SemEval-2017 Task 11: End-User Development using Natural Language

Juliano Efson Sales, Siegfried Handschuh, André Freitas
Department of Computer Science and Mathematics
University of Passau, Germany
{juliano-sales, siegfried.handschuh, andre.freitas}@uni-passau.de

Abstract

This task proposes a challenge to support the interaction between users and applications, micro-services and software APIs using natural language. It aims to support the evaluation and evolution of the discussions surrounding the application natural language processing techniques within the context of end-user natural language programming, under scenarios of high lexical and semantic heterogeneity.

1 Introduction

The specific syntax of traditional programming languages and the user effort associated with finding, understanding and integrating multiple interfaces within a software development task, defines the intrinsic complexity of programming. Despite the widespread demand for automating actions within a digital environment, even the basic software development tasks require previous (usually extensive) software development expertise. Domain experts processing data, analysts automating recurrent tasks, or a businessman testing an idea on the web depend on the mediation of programmers to materialise their demands, independently of the simplicity of the task to be addressed and on the availability of existing services and libraries.

Recent advances in natural language processing bring the opportunity of improving the interaction between users and software artefacts, supporting users to program tasks using natural language-based communication. This ability to match users' actions intents and information needs to formal actions within an *application programming interface* (API), using the semantics of natural language as the mediation layer between both, can drastically impact the accessibility of software development. Despite the fact that some software development tasks with stricter requirements will always depend on the precise semantic definition of programming languages, there is a vast spectrum of applications with softer formalisation requirements. This subset of applications can be defined and built with the help of natural language descriptions.

This SemEval task aims to develop the state-of-the-art discussions and techniques concerning the semantic interpretation of natural language commands and user action intents, bridging the semantic gap between users and software artefacts. The practical relevance of the challenge lies in the fact that addressing this task supports improving the accessibility of programming (meaning a systematic specification of computational operations) to a large spectrum of users which have the demand for increasing automation within some specific tasks. Moreover, with the growing availability of software artefacts, such as APIs and services, there is a higher demand to support the discoverability of these resources, i.e. devising principled semantic interpretation approaches to semantically match interface descriptions with the intent from users.

The proposed task also intersects with demands from the field of robotics, as part of the human-robot interaction area, which depends on a systematic ability to address user commands that lie beyond navigational tasks.

From the point-of-view of computational linguistics, this challenge aims to catalyse the discussions in the following dimensions:

- Semantic parsing of natural language commands;
- Semantic representation of software interfaces;

- Statistical and ontology-based semantic matching techniques;
- Compositional models for natural language command interpretation (NLCI);
- Machine learning models for NLCI;
- API/Service composition and associated planning techniques;
- Linguistic aspects of user action intents.

2 Commands & Programming in Natural Language

The use of natural language to instruct robots and computational systems, in general, is an active research area since the 70's and 80's (Maas and Suppes, 1985; Guida and Tasso, 1982) (and within references). Initiatives vary over a large spectrum of application domains including operating system's functions (Manaris and Dominick, 1993), web services choreography (Englmeier et al., 2006), mobile programming by voice (Amos Azaria, 2016), domain-specific natural programming languages (Pane and Myers, 2006), industrial robots (Stenmark and Nugues, 2013) and home care assistants.

The variability of domains translates into a wide number of research communities comprising different foci and being expressed by distinct terms such as *natural language interfaces*, *end-user development*, *natural programming*, *programming by example* and *trigger-action development*. Some of these terms embrace wide domains, also including non-verbal (visual) approaches.

2.1 Semantic Parsing & Matching

The interpretation of natural language commands is typically associated with the task of parsing the natural language input to an internal representation of the target system. This internal representation is usually associated with a n-ary predicate-argument structure which represents the interface for an action within the system. The identification of which action the command refers to and its potential parameters are at the centre of this task.

Taking as an example the natural language command:

> *Please convert US$ 475 to the Japanese currency and send this value to John Smith by SMS.*

We can conceptualise the challenges involved in the command interpretation process in three dimensions: *command chunking*, *term type identification* and *semantic matching*. The chunking dimension comprises the identification of terms and segments in the original sentence that can potentially map to the system actions and parameters. The example command embodies two actions: *converting currency* and *sending SMS*. For the first action, the command interpreter needs to identify the currencies involved in the transaction and the financial amount (term type identification).

