has a message-type field, plus other arguments depending on this type and the rest of the meaning of the sentence. Here we will show some examples of the messages generated for various types of sentences.

The examples will be shown in the form of items from the GoldStandard.json file. Each of these gold standard items has a unique identifier for the sentence, the text of the sentence, and the gold standard message expected to be produced. These gold standard messages have been derived by running each sentence through a language comprehension system like the one described in the main paper, collecting the message produced, and then hand editing, where needed, to get an intended meaning that a robot could understand.

Declarative sentences

Declarative sentences produce messages of type `object-description`. This message has an argument called `object` to indicate the object being described and an argument called `property` showing the property to be assigned to the object.

(12) a.
```
{"id":"S-003",
   "text":"The medium block is green.",
   "message":{"type":"object-description",
              "object":"obj-008",
              "property":{"class":"color","value":"green1"}}}
```

b.
```
{"id":"S-004",
   "text":"The red triangle is clear.",
   "message":{"type":"object-description",
              "object":"obj-003",
              "property":{"class":"property","value":"clear"}}}
```

Action commands

All action commands produce messages of type `command`. Each command has an `action` argument, most have an `object` argument, and others have varying arguments. Following are some typical examples.

(13) a.
```
{"id":"S-024",
   "text":"Pick up the green sphere.",
   "message":{"type":"command","action":"action-006",
              "object":"obj-006"}}
```

b.
```
{"id":"S-029",
   "text":"Put the green sphere in the pantry.",
   "message":{"type":"command","action":"action-007",
              "object":"obj-006",
              "relation":{"id":"in1","arg2":"pantry"}}}
```

c.
```
{"id":"S-067",
   "text":"Move the orange triangle on the red triangle
           to the stove.",
   "message":{"type":"command","action":"action-047",
              "object":"obj-002",
              "relation":{"id":"on1","arg2":"stove"}}}
```

d.
```
{"id":"S-069",
   "text":"Store the large green sphere on the red triangle.",
   "message":{"type":"command","action":"action-045",
              "object":"obj-006",
              "relation":{"id":"on1","arg2":"obj-003"}}},
```

An interesting thing to note in the sentence in (13c) is that the entire referring expression *the orange triangle on the red triangle* is grounded down to a reference to a single object.

Conditional commands

The command *'If you see the soda then pick it up.'* illustrates a conditional command which has a message type of `conditional` and `if-clause`.

(14)
```
{"id":"S-121",
   "text":"If you see the soda then pick it up.",
   "message":{"type":"conditional","action":"action-006",
              "object":"obj-036",
              "object-desc":{"id":"obj-036",
                             "category":"object","shape":"soda1"},
              "if-clause":{"test":{
                             "action":"action-008",
                             "agent":"rosie",
                             "object":"obj-036"}}}}
```

Notice that in the message the main action and object are those from the *then* clause, but since this is a `conditional` message they will only be executed if the condition defined in the `if-clause` is met.

The gold-standard message in (14) illustrates another important point. Often commands to a robot mention objects that are not in view and that are unknown to the agent at that time. These objects will not be in the world model, nor will they have handles. The `object-desc` field is added to give a description of what the object should look like. The agent will attempt to match this description to visible objects as it is moving along, and when it sees a soda it will pick it up. Notice that the pronoun *it* is grounded to the new object to be found that was just created.

Sentences with until clauses

The sentences in (5) describe actions to be carried out until some condition is true. In (15) we show the gold-standard messages for sentences (5a) and (5d). Each looks like many of the commands in (13), except instead of an `object` argument there is an `until-clause` argument.

(15) a.
```
{"id":"S-126",
   "text":"Go until there is a doorway.",
   "message":{"type":"command","action":"action-002",
              "until-clause":{"exists":"door"}}}
```

b.
```
{"id":"S-129",
   "text":"Drive until you sense a wall.",
   "message":{"type":"command","action":"action-002",
              "object-desc":{"id":"obj-021",
                             "category":"object",
                             "spatial-shape":"wall1"},
              "until-clause":{"test":{
                                 "action":"action-008",
                                 "agent":"rosie",
                                 "object":"obj-021"}}}}
```

In (15a) the `until-clause` simply gives a description of an unknown object by its handle. The one in (15b) is more complicated, describing a `sense1` action by the agent with another unknown object as its `object`.

