MightyFields Voice: Voice-based Mobile Application Interaction

Jernej Zupančič
Jožef Stefan Institute
Ljubljana, Slovenia
Jožef Stefan International
Postgraduate School
Ljubljana, Slovenia
jernej.zupancic@ijs.si

Miha Štravs
Faculty of Mathematics and Physics
Ljubljana, Slovenia
Faculty of Computer and
Information Science
Ljubljana, Slovenia
miha.stravs996@gmail.com

Miha Mlakar
Jožef Stefan Institute
Jamova cesta 39
Ljubljana, Slovenia
miha.mlakar@ijs.si

ABSTRACT

We present MightyFileds Voice (MFVoice), a service and an extension of the MightyFields application that enables voice interaction with a mobile application. The user can issue voice commands for transitioning between application views and filling out the forms. Google speech-to-text engine is used to obtain text, which is then fed into the developed MFVoice service together with the structured application view representation. MFVoice service then returns appropriate action to take, which is executed by the Mighty Fields application extension. The MFVoice natural language understanding service was tested in real-life use cases, achieving 93% intent recognition accuracy, 88% entity recognition success when the system was used as intended. When no training to the user was provided, intent and entity recognition achieved 68% and 52% accuracy, respectively. Note that in case of no training provided, the users assumed general knowledge of the language semantics, which is out-of-scope for the current state-of-the-art research in natural language.

KEYWORDS

voice assistant, voice interaction, natural language understanding

1 INTRODUCTION

Interaction with devices by voice has become quite common in recent times. More known examples of applications allowing voice commands are voice assistants like Cortana [4] and Siri [1]. Voice interaction is attractive to users as it offers a hands-free application interaction and is therefore a desired feature in many applications. This feature is useful for people with spelling difficulties. It can also help those with physical disabilities who often find typing difficult. The proposed service is not used for two-way conversation, as in platforms such as the one from Rasa [3]. However, the part of the service used to recognize user's command, is very similar to the ones from other virtual assistants. The modifications applied take into account the specifics of the task at hand.

In this paper we focus on the task of filling out custom forms through the voice interaction. Here, a custom form is a small information gathering application, made for specific purpose, e.g., electric grid inspection form, or police report regarding an incident. Since the domain is open ended, i.e., each individual can make his or her own custom forms, the voice understanding feature cannot be specialized and has to work satisfactory in general setting. Three steps are performed to enable voice interaction. First, speech is transformed into text by using Google speech-to-text (STT) engine [2]. Second, approach from [5] is utilized to extract intent keywords. The full intended command is then inferred based on what the user is currently seeing on the screen and from the rest of the spoken words. Third, the recognized action is performed within the application itself.

In Section 2, an architecture of our service is presented. In Section 3, we present our MFVoice natural language understanding (NLU) service and show its implementation. We then explain the tests conducted on the service and their results in Section 4 and discuss them in section 5. We conclude the paper with a summary in Section 6.

2 MFVOICE ARCHITECTURE

MFVoice comprises several parts (Figure 1) that enable voice interaction:

(1) MF application itself: this is the main MightyFields application.
(2) MF agent: the program that enables programmatic access to the application view - reading and interacting.
(3) STT: a service that transforms spoken commands into text.
(4) MFVoice NLU service: the service that parses free text and returns structured information about recognized intent and entities.

3 THE MFVOICE NLU SERVICE

The MFVoice NLU comprises the following steps (Figure 2):

(1) Application view context processing
(2) Intent recognition
(3) Entity recognition

When the application context and transcription of the voice command are provided to the NLU application programming interface (API), the service first identifies possible actions to take, given the context, then it processes the context content, which in turn enables recognition of the intent and, finally, the entities. The so-obtained structured action data is then forwarded back to the MF agent, which can execute appropriate actions. In this section we will describe each of the MFVoice NLU parts in more details.

3.1 Application View Context Processing

The application view context provides structured data on the elements that are visible on the screen. This includes field labels, field IDs, possible values of fields (where applicable), interaction options, and available tabs for multi-page forms. Upon the API