Other semantic interpretation processes might be involved. In the case of the second action, besides identifying *John Smith* as the message's receiver, the interpreter also needs to resolve the co-reference of *this value* to the currency conversion result and instantiate it as a parameter in the content of the message. This first level of interpretation of the command would generate an output such as:

SEQUENCY {
ACTION: [convert currency]
PARAMS: [US$ 475] - [(to) Japanese currency]

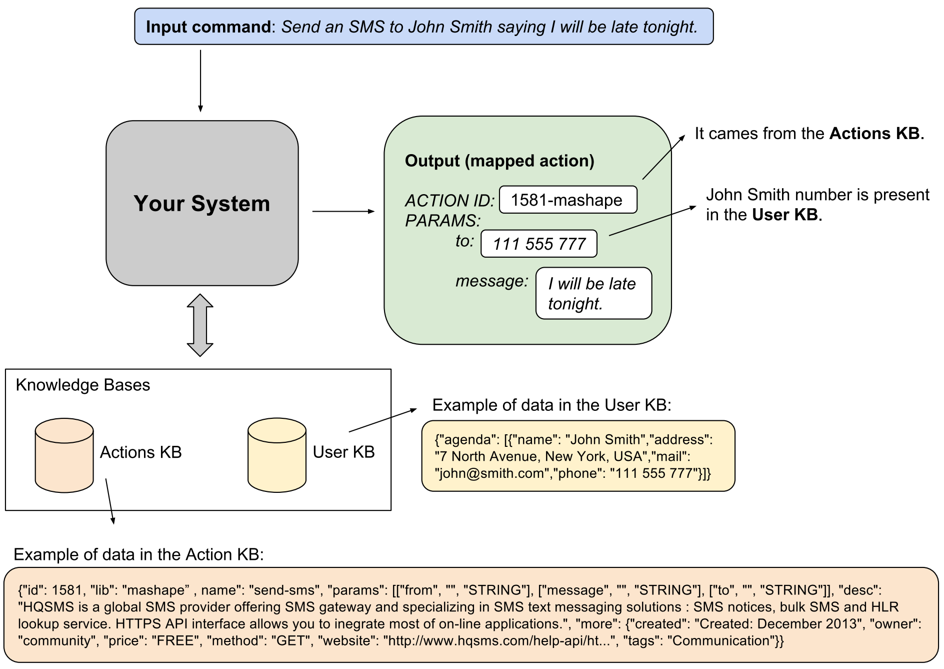
ACTION: [send sms]
PARAMS: [this value] - [(to) John Smith]
}

The *matching process* corresponds to the mapping between terms from the user vocabulary to the terms used in the internal representation of the system (the API). In the given example, the system should find an action that can convert currencies and another that can send SMS messages.

In the example, depending on the parametrisation of the command interface, the value *[US$ 475]* needs to be split into two parameters, and these parts, mapped to the internal vocabulary of the system (*US$* need to be interpreted as *USD* while *Japanese currency* needs to be translated to *JPY*. For the second action, similarly, *John Smith* will be used to retrieve a phone number from a user personal data source.

The final execution command is the result of the matching processing, as shown below:

ACTION ENDPOINT: *[action id]*
PARAMS:


Input command: *Send an SMS to John Smith saying I will be late tonight.*
Your System
Output (mapped action)
ACTION ID: 1581-mashape
PARAMS:
to: *111 555 777*
message: *I will be late tonight.*
It cames from the **Actions KB**.
John Smith number is present in the **User KB**.
Knowledge Bases
Actions KB
User KB
Example of data in the User KB:
Example of data in the Action KB:

{"agenda": [{"name": "John Smith","address": "7 North Avenue, New York, USA","mail": "john@smith.com","phone": "111 555 777"}]}

{"id": 1581, "lib": "mashape" , name": "send-sms", "params": [["from", "", "STRING"], ["message", "", "STRING"], ["to", "", "STRING"]], "desc": "HQSMS is a global SMS provider offering SMS gateway and specializing in SMS text messaging solutions : SMS notices, bulk SMS and HLR lookup service. HTTPS API interface allows you to inegrate most of on-line applications.", "more": {"created": "Created: December 2013", "owner": "community", "price": "FREE", "method": "GET", "website": "http://www.hqsms.com/help-api/ht...", "tags": "Communication"}}

Figure 1: An overview of the task.

from: *"USD"*
to: *"JPY"*
from_amount: *475*

The task can be addressed using different semantic interpretation abstractions: shallow parsing, lambda-calculus-based semantic parsing (Artzi et al., 2014), compositional-distributional models (Freitas and Curry, 2014; Freitas, 2015), information retrieval approaches (Sales et al., 2016). Additionally, pre-processing techniques such as clausal disembedding (Niklaus et al., 2016) and co-reference resolution are central components within the task.

