A Method for Dataset Creation for Dialogue State Classification in Voice Control Systems for the Internet of Things

Ivan Shilin
shilinivan@corp.ifmo.ru

Liubov Kovriguina
lyukovriguina@corp.ifmo.ru

Dmitry Mouromtsev
mouromtsev@mail.ifmo.ru

Gerhard Wohlgenannt
gwohlg@corp.ifmo.ru

Roman Ivanitskiy
litemn@yandex.ru

ITMO University
Saint-Petersburg, Russian Federation

Abstract

In recent years, speech-based interaction became an important method of communication with devices in the Internet of Things (IoT). Voice control interfaces involve all the challenges and difficulties of natural language understanding and human-computer communication. In this paper, we present a methodology to create initial training data for voice-controlled devices which helps to design and track dialogue system states. Using crowdsourcing, in a first step we collect simple commands that users might give to devices. These commands are analyzed and manually classified into 50 user-system interaction scenarios. In a second step, we design a set of potential system states after processing the initial user commands, and crowd workers are asked to provide multi-turn dialogues between a user and the device, which simulate the processes of resolving a system state towards completion. The resulting dataset contains 320 commands and their classification into interaction scenarios for the first step, and 640 multi-turn dialogues for step two, generated given 12 potential system states. Finally, we present a baseline for automatic classification of utterance type and slot types in user commands, which is important for dialogue state detection. The proposed methodology allows collecting dialogues for IoT devices, which cover a variety of system states and interaction patterns. **Keywords:** *voice control systems for Internet of Things, slot type classification, command type classification, dataset for voice-controlled devices.*

1 Introduction and Related Work

Gartner, Inc. forecasts that 8.4 billion connected things will be in use worldwide in 2017, up 31 percent from 2016, and will reach 20.4 billion by 2020. The total spending on endpoints and services will reach almost $2 trillion in 2017[1]. The pool of devices and Internet of Things (IoT) architectures and protocols grows rapidly, however, this domain lacks communication instruments, regarding both interaction between devices and human-machine interaction and cooperation. Communication interfaces for IoT devices are typically specific to producers, thus, the development of conversational agents and voice control systems for IoT is in high demand due to the universal nature of natural language and speech communication [Portet et al., 2013, Aldrich, 2003].

There are two different paradigms for the design of dialogue systems and voice interfaces. On the one hand, in the *modular* approach, the system is assembled using various knowledge extraction algorithms and typically integrates a number of modules for natural language understanding, a dialogue manager, knowledge bases, rule bases, ontologies and other models [Jurafsky and Martin, 2017].

On the other hand, *end-to-end* systems are trained directly from conversational data. This approach requires no hand-crafted feature engineering and annotation, but large dialogue datasets for training [Lowe et al., 2017, Serban et al., 2015]. This requirements restricts end-to-end systems to certain domains and natural languages.

In the case of voice-controlled systems for IoT devices, the problems are aggravated. Here, often both specific training data and rule or knowledge bases are missing, and furthermore there is a large diversity of IoT architectures and protocols. Moreover, linguistic resources, describing devices, technologies, communication between users and IoT-systems are limited or do not exist. For example, the Google Speech Commands dataset[2] includes 65 000 short, one second long, commands (like "*left*!", "*stop*!"). Another source, the Mozilla Common Voice dataset[3], comprising 500 hours of speech from 20 000 different volunteers, was developed for keyword spotting tasks and web applications control. These and similar datasets are intended to train speech recognition systems.

In this paper, we propose a first step to address the problem of missing datasets for training end-to-end systems. We introduce a methodology for dataset creation using a combination of crowdsourcing and domain experts, and implement and evaluate the method by creating initial small-scale datasets of IoT dialogues for the Russian language. This dataset is primarily intended for the development of voice-controlled systems for IoT, such systems are within the emergent paradigm of artificial cognitive systems. Within the paper, we not only focus on dataset creation, but also on automatic morphosyntactic annotation and classification of user commands and slot type identification.

The methodology can be summarized as follows: The first step involves crowdsourcing to create first-turn data, which contain initial commands or requests of a user to an IoT device. Then, we analyzed the first-turn commands and elaborated 50 scenarios of user-system interaction. Independently, we designed an initial set of system states (12 states), to which the system can transition after understanding the command and collecting context knowledge.