Information society '20, October 5–9, 2020, Ljubljana, Slovenia

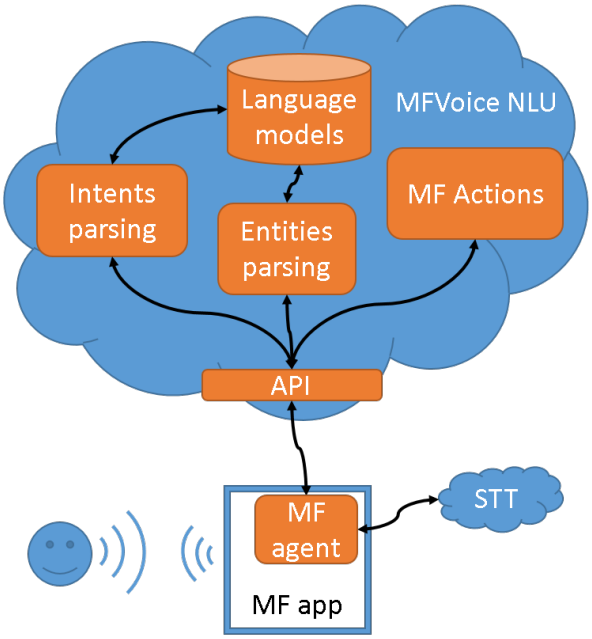


Figure 1: The MFVoice architecture overview

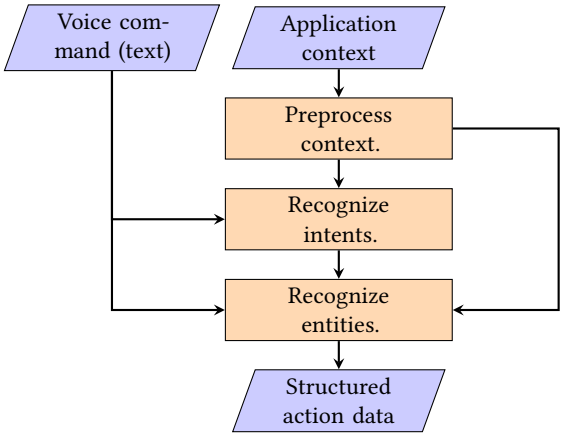


Figure 2: The MFVoice service processing pipeline

call, the application context is pre-processed. The text visible to the user is normalized and transformed into search friendly form that is used in intent and entity recognition.

The transformation is cached to speed-up its future use in subsequent steps. Least recently used cache is used, since user interacts with one application view as long as he or she does not fill-out the form in short time period.

3.2 Intent Recognition

Intent defines what the user wants the application to do. In the case of form filling, the following intents were identified:

(1) "Choose" an element from a list of elements. This is often used to pick the element from a dropdown or checklist elements.
(2) "Write" some text into a field. This is used for input or text-area elements.
(3) "Clear" the value of a field to delete a wrong value entry for any kind of element.
(4) "Tap" the field or graphical element. This is used to interact with buttons and navigate the application.

Due to non-existent training data, keyword-based intent recognition was utilized. For each intent a set of intent "key-phrases" were defined:

(1) "Choose": choose, pick, select, is
(2) "Write": is, write, input
(3) "Clear": clear, delete, remove
(4) "Tap": choose, go to, tab, click, pick

In the intent inference step, the score for each of the "key-phrases" is computed. If the highest score exceeds the predefined threshold, the corresponding intent is chosen [5]. To simplify the NLU pipeline only one intent and one field per utterance is allowed. This is especially problematic in voice assistants, since the users naturally communicate differently when talking than when writing. To resolve the disambiguation of one "key-phrase" being associated with multiple intents, the following order of intents is taken into account: "Choose", "Write", "Clear", "Tap". Further, since not every intent is possible with every field, the intent list is first filtered and only the intents that make sense in the current context are kept and iterated over. For instance, if the context comprises only of the input fields and navigation tabs, "Choose" does not make sense.

The "key-phrases" used for inferring the intent are tagged with the intent tag (IT), which is later used in the entity recognition step.

3.3 Entity Recognition

There are two types of entities present in our use-cases: "label entities" (e.g., name in *"The **name** is John."*) and "value entities" (e.g., John in *"The name is **John**."*).

Label entities are the labels of the fields in the user generated form. Since they are user generated, their value is not restricted. Their semantic meaning is sometimes harder to grasp automatically, since not much additional info is usually provided (for instance description). In general, the recognition of label entities cannot be learning-based, since the users are not expected to provide examples.

Value entities can further be divided into known value entities (button labels, the items in the drop-down lists or checkbox lists and similar) or unknown value entities (text input fields). The reasoning for known value entities is the same as for the label entities – they are user generated and cannot be learned in general. For the unknown value entities, the value corresponding to a certain label should be recognized from the free text, generated from the STT service. The unknown value entities can comprise one or several words.

Score, related to the probability of the entity appearing in the text, is used to recognize the label entities and known value entities, while a heuristic is used to recognize the unknown value entities.

3.3.1 Label and Known Value Entities Recognition. These entities are the ones set by the user in the form creation phase. To recognize the entities from free text, text similarity scores are applied and evaluated for each possible label or known value entity [5]. Only the entities with scores that exceed threshold are recognized and can be used in the next pipeline steps.