While approaches and test collections emphasising the shallow parsing aspect of the problem are more present in the literature (Section 3), others focusing on a semantic matching process involving a broader vocabulary gap (Furnas et al., 1987) are less prevalent. Part of this can be explained by the domain-specific nature of previous works (e.g. focus on spatial commands (Dukes, 2014)).

In contrast, this task emphasises the creation of a test collection targeting an open domain scenario, with a large-scale set of target actions, assessing the ability of command interpretation approaches to address a larger vocabulary gap. This scenario aims to instantiate a real use case for end-user natural language programming, since the action knowledge base used in the test collection maps to real-world APIs and so a semantic interpreter developed over this test collection can become a concrete end-user programming environment.

3 Similar Initiatives

Most of the applications related to the parsing of natural language commands are within the context of human-robot interaction. The *Human Robot Interaction Corpus* (HuRIC) describes a list of spoken commands between humans and robots. It is composed of three datasets which were developed under the context of three different events. They are annotated using Frame Semantics together with Holistic Spatial Semantics (Bastianelli et al., 2014).

Artzi et al. (2014) and Tellex et al. (2014) give a more focused contribution in the interpretation of spatial elements. In both cases, the vocabulary variability is more constrained. Similar vocabulary variability assumptions are present in Thomason et al. (2015) and Azaria et al. (2016).

In 2014, SemEval hosted a task related to the parsing of natural language spatial commands (Dukes, 2014), also targeting a robotics scenario. More specifically, the task proposed the parsing of commands to move a robot arm that moved objects within a spatial region.

The proposed task can be contrasted with these previous initiatives in the following dimensions: (*i*) more comprehensive knowledge base of actions, (*ii*) generic (open domain) user programming scenarios and (*iii*) exploration of the interaction between actions and user personal information (Section 4).

The work that has more similarity with this test collection is the problem defined by Quirk et al. (2015) under the *ifttt.com* platform, which targets the creation of an if-then receipt from a natural language description provided by the user. The first difference between the two tasks is the fact that, while the program structure is limited to if-then recipes in Quirk et al., other more complex structures are supported in this task. Secondly, in the case of Quirk et al., the task requires only the mapping of the actions that comprise the recipe, keeping aside the instantiation of the parameter values, while our proposed task emphasises both. Finally, the presence of these two characteristics introduces the challenge of mapping co-references and metonymy within the task.

4 Task Definition

The task comprises 210 scenarios which consist of a total of 438 natural language commands. Figures 1 and 2 depicts an overview of the task. A *scenario* is a set of sentences that defines a program in natural language. The excerpt below shows an example of a scenario:

> *"When a message from Enrico Hernandez arrives, get the necklace price; Convert it from Chilean Pesos to Euro; If it costs less than 100 EUR, send to him a message asking him to buy it; If not, write saying I am not interested."*

Associated with each scenario, there is a program which is composed of actions from the Action Knowledge Base (Action KB). In addition to the actions, the program also uses `If` and `Foreach` constructors, having the same semantics commonly expressed in programming languages to define the execution flow.

Like a programming language function, an action can have input parameters and return values. Table 1 shows examples of natural language commands describing scenarios.

Natural language scenario commands
If a receive a deposit from John Sanders in my bank account, send this message to him: "Hello John, thanks for your gift, I receive your deposit of some money to me, thanks a lot, buddy."
Send an email to Mark asking him for the picture we took in Munich. When I receive the answer, get the attached image and publish it on my Flickr account with the tags #munich, #germany, #my-love
Find "Bachianas N.5 of Villa-Lobos" on Youtube. Get the link and send to my mum.
List tweets containing #ChampionsLeague.
Find a picture of Darth Vader on Flickr. Post this text to my friends on Facebook with the picture of Darth Vader: May The Force Be With Us Next Friday!!!
Search on eBay for the iPhone 7 with the maximum price of 700 Euro and send the result list by e-mail to my wife.
Message Dr Brown by email, asking a suitable day for a meeting; When I receive the information, sent to my wife by email;
Search for a picture of Yoda. Attach that image in a Facebook post and write this: Friends, let's go to the cinema to the see Star Wars on Friday.
When I receive an email from Helena, get the attachment. Print it and write to Mr Sanders by Skype: Hi Mr Sanders, the document is in the printer.
If someone reports a problem in GitHub, send the problem title by Skype to John, if the project name is FinanceSystem. For all other systems, send a message to the Tech Manager.
If Manchester United wins, put Thriller of Michael Jackson in Spotify "celebrations" playlist and call me to say "we are the champions, my friends".
Open the door always when reached Central Park.
Get a quote about science. Get a photo of Paris. Attach the photo in an email, write the quote and send to maria@hotmail.com.
Get the translation of the hashtag #sqn. Convert it to a QR code and send to my Skype account.