[1] `https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-thin`

[2] `https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html`

[3] `https://voice.mozilla.org/ru/data`

In a second iteration of crowdsourcing, the users are asked to generate natural language responses to the respective system state, and to simulate a dialogue which resolves the situation to a successful final state. Moreover, we present and evaluate a baseline approach to classify the first-turn user commands into command types and and a set of 5 binary features ("slots"), which can be used to track the dialogue state in future work.

Applying this methodology, we collect 320 first-turn command items, and 640 multi-turn dialogues from the second crowdsourcing step. The proposed methodology allows for collecting dialogues of the necessary variety and representativeness for the described task, and it can be used to extend the dataset created in our experiments. To our best knowledge, there is no previous work which provides a comparable methodology or datasets. As a final remark about the dataset, it also contains automatically created annotations according to Universal Dependencies 2.0 standard (for morphology and syntax). The morphological and syntactical annotations were performed with the Russian language models for the Stanford CoreNLP library [Kovriguina et al., 2017]. The current demonstration files of the dataset with the corresponding annotations are published in the project repository[4].

The paper is structured as follows: Section 2 explains core aspects of this work, such as the methodology used for creating the dataset and to design the interaction scenarios and dialogue manager states. In Section 3 we present the experiments to automatically classify first-turn user commands, and we conclude with Section 4.

2 Dataset Creation for Dialogue State Tracking

In this section, we first discuss some fundamentals of dialogue managers and its influence on the data necessary for the development of conversational intelligence for IoT and then present the two steps of our methodology to create datasets for dialogue state identification in IoT systems.

2.1 Fundamentals of Dialogue State Tracking for IoT Devices

Typical modern architectures of conversational agents include dialogue managers as a core module. Dialogue managers can be implemented with the use of several approaches, such as finite-state automata, frame-based methods, the information state update (ISU) approach, the belief-desire-intention (BDI) model, reinforcement learning, etc. [Sungjin et al., 2016].

Dialogue state trackers aim to identify the current conversation state based on a user's input and the previous conversation history, so that the dialogue manager can choose the best next action. Typically, a slot-filling algorithm in the natural language understanding module identifies relevant objects in the user's utterance and tries to find an appropriate slot, which is then processed by the dialogue manager. Analysis of human conversations with smart environment has shown, that resolving coreference, especially its abstract type, is a much harder task, because users are inclined to use abstract lexis (*light, sound*) to denote the device (*lamp, audio system*, correspondingly). Therefore, a natural language understanding module needs algorithms, which can find empirical referents for abstract and new, previously unseen, concepts [Jia et al., 2017].

[4] `https://github.com/organizations/MANASLU8/VoiceIoT`

Conceptualizing conversational agents for the IoT domain as being emergent types of artificial cognitive architectures [Vernon, 2014, Profanter, 2012], we believe, that it is critical to involve context-sensitive knowledge obtained from devices, data storages, and device knowledge base into the space of dialogue states.

In a first step towards the goal of dialogue state tracking models for Internet of Things, it is necessary to isolate and analyze user request patterns for different devices. From this data, potential dialogue states can be designed. Furthermore, it is necessary to obtain users responses to given dialogue states, as well as to get natural language equivalents for the system state and the user interactions. In the following two-step dataset creation process, we present the methodology and first dataset creation experiments which tackle the described problems.

2.2 Step I: First-Turn Dataset Creation and Scenario Identification

In the first step of dataset creation, we collect initial commands by the device user. We employ a crowdsourcing strategy, and provide the user with a task description, which, in summary, asks crowd workers to choose an arbitrary smart device from any smart environment and provide 2-4 operation commands for the chosen device. For example, such a command might be "Wash at 40 degrees" (given to a *washing machine*). In this step, 29 crowd workers were involved.

The crowd workers proposed a vast list of devices, from smart lamps, smart electricity/water meters to smart bread makers, smart bathroom sensors and garden paths with heating. For those devices, the crowd workers were instructed to formulate several natural language requests (or commands). As there were no restrictions on device selection, some devices were chosen by several users.

After collecting the results from the crowd workers, we analyzed and merged the commands and ended up with 50 scenarios, each of which containing from 2 to 23 natural language commands, 320 commands in total. All counts are given after the removal of duplicate commands and scenarios from the dataset.