Task instruction sentences

Consider the two related sentences in (6a) and (6b). The first commands an unknown *throw away* action on an object, and the second responds for a request for help by the agent by stating the goal of this action. The messages we should expect for these sentences are shown in (16).

(16) a.
```
{"id":"S-047",
   "text":"Throw away the trash.",
   "message":{"type":"command","action":"action-021",
              "object":"trash1",
              "modifier":"away1"}}
```

b.
```
{"id":"S-149",
   "text":"The goal is that the trash is in the garbage.",
   "message":{"type":"object-description",
              "object":"goal",
              "subclause":{
                "object-relation":{
                  "object":"trash1",
                  "relation":{"id":"in1","arg2":"garbage"}}}}}
```

The message in (16a) looks much like the commands we have seen before, with the known action `throw1`, but also with a `modifier`. The agent must ask for clarification. The response is the message in (16b), an `object-description` for the goal of the modified action. In this case, rather than assigning a property to the object as we have seen before in (12), the goal object is associated with a `subclause` which defines a relation between the `object` of the original command and a `location`. This gives the agent enough information to carry out the action.

Questions

A number of questions are listed in (7), and (17) shows the gold-standard messages we expect to be generated for some of them.

(17) a.
```
{"id":"S-177",
   "text":"Is the large sphere green?",
   "message":{"type":"object-question",
              "object":"obj-006",
              "property":{"class":"color","value":"green1"}}}
```

b.
```
{"id":"S-179",
   "text":"Is the small orange triangle
           behind the green sphere?",
   "message":{"type":"object-question",
              "object":"obj-007",
              "relation":{"id":"behind1","arg2":"obj-006"}}}
```

c.
```
{"id":"S-186",
   "text":"What is inside the pantry?",
   "message":{"type":"what-is-question",
              "relation":{"id":"in1","arg2":"pantry"}}}
```

d.
```
{"id":"S-187",
   "text":"Where is the red triangle?",
   "message":{"type":"where-is-question",
              "object":"obj-003"}}
```

e.
```
{"id":"S-190",
   "text":"What is this?",
   "message":{"type":"what-is-question",
              "object":"obj-002"}}
```

f.
```
{"id":"S-191",
   "text":"What color is the large sphere?",
   "message":{"type":"predicate-question",
              "object":"obj-006"}}
```

In (17) there are a number of different message types, but the rest of the messages is pretty simple, just defining an object and some property or relation to test for.

Word definition sentences

Several sentences in the corpus are used to define new words to the system. These are represented by messages of type `word-definition`, each of which has a `word` argument and a `property` argument. The purpose of these messages is for the agent to build this new word into its vocabulary, having it represent the class and value found in the `property` argument. There are some examples in (18).

(18) a.

```
{"id":"S-195",
   "text":"Mauve is color.",
   "message":{"type":"word-definition",
              "word":"mauve",
              "property":{"class":"color","value":"mauve3"}}}
```


b.

```
{"id":"S-197",
   "text":"Octagon is a shape.",
   "message":{"type":"word-definition",
              "word":"octagon",
              "property":{"class":"shape","value":"octagon1"}}}
```


c.

```
{"id":"S-198",
   "text":"Study is a location.",
   "message":{"type":"word-definition",
              "word":"study",
              "property":{"class":"location","value":"study2"}}}
```


d.

```
{"id":"S-199",
   "text":"In-a-row is a relation.",
   "message":{"type":"word-definition",
              "word":"in-a-row",
              "property":{"class":"relation","value":"in-a-row20"}}}
```


e.

```
{"id":"S-200",
   "text":"Transport is an action.",
   "message":{"type":"word-definition",
              "word":"transport",
              "property":{"class":"action","value":"transport19"}}}
```


Evaluation Techniques

The idea of this benchmark is that the data provided could be used with any language comprehension system capable of dealing with the class of sentences in this corpus. To use the benchmark, such a system must be fitted with some adapters to give it the information provided in the World.json and Ontology.json files, and then translate its output into the form given in the GoldStandard.json file. Once this is done, this output file can be compared with the GoldStandard.json file to evaluate the correctness of this system. It is our intention to provide a program for doing this comparison in a future release.

This benchmark is part of a system in active development, so there are likely to be some problems with it. We welcome comments on its accuracy and usability.