Table 1: Examples of natural language commands describing scenarios.

The values of the parameters map to constants (e.g. integer numbers, string values) or to *tags*, which represent returning data from previously executed actions. There are two types of tags.

- <**returnX**> The return tag represents the content returned by the action X, where X is a sequential identifier.
- <**item**> The item tag is used only in the context of Foreach constructors. It represents an iterated item.

Both types of tags have some additional naming assumptions in order to simplify the syntax of the generated program. Examples of valid tags are:

- <**return1**> - meaning the data returned by the first action in the scenario.

- **<item>.url** - represent the attribute *url* of the item.

In addition to the scenarios, the test collection consists of:

- **Action KB**: The set of available API functions along with their respective documentation. The information describing the API functions does not follow a strict pattern. While some documentation has rich natural language descriptions or show usage examples, others are succinct and just contain the frame and parameter names. The same occurs concerning data format, data type and returning data. This test collection reflects the variability and heterogeneity that we find in real-world APIs.

- **User KB**: A personal user information dataset, which is necessary to make commands more natural by supporting co-reference resolution. It allows commands like “Call John”, once the system can identify the proper phone number from the User KB.

An example excerpt of the User KB is described below:

```
[
  {
    "name": "Maria Alice",
    "address": "Rua Central, 35,
                Rio de Janeiro,
                Brasil",
    "facebook": "malice",
    "group": "classmates",
    "mail": "maria@alice.com.br",
    "phone": "555 111 222",
    "skype": "maria.alice",
    "tags": "my wife",
    "twitter": "malice"
  },
  {
    "name": "John Sanders",
    "address": "7 North Avenue,
                New York, USA",
    "facebook": "jsanders",
    "group": null,
    "mail": "john@fam.com",
    "phone": "111 555 777",
    "skype": "johnjohn",
    "tags": null,
    "twitter": "jsanders"
  },
  ...
]
```

id	action_name(params*)
700000	make_a_payment(invoice)
600603	send_an_email(attachment_url)
700002	read_file_content(file)
700003	extract_content(info)
503679	convert(to)
700005	get_contacts(group)
600490	upload_public_photo_from_url(tags)
700006	search_image_on_Flickr(query)
700007	search_video_on_YouTube(query)
600431	create_a_link_post(link_url)
601733	post_a_tweet_with_image(image_url)
700008	tweets_from_search(search_for)
502328	directions(starting)
600352	new_item_from_search(search_terms)
600979	share_a_link(image_url)
700009	create_calendar_item(which_day?)
600761	print_document(document_url)
601535	post_message(user_name)
500797	convert-file(file)
700011	any_new_post_by_someone(user)
600591	any_new_issue(user)
601206	new_article_in_section(section)
600187	add_a_bitlink(url)
601732	post_a_tweet(tweet_text)
601888	picture_of_the_day(section)
600840	add_photo_to_album(album_name)
600408	new_final_score(team)
601684	new_story_from_section(which_section?)
600596	create_an_issue(body)
601791	air_quality_changed(device)
503062	search(depart-date)
502335	check(text)
600326	take_snapshots(which_camera?)
503155	get-top-definition(hashtag)

Table 2: Examples of action frames used in the scenarios.

The natural language scenarios, Action KB and User KB are all described using JSON as a serialisation format. The Action KB is composed of about 3800 micro-services from Mashape (mashape.com) and 1900 actions and triggers from the ifttt.com platform. APIs from Mashape and ifttt.com are public, and their instantiation for the challenge was approved by the platform owners.