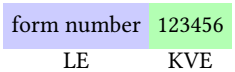
In some instances, the MFVoice NLU has to modify the text received by the STT engine. Common examples for this are:

(1) Letter-by-letter dictation transformation, e.g. in transcribed command "the form ID is 1 2 3 4 5 a" the empty spaces in the form ID have to be deleted so "12345a" is obtained.
(2) Zero padded numbers transformation, e.g. in transcribed command "the house number is 23", sometimes the dropdown values include only known value entity "023". Therefore, the preceding zeros have to be dropped when computing the text similarity.
(3) Numbers in text transformation, e.g. in transcribed command "pick option three", the number "3" is transcribed as "three". In those cases, the textual representation has to be transformed into a number.

The words that correspond to the highly-scored entities are tagged with the label entity (LE) or known value entity tag (KVE), to be later used in the unknown value entity recognition.

Some examples of label entity and known value entity recognition are:

(1)



(2)

Note that when the user speaks command "female", the MFVoice NLU service recognizes the known value entity that belongs to the field with label "sex". Additionally, even without specifying the intent keyword, the application logic infers that the user wants to pick the option "female" in the field "sex".

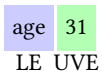
Since the text similarity metrics are used for scoring the labeled and known value entities, in some cases the entities are not recognized correctly:

(1) Multi-word synonyms are not recognized, e.g. "city" ~ "place of living"

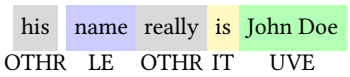
While the systems supports synonyms, they have to be manually entered by the form creator and are therefore less practical.
(2) Multiple occurrences of the same or very similar label entities or known value entities cannot be properly disambiguated. Consider, for example, a form that comprises house number field with possible value "4", and household size field also with possible value "4". User usually fills the form in a linear way, top to bottom. When the user encounters the first of the mentioned field, he or she may voice command "four". In this case, NLU service will provide two possible actions: "house number is 4" and "household size is 4".

3.3.2 Unknown Value Entity Recognition. The unknown value entity recognition is computed only when the intent "Write" is considered, since this is the only type of the"open" form field that allows for unknown values. The following heuristic is used to tag the unknown value entity (UVE):

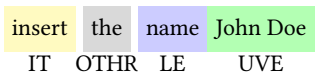
(1) If IT tag is not present in the text, every word not tagged with LE is tagged with UVE.



(2) If IT tag is present in the text, then begin tagging word to the left or to the right of LE-tagged word with OTHR tag. Stop if text-end or IT-tagged word are reached. Check if there is any remaining word:
(a) If there are remaining words, tag those with UVE tag.

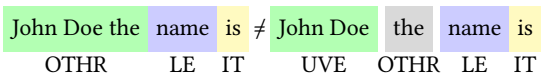


(b) If there are no remaining words, re-tag all the words to the right of LE tag with UVE tags.

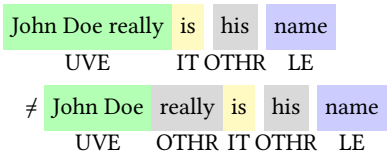


The previous steps capture the majority of the unknown value entity recognition cases. However, there are still commands that would not be understood by the MFVoice NLU service:

(1)



(2)



4 TESTING

The MFVoice NLU service was tested in two ways: laboratory testing and real-world testing. For laboratory testing, the text was entered into the service directly, bypassing the STT service. This way, the STT performance issues were ignored and only the recognition capability of the MFVoice NLU service was tested. The examples, however, were still obtained from the final MFVoice users. The test user was presented with an application screen and told to fill the form using only his or hers voice.

For the real-world testing, the users were given written instructions on how to use the app, however, no instruction on how to actually voice commands were given. First, the form was filled out using screen and keyboard interactions. Second, the field that a user wants for fill with a voice command was marked. Third, voice interaction was activated and the command was spoken. Fourth, the transcribed voice command, the context, and the marked item were stored for future analysis. We did not provide any examples on how to use MFVoice. This allowed us to research what the users actually expect from the system.

The forms used in testing included six free-text input field widgets (name, surname, age, settlement, street, house number), one radio widget with two options (gender: male, female), one checkbox field with five options (language: Slovene, Slovak, Spanish, Swedish, Sumerian), and four dropdown fields (country, settlement, street, and house number).

4.1 Laboratory Testing Set-up and Results

We have gathered 70 and 69 commands for application interaction in Slovenian and English languages, respectively. Laboratory testing is performed upon each git push to the code repository and is run within the continuous integration pipeline. This enables us to track the performance of the MFVoice NLU pipeline.

Table 1: Intent confusion matrix for commands in Slovenian

	write	choose	clear	tap	missing
write	22	0	0	0	2
choose	0	21	0	0	0
clear	0	0	2	0	0
tap	0	0	0	20	3

4.1.1 Intent Recognition. After each continuous integration pipeline run, the intent confusion matrix is computed. Table 1 is an example of the intent confusion matrix for voice commands in Slovenian for the last version of MFVoice. According to the matrix, the accuracy of intent recognition is above 90%. The only errors were the ones, where the system was not able to determine the item to be interacted with, which was labeled with the "missing" classification label.