A scenario is a string quadruple (D, C, P, L), where D is a device, which is typically represented by a list of sensors. C is an array of potential commands, connected to this device. P is a multidimensional array of parameters, associated with each command, and L is a set of arbitrary natural language commands, generated by informants (in our case: crowd workers). Therefore, a scenario contains starting phrases (first-turn utterances), which are typically used as commands for the device.

An example of a scenario[5] is as follows:

- Device D: a *washing machine* that has a *sensor for load detection*;
- Commands C: *wash*[6];
- Parameters P: *washing temperature* of type *integer*, this parameter also has a default value.

[5]In the text of the paper all examples are given in English translation. The original dataset is in Russian language.

[6]This command starts the washing process, but does not turn the washing machine on.

- Natural language commands L:

    1. *Start washing at 23:05!*
    2. *Wash the clothes!*
    3. *Wash at 90 degrees!*

As a side note, the crowd workers have produced paraphrases for some commands, which are preserved in the dataset and can be used in future work, for example for paraphrase detection for a command.

2.3 Step II: Dialogue Manager States and Multi-turn Dialog Generation

Given the first-turn commands and the collected scenarios from step I, we analyzed potential system response states based on those command. Assuming a correct understanding of the command by the dialogue system, this led to a list of 12 possible system reactions to the first-turn requests, which correspond to dialogue manager states.

To make those dialogue manager states more intuitive, we present some examples of system reaction codes and natural language responses for a scenario with a washing machine:

1. System State / Response code 1. *All necessary devices, commands and parameters have been found, there are no conflicts, the command will be run now.*
   Corresponding natural language response: *Washing starts!*

2. System State / Response code 2. *All necessary devices, commands and parameters have been found, but a multiple choice problem exists (several entities/devices satisfy the query).*
   Corresponding natural language response: *Which washing machine should be used: the one in the kitchen or the one in the bathroom?*

3. System State / Response code 3. *A parameter or parameter value was set incorrectly.*
   Corresponding natural language response: *The washing temperature cannot be set to 15 degrees. Please choose between 30 and 90 degrees!*

As mentioned, the system reaction codes correspond to the states of the dialogue manager, given that user request was correctly understood by the system. The proposed list of 12 dialogue manager states includes states of parameter conflicts, scheduled requests, identifying unknown or fuzzy parameters, cases with multiple choices, non-saturated frames, mistakes and the reporting of side effects.

After the definition of the 12 dialogue manager states, we did a *second run of crowdsourcing processes*. This time, the crowd workers were given the list of first-turn natural language utterances collected in step I, and the list of 12 system reaction codes. The workers were asked to verbalize the system reaction code in natural language, and create a dialogue of user responses and system reactions which completes the dialogue successfully.

The final dataset of dialogues generated in step II includes 640 dialogues, and number of turns per dialogue varies from 2 to 5. A sample dialogue is given below (U - user, S - system):

– U: *Turn on the light!*

– S: *Do you want to turn on the light in the kitchen or in the whole flat?*

– U: *Only in the kitchen.*

– S: *Ok, done!*

3 Experiments on Command Type and Slot Identification

In this section we describe two experiments performed on the commands collected in step I. So, the raw data comprises the 320 first-turn natural language requests to the smart environment. The *goal* of the experiments is to automatically classify the commands a) by the type of command, b) by *slots*, which represent various characteristics of the command, for example if the command contains a condition or specifies some device parameters. This classification of basic command type and slots will help in the construction of a dialogue system, which is part of future work.

3.1 Description of Command Classes

For the purpose of classifying the first-turn user commands, we use six features. The first feature is the command type:

Command type: We currently distinguish three command types: *request* – 1, (e.g., "*How much water did I spend last month?*"), *explicit command* – 2, which can be directly mapped to an entity in a knowledge base via its label, and *implicit command* – 3, which cannot be directly mapped, and the system has to guess, what to do (e.g., "*Reduce the temperature!*").

These three types have little in common with typical speech acts classification and were introduced as a result of distinguishing implicit and explicit commands and requests (typically, to databases external to the IoT system) from commands.

