INTERSPEECH 2017: Show & Tell Contribution
August 20–24, 2017, Stockholm, Sweden

Modifying Amazon's Alexa ASR Grammar and Lexicon – A Case Study

Hassan Alam[1], *Aman Kumar*[2], *Manan Vyas*[3], *Tina Werner*[4], *Rachmat Hartono*[5]

BCL Technologies, San Jose, California, USA

hassana@bcltechnologies.com, amank@bcltechnologies.com, mvyas@bcltechnologies.com, twerner@bcltechnologies.com, rachmat@bcltechnologies.com

Abstract

In this proof-of-concept study we build a tool that modifies the grammar and the dictionary of an Automatic Speech Recognition (ASR) engine. We evaluated our tool using Amazon's Alexa ASR engine. The experiments show that with our grammar and dictionary modification algorithms in the military domain, the accuracy of the modified ASR went up significantly - from 20/100 correct to 80/100 correct.

Index Terms: speech recognition, human-computer interaction, grammar modification

1. Introduction

In this study we evaluated the Amaozn's Alexa Automatic Speech Recognition (ASR) engine and (1) modified the ASR grammar; (2) added new words and constructions to the ASR dictionary; (3) integrated the new grammar and the dictionary with the acoustic matching module. These experiments show that (1) we can successfully change the grammar and the dictionary with ease using domain-specific natural language; (2) the accuracy of the modified ASR goes up significantly in the military domain after we modify the grammar and the dictionary. Amazon's Alexa uses a powerful ASR engine that provides capabilities to the end-users to interact with the tool in a more intuitive way using their normal speech. The more the user uses Alexa, the tool gets used to customer's speech pattern, vocabulary, and personal preferences because of its machine learning capabilities. We used Alexa's this very feature to our advantage in building this tool.

2. Grammar and Dictionary Modification Modules

2.1. Grammar Modification

The fundamental goal of this module is to revise the ASR grammar that directly modifies the parsing rules. In addition to adding newer constructions, this module prohibits 'noise' or irrelevant constructions to be processed. The schema for this grammar modification module is given in the Figure 1 below.

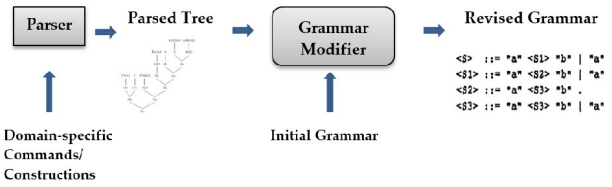

Figure 1. *Architecture for the grammar modification module*

The figure above schematically explains the process of revising the grammar in the backend (parser level processing) of this system. For each new domain-specific commands or constructions we generate the parse tree which broadly depicts the representation of the grammar rules need to generate the newer set of commands or constructions. These steps are repeated for each new construction reusing the revised grammar from earlier iterations. In the event that a new construction outputs multiple parsed trees, the algorithm will iterate for each parse tree until all the passed trees are exhausted and processed.

The specific steps to implement this algorithm are given below:

1. For each non-terminal <N> in the parse tree, except the 'start' of the parsed tree
   a. Add a rule that define a new terminal <N'> such that <N'> generates all phrases that <N> generates except for the phrase in the new constructions that <N> generates.
   b. Add a rule to define a new non-terminal $<N_0>$ such that $<N_0>$ generates only the phrases in the new constructions that <N> generates.

2. Modify the rules that contains the start of the parsed tree so that no start function generates parsed tree without the new construction.

2.1.1. Scalability

In order to manage the scalability problem we implemented *Minimum Description Length* (MDL) methods to combine rules and build a comprehensive and compact grammar modification module. We used methodology used by Grunwald (2005), Zadrozny (1997) to implement the MDL algorithm.

2.2. ASR Dictionary Modification

We updated the Alexa's ASR dictionary at two levels –

1. *Lexical Lookup* - where newer lexical forms from the military domain are added to the dictionary;
2. *Pronunciation lookup* – The audio pattern of the new token or lexical item is modeled to match the spoken input to a grammar

Amazon's Alexa uses multi-step techniques to perform this important process. We leveraged upon the existing set up rules to manually add the new lexical items and use the training module for recognition purpose.

The ASR engine such as Alexa, with a large vocabulary converts each normalized token to sequence of phoneme or a set of possible sequences of phonemes. We used the standard text-to-phoneme conversion steps. Some of the standard text-to-phoneme conversion processes that we implemenedt are quoted here.

- ***Pronunciation lexicon lookup****:* One of possibly many lexicons available to a recognizer can provide the phoneme sequence for a token. Both the ABNF

Form and XML Form permit a grammar to specify one or more lexicon documents. Recognizers typically provide a built-in lexicon for each supported language though the coverage will vary between recognizers. The algorithm by which the lookup resolves a token to a pronunciation is defined by the lexicon format and/or the speech recognizer and may be language-specific. Case-insensitive string matching is recommended.

- ***Morphological analysis**:* this recognizer has the capability to determine the transformation from a base token and phoneme string to a morphological variant and its pronunciation. For example, given the pronunciation for "Hyundai" a rule could infer the pronunciation for the pluralized form "Hyundai's".
- ***Automatic text-to-phoneme conversion**:* for many, but not all, languages and scripts there are rules that automatically convert a token into a phoneme sequence. For example, in English most but not all words ending with the letter sequence "ise" end with the phoneme sequence "ai z". A speech recognizer may use automated conversion to infer pronunciations for tokens that cannot be looked up in a lexicon.

3. Experiments

We ran two experiments toward this preliminary study. The results of the two experiments are given in Table 1. In the first experiment (Experiment 1) we used the ASR system (Amazon's Alexa) as is, meaning we did not modify the grammar, the dictionary and the acoustic matching modules. In the second experiment (Experiment 2) we modified those three modules using methods described in section 2.

For both the experiments, the test data were obtained in two domains – (1) camera control (cell phone camera) commands; (2) military domain commands.

In experiment 2, the grammar was modified manually by using simple natural language command entries. The dictionary was updated with lexical items such as: torpedo, Mark VIII torpedo, Fume Whitehead, SCS, ECM, Mayday etc. In this experiment we modified the grammar by adding commands such as "I want to capture this view!", "Arm Mark VIII torpedo", "Trajectory holding steady", "Mayday" etc. in natural language. In the backend of this proof-of-concept tool, the ASR grammar parser gets modified using the method discussed in section 2. We also checked the dictionary module. In this case, we didn't have to add new entry as the command words were already in the dictionary.

Once the system updates the grammar and/or the dictionary, it gives the response (e.g., 'Launching Camera').

Results of the preliminary system in the military domain are given in Table 1.

If Experiment 1 (with no grammar or dictionary modification) only 20/100 commands in the military domain were successfully implemented. In Experiment 2, however, once we modified the dictionary and the grammar the success