Table 2 shows examples of action frames used in the dataset and Table 3 shows metrics about the scenarios, actions and the associated natural language commands, showing the natural language signature of the test collection.

metric	training	test	total
# of scenarios	179	31	210
avg # of sentences per scenario	2.72	2.35	2.66
# of actions	374	64	438
avg # of actions per scenario	2.08	2.06	2.08
avg # of params per action	1.59	1.22	1.52
# of conditionals in actions	53	14	67
# of co-references in actions	124	17	141
# of metonymy in actions	94	11	105

Table 3: Metrics about the scenarios, actions and the associated natural language command.

4.1 Annotation

The scenarios containing the natural language commands were created using high-level task descriptions. These high-level task descriptions were sent to a crowdsourcing platform (CrowdFlower), in which workers were requested to express in natural language the commands which entail the scenario descriptions. Motivated by those scenario descriptions, the users proposed a set of commands which addresses the specification.

The excerpt below shows an example of a scenario description:

> *You are arranging a meeting with some people in Andre's office. Adamantios is coming for that meeting, but he does not know how to drive in Passau. Additionally, you do not know where the office is.*

One possible output for that description is:

- Ask Andre for the address of his office;
- Make a map from the university to it;
- Send the map to Adamantios including driving directions.

For each scenario description, in average ten workers were invited to suggest the natural language commands. The crowdsourcing process was followed by a data curation process which discarded 70% of the commands due to low quality issues. The other part of the sample was reviewed to correct misspelling and adjusted to comply with the task requirements while preserving the original syntactic structure and vocabulary.

5 Analysis of The Task Complexity

The task aims to explore vocabulary and syntactic structure variation within the natural language commands. It also targets the orchestration of different natural language processing techniques, including syntactic parsing, semantic role labelling, fine-grained semantic approximation and co-reference resolution.

5.1 Semantic approximation

Different actions and parameters can be expressed using distinct lexicalizations (synonymy) and abstraction levels. For example:

> *"If someone reports a problem in GitHub, send* the problem*'s* headline *by Skype to John."*

In the examples, the action in the knowledge base is expressed as *"any new issue"*, while intended *"headline"* in the returned value is expressed as *"Issue Title"*. Given the context, it is expected the system to be able to identify the equivalence between the pairs of terms (*problem*, *issue*) and (*title*, *headline*).

5.2 Syntactic variation

Additionally, interpreters are expected to cope with syntactic variation.

> *"If Manchester United wins, call me."*

> *"Get ready to call me in the case of victory of Manchester United."*

5.3 Co-reference and metonymy resolution

The first type of resolution needed is the pronominal co-reference, where a pronoun refers to a constant which was previously mentioned within the context of the same scenario. The metonymy resolution consists of using the reference to an attribute or type to refer to a constant or to a different attribute of a constant. For example:

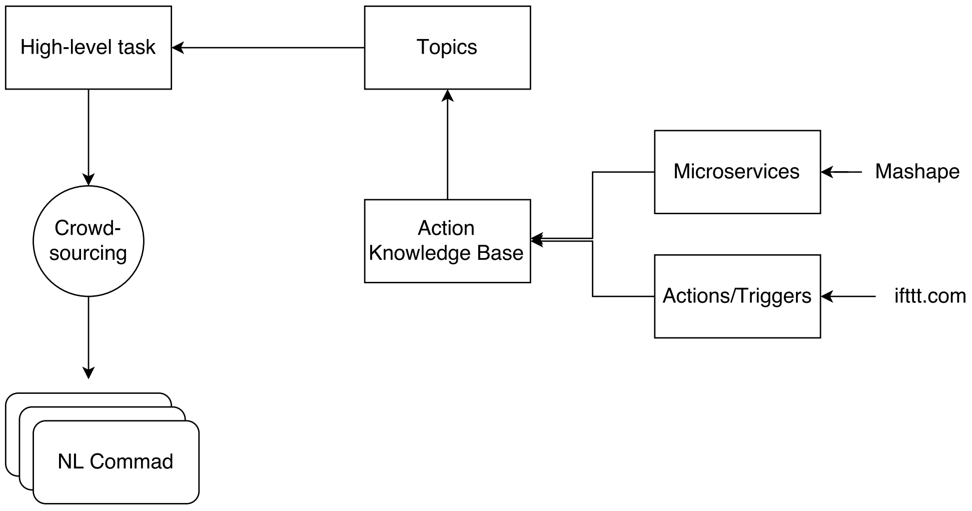


Figure 2: The scenario creation workflow.