4.1.2 Entity Recognition. For each command also the field labels and values recognized by the NLU service and the ground truth labels and values are compared. Examples where the NLU fails to recognize the label or value correctly are:

(1) "Age 26 years." Expected value: "26", got "26 years"
(2) "She is 26 years old." Expected label: "age", got nothing.
(3) "She lives in Ljubljana." Expected label: "Place", got nothing.

4.2 Real-life Testing Set-up and Results

We have gathered 172 spoken voice commands in the real-life setting in Slovenian. Unfortunately, there were only 86 commands that were labeled correctly by the test users and STT performed well there. STT issues occurred in 42 out of 172 cases (24%). These could either be result of too much background noise, command not being recorded properly, or just the problem with the STT service used for the Slovenian language. Incorrect user labeling occurred in 42 out of 172 cases (24%). The most common mistakes in those cases were: the user forgot to set the ground truth either by entering the value or choosing the item, the user obviously picked the wrong item (e.g., for command "the name is John" an item with label age was selected).

Out of 86 valid commands, 45 were recognized correctly and 41 incorrectly. For 23 cases the label value was completely missing and could not be inferred from the surrounding text (e.g., "John", "45", "Ljubljana"). For 18 cases the label value could be inferred from the surrounding text (e.g., "he is 23 years old", "she lives in Ljubljana"). In some cases (12) this would require some general reasoning about the words and their relations and in other the unknown value entity included additional text, e.g. "his name miki", was not recognized because of minor STT-engine mistakes (1), or the known value entity score was not high enough to be included (5). This results in 88% accuracy for entity recognition when the system was used as intended, 72% when the synonyms were assumed, and 52% when general knowledge of the language semantics was assumed.

Note that the testing was performed without some planned features implemented. The *Zero padded numbers transformation* and *Numbers in text transformation* steps were missing. The accuracy percentages should improve to 94%, 76%, and 56% for uses as intended, assuming synonyms, and assuming general knowledge of language semantics, respectively.

5 DISCUSSION

According to the results, the intent recognition process performs very good, despite the fact that it is only based on keyword recognition and the context processing. We do not think that any additional work would benefit the performance in this regard, with the exception of adding additional intent keywords, which will be obtained during the application usage.

After the user familiarizes with the way the MFVoice application works, also the named entity recognition performs well. Most of the errors were actually a result of a user expecting the system to be too advanced. All 43 incorrectly recognized entities were the result of MFVoice not being able to reason that, for instance, "John" is a person name. While this could be done for certain special cases, e.g. person names and geographic names, at the moment this cannot be solved in general. This is a result of letting the users to create their own forms, which are often very domain specific. In the future we will perform the testing of the system after users are given some basic training on how to use MFVoice. This should greatly improve the percentage of properly labeled instances and also help us uncover additional edge cases to be addressed by the entity recognition pipeline.

The MFVoice NLU was designed in a way to easily support multiple languages. In the current form, to support a new language, the translations of the intent keywords and language word vectors have to added. For certain languages the module for unknown value entity has to be adjusted, since the sentence syntax can be different. This enabled us to quickly add support for English, after Slovenian voice interaction performed well.

6 CONCLUSION

In this paper we presented our service that is used for filling forms with voice commands in a mobile application. While some operating system do include voice interaction, e.g., Cortana [4] and Siri [1]), their use in a dedicated application is limited. MFVoice enables more advanced voice interaction. MFVoice application first gets the text which was converted from speech by using the Google STT engine [2]. Then, the MFVoice NLU service uses keyword recognition and context preprocessing to infer the command the user intended. Because of the simplicity of the implementation, the service is less accurate when commands are voiced in the form of long and complex sentence. However, this simplicity does make the service more robust and accurate with commands voiced in concise form. We believe that users should have a comfortable user experience, after they get used to forming commands in a more concise manner.

ACKNOWLEDGMENTS

Comland d.o.o. funded the research presented in this paper.

REFERENCES

[1] Apple. 2020. Siri. https://www.apple.com/siri/. (2020).
[2] Google. 2020. Speech-to-text: automatic speech recognition. https://cloud.google.com/speech-to-text/. (2020).
[3] Rasa Technologies Inc. 2020. Rasa. https://rasa.com/. (2020).
[4] Microsoft. 2020. Cortana - your personal productivity assistant. https://www.microsoft.com/en-us/cortana/. (2020).
[5] Miha Štravs and Jernej Zupančič. 2019. Named entity recognition using gazetteer of hierarchical entities. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 768–776.