The other **5 features are the slots** describing important command characteristics[7]:

1. Device, 1 - if mentioned, 0 - otherwise ("Turn on *the desk lamp.*");

2. Condition, 1 - if mentioned, 0 - otherwise ("*If the outdoor temperature is higher than 25°C*, set the split system to +18°C.");

3. Location, 1 - if mentioned, 0 - otherwise, ("Start watering in *greenhouse 17.*");

4. Parameter, 1 - if mentioned, 0 - otherwise, ("Set the brightness of the chandelier to *600 Lumen.*");

5. Schedule, 1 - if mentioned, 0 - otherwise, ("Turn on the oven *at 9 pm.*").

[7]Examples of requests are given in brackets, and the entity, corresponding to the slot type, is italicized

Thus, a vector of labels, characterizing a command, looks as follows:
(2, 0, 1, 1, 0, 0) for the command "*Turn off the light, when I leave the room*".
This is an explicit command (2), a device is not mentioned explicitly (0), a condition (1) and location (1) are present, no parameters are specified (0) and the command is not scheduled (0).

3.2 Model Setup

Given those six features, we created a gold-standard dataset by manually annotating each of the 320 commands collected in step I with a vector of 6 features: element at '0' position denotes command type, elements at positions '1-5' denote the slot types. The sequence of *slot types* is a one-hot vector, if objects corresponding to the particular slot are present in the request, the value is '1', else '0'.

For model training and evaluation, we used word embedding vector representations of the user commands. First, we train word vectors on the Aranea Russicum Maius corpus[8]. The size of the corpus is 1,200,001,911 tokens. The model was trained on the raw text without preprocessing using the word2vec library [Mikolov et al., 2013], using the skip-gram algorithm and a vector size of 100 dimensions. With those word embeddings, we created vector representation of the whole user request by averaging the word vectors of words in a given request. In future work, we will experiment with more elaborate methods to represent the commands, for example weighting schemes for words, sentence embeddings, etc.

3.3 Experiment Results for Command Type Classification

First, we classify the commands into the three command types described above. In the gold data labelled by experts, the command types are distributed as follows: 64% of utterances are explicit commands to the system, 25% are requests and 11% are implicit commands. The classification of command types in those three classes was performed with the Weka machine learning library[9] using the word embedding representations of user commands.

3.4 Experimental Results for Slot Type Classification

The same classification algorithms and word embedding representations were used in the slot type classification tasks. According to manual annotation, the slots in user utterances are distributed as follows: Devices are mentioned in 31% of utterances, conditions - in 27%, locations - in 24%, and parameters and scheduling - in 14% each. Here, we have five binary classification tasks per user command, one per slot type.

Classification results for *Device*, *Condition* and *Location* slots can be adopted as baseline and used in future work. However, as indicated by predication accuracy of the slots *Parameter* or *Scheduling*, the baseline classifier seems to not have learned to discriminate those. For the *Parameter* and *Scheduling* slots, a similar prediction

[8] `http://aranea.juls.savba.sk/aranea_about/index.html`
[9] `https://www.cs.waikato.ac.nz/ml/weka/`

Table 1: Utterance Type Classification

Classification Algorithm	Accuracy
k-nearest Neighbour (IBk)	0.7531
C4.5 Decision Trees (J48)	0.6406
Rule Learner (PART)	0.6313
Naive Bayes (NaiveBayes)	0.6719
Holte's 1R (One Rule) Classifier (OneR)	0.6125
Support Vector Machines (SMO)	**0.7937**
Logistic Regression (Logistic)	0.6562
AdaBoost (AdaBoostM1)	0.6500
LogitBoost (LogitBoost)	0.7500
Descision Stumps for Boosting (DecisionStump)	0.6344

accuracy can be reached by always predicting value 0. Moreover, we plan to evaluate the classifiers on more balanced datasets in future work.

4 Conclusions

The paper presents and applies a methodology for creating datasets for voice communication with IoT devices. The method focuses on user commands which lead to specific dialogue manager states, and are finally resolved in a dialogue between the IoT system and the device user. The datasets created with crowdsourcing reflect those dialogue elements, and were also applied to classify user commands according to six features which will be used for dialogue state tracking in future work.

The main contributions of the paper include (i) a procedure for crowdsourcing voice control data from end users, (ii) the provision of datasets, (iii) automatic morphosyntactic annotation of the datasets, and (iv) baseline evaluation of various machine learning algorithms. The datasets created, esp. the natural language responses, which users formulated on behalf of the system, may also be re-used in other tasks, for example as patterns in natural language generation tasks.