> *"If an issue is created, send **its** content to the **Tech Manager**."*

This excerpt shows both cases. The bold **its** makes reference to **an issue**, while **Tech Manager** is a metonymy for the Tech Manager's email (sandra@andrade.com.br according to the user KB).

6 Evaluation

The final dataset contains commands and their associated mappings to the Action KB. Given a command in natural language, it is expected that the participating systems provide:

- The correct action;
- The correct mapping of text chunks in the natural commands to parameters;

The participating systems were evaluated considering four criteria:

1. Resolved individual actions ignoring parameter values;
2. Resolved individual actions considering parameter values;
3. Resolved scenarios ignoring parameter values;
4. Resolved scenarios considering parameter values.

Criteria 1 and 2 are quantified by using precision and recall, while 3 and 4 are quantified by the percentage of the total number of scenarios which were addressed.

Participating teams were allowed to use external linguistic resources and external tools such as taggers and parsers.

7 Participants and Results

Initially, nine teams demonstrated interest in the tasks, but only one participated in the challenge.

Kubis et al. (2017) proposed the EUDAMU system, which implements an action ranking model based on TF/IDF and a type matching system.

The EUDAMU system is composed of a pipeline divided into six steps. It starts by preprocessing the dataset using three tools (NLTK, Core-NLP and SyntaxNet). In the pre-processing step, natural language commands are tokenized and each token is enriched with its lemma, part-of-speech and named entity labels. Additionally, it also adds the constituent and dependency structures associated with the commands. The final pre-processing step annotates the commands with types which supports the system to resolve co-references between the actions and references from the User KB. The same procedure (with the exception of the last step) is applied for the Action KB.

The preprocessing phase is followed by the *Discourse Tagger*, which is responsible for individualising the command from the paragraph description of the scenario. The team implemented this component using a rule-based approach. The next step is *Action Ranker*, which applies a TF-IDF

Criterion	**Metric**	**Value**
Individual actions solved ignoring parameter values	precision	0.5490
	recall	0.7066
Individual actions solved considering parameter values	precision	0.0533
	recall	0.0533
Scenarios solved ignoring parameter values	accuracy	41.93%
Scenarios solved considering parameter values	accuracy	0%

Table 4: Results from Kubis et al.

model to rank the actions. The model was indexed using all textual content present in the Action KB, plus the actions which were mapped with in the training mappings file. The next step is the *Reference Matcher* that is designed to identify which output of a given action act as the parameters of a subsequent action. The next step is the *Parameter Matcher*. It infers parameter and value types which can serve as a support to the action matching process. Finally, based on the knowledge generated and stored in the previous steps, the rule-based *Statement Mapper* provides a list of up to 10 elements of possible matching action instances. Additional details of the proposed method can be found in the original paper (Kubis et al., 2017). Table 4 shows its results.

While the proposed solution has a high recall for the number of resolved actions, it fails mainly in providing the correct value for all the required parameters. Two types of linguistic settings showed to be more challenging:

- Description of commands split into two sentences. For example:

  "Get the price of the book The Intelligent Investor. If it costs less than 25 Euros, buy it."

  where "*25 Euros*" is the parameter value of the action defined in the first sentence.

- Capturing actions with more specific/fine-grained semantics. For example:

  "Once I have bet my running distance target of the week, set my current weight as 100 Kg in Fitbit."

  where the system ignored the temporal expression *"of the week"* and suggested the "*Daily step goal achieved*" instead of "*Weekly distance goal reached*" action. A second example of the same case is expressed in the command:

  "Suspend the execution of my Samsung washer."

  where the term "*Samsung*" was ignored when selecting actions.

8 Summary

In the Semeval 2017 Task 11 we developed a test collection to support the creation of semantic interpretation methods for end-user programming environments. The test collection focuses on the following features in comparison with existing approaches: (*i*) open domain, (*ii*) large syntactic and vocabulary variability, (*iii*) dependent of co-reference and metonymy resolution. Moreover, as the test collection uses APIs available on the open web, it can be used to build real end-user programming environments. While there is space for the improvement of the precision and recall on the identification of the command actions, the main challenge remains in the matching of the parameters between natural language commands and the API.

References

Jayant Krishnamurthy Tom M. Mitchel Amos Azaria. 2016. Instructable intelligent personal. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16.