Finally, there are multiple directions for future work. We plan to extend the dialogue datasets with crowdsourcing techniques, to improve on the classification baselines presented in the experiments section, and to apply the collected data for the creation of IoT dialogue systems.

Acknowledgments

L.Kovriguina acknowledges support from the Russian Fund of Basic Research (RFBR), Grant No. 16-36-60055. G.Wohlgenannt acknowledges support from the Government of the Russian Federation (Grant 074-U01) through the ITMO Fellowship and Professorship Program.

Table 2: Slot Type Classification

2*Classification Algorithm	Accuracy for each slot type				
	Device	Condition	Location	Parameter	Scheduling
k-nearest Neighbour (IBk)	0.6594	0.7000	0.6781	0.8312	0.8094
C4.5 Decision Trees (J48)	0.7219	0.7250	0.7406	0.8281	0.7906
Rule Learner (PART)	0.6906	0.7250	0.7375	0.8125	0.7937
Naive Bayes (NaiveBayes)	0.7031	0.7437	0.7094	0.7437	0.7844
Holte's 1R (One Rule) Classifier (OneR)	0.6406	0.6750	0.6719	0.8375	0.8281
Support Vector Machines (SMO)	**0.7906**	**0.7906**	**0.8000**	**0.8531**	**0.8781**
Logistic Regression (Logistic)	0.6781	0.7469	0.6906	0.7469	0.7937
AdaBoost (AdaBoostM1)	0.7437	0.7531	0.7625	0.8312	0.8469
LogitBoost (LogitBoost)	0.7156	0.7281	0.7718	**0.8531**	0.8656
Descision Stumps for Boosting (DecisionStump)	0.6906	0.7281	0.7594	0.8625	0.8656

References

[Aldrich, 2003] Aldrich, F. K. (2003) Smart Homes: Past, Present and Future // Inside the Smart Home, Springer, London, 2003, pp. 17–39.

[Jia et al., 2017] Jia, R., Heck, L., Hakkani-Tür, D., and Nikolov, G. (2017) Learning concepts through conversations in spoken dialogue systems // Acoustics, Speech and Signal Processing (ICASSP) 2017 IEEE International Conference on., IEEE, 2017, 5725–5729.

[Jurafsky and Martin, 2017] Jurafsky, D., and Martin, J. H. (2017) Dialogue Systems and Chatbots // Speech and Language Processing, 2017, 11(2):121–137.

[Kovriguina et al., 2017] Kovriguina, L., Shilin, I., Putintseva, A., and Shipilo, A. (2017) Russian Tagging and Dependency Parsing Models for Stanford CoreNLP Natural Language Toolkit // International Conference on Knowledge Engineering and the Semantic Web, Springer, 2017, 101–111.

[Lowe et al., 2017] Lowe, R. T., Pow, N., Serban, I. V., Charlin, L., Liu, C.-W., and Pineau, J. (2017) Training End-to-end Dialogue Systems with the Ubuntu Dialogue Corpus // Dialogue & Discourse, 2017, 8(1), 31–65.

[Mesnil et al., 2015] Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., He, X., Heck, L., Tur, G., Yu, D., et al. (2015) Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding // IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), 2015, 23(3), 530–539.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013) Efficient estimation of word representations in vector space // arXiv preprint arXiv:1301.3781, 2013.

[Portet et al., 2013] Portet, F., Vacher, M., Golanski, C., Roux, C., and Meillon, B. (2013) Design and evaluation of a smart home voice interface for the elderly: acceptability and objection aspects // Personal and Ubiquitous Computing, 2013, 17(1):127–144.

[Profanter, 2012] Profanter, S. (2012) Cognitive architectures // HauptSeminar Human Robot Interaction, 2012.

[Serban et al., 2015] Serban, I. V., Lowe, R. T., Charlin, L., and Pineau, J. (2015) A Survey of Available Corpora For Building Data-Driven Dialogue Systems // arXiv preprint arXiv:1512.05742, 2015.

[Sungjin et al., 2016] Sungjin, L., and Stent, A. (2016) Task Lineages: Dialog State Tracking for Flexible Interaction // Proceedings of the SIGDIAL 2016 Conference, 2016, pp. 11–21.

[Vernon, 2014] Vernon, D. (2014) Artificial cognitive systems: A primer // MIT Press, 2014.