Yoav Artzi, Dipanjan Das, and Slav Petrov. 2014. Learning compact lexicons for ccg semantic parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1273–1283. http://www.aclweb.org/anthology/D14-1134.

Emanuele Bastianelli, Giuseppe Castellucci, Danilo Croce, Luca Iocchi, Roberto Basili, and Daniele Nardi. 2014. Huric: a human robot interaction corpus. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn

Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), Reykjavik, Iceland.

Kais Dukes. 2014. Semeval-2014 task 6: Supervised semantic parsing of robotic spatial commands. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics and Dublin City University, Dublin, Ireland, pages 45–53. http://www.aclweb.org/anthology/S14-2006.

Kurt Englmeier, Javier Pereira, and Josiane Mothe. 2006. Choreography of web services based on natural language storybooks. In *Proceedings of the 8th International Conference on Electronic Commerce: The New e-Commerce: Innovations for Conquering Current Barriers, Obstacles and Limitations to Conducting Successful Business on the Internet*. ACM, New York, NY, USA, ICEC '06, pages 132–138. https://doi.org/10.1145/1151454.1151485.

André Freitas. 2015. Schema-agnostic queries over large-schema databases: a distributional semantics approach. PhD Thesis.

Andre Freitas and Edward Curry. 2014. Natural language queries over heterogeneous linked data graphs: A distributional-compositional semantics approach. In *Proceedings of the 19th International Conference on Intelligent User Interfaces*. ACM, New York, NY, USA, IUI '14, pages 279–288. https://doi.org/10.1145/2557500.2557534.

George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. 1987. The vocabulary problem in human-system communication. *Commun. ACM* 30(11):964–971. https://doi.org/10.1145/32206.32212.

Giovanni Guida and Carlo Tasso. 1982. Nli: a robust interface for natural language person-machine communication. *International Journal of Man-Machine Studies* 17(4):417 – 433. https://doi.org/http://dx.doi.org/10.1016/S0020-7373(82)80042-4.

Marek Kubis, Pawel Skorzewski, and Tomasz Zietkiewicz. 2017. EUDAMU at SemEval-2017 Task 11: Action ranking and type matching for end-user development. In *"Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)"*. Association for Computational Linguistics.

Robert Elton Maas and Patrick Suppes. 1985. Natural-language interface for an instructable robot. *International Journal of Man-Machine Studies* 22(2):215 – 240. https://doi.org/http://dx.doi.org/10.1016/S0020-7373(85)80071-7.

Bill Z. Manaris and Wayne D. Dominick. 1993. Nalige: a user interface management system for the development of natural language interfaces. *International Journal of Man-Machine Studies* 38(6):891 – 921. https://doi.org/http://dx.doi.org/10.1006/imms.1993.1042.

Christina Niklaus, Bernhard Bermeitinger, Siegfried Handschuh, and André Freitas. 2016. A sentence simplification system for improving relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 170–174. http://aclweb.org/anthology/C16-2036.

John F. Pane and Brad A. Myers. 2006. *"End User Development"*, "Springer Netherlands", "Dordrecht", chapter "More Natural Programming Languages and Environments", pages "31–50". https://doi.org/"10.1007/1-4020-5386-X_3".

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL-15)*. Beijing, China, pages 878–888. http://www.cs.utexas.edu/users/ai-lab/pub-view.php?PubID=127514.

Juliano Efson Sales, Andre Freitas, Brian Davis, and Siegfried Handschuh. 2016. A compositional-distributional semantic model for searching complex entity categories. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, Berlin, Germany, pages 199–208. http://anthology.aclweb.org/S16-2025.

M. Stenmark and P. Nugues. 2013. Natural language programming of industrial robots. In *Robotics (ISR), 2013 44th International Symposium on*. pages 1–5. https://doi.org/10.1109/ISR.2013.6695630.

Jesse Thomason, Shiqi Zhang, Raymond Mooney, and Peter Stone. 2015. Learning to interpret natural language commands through human-robot dialog. In *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press, IJCAI'15, pages 1923–1929. http://dl.acm.org/citation.cfm?id=2832415.2832516.

Matthew R. Walter, Sachithra Hemachandra, Bianca Homberg, Stefanie Tellex, and Seth Teller. 2014. A framework for learning semantic maps from grounded natural language descriptions. *International Journal of Robotics Research* 31(9):1167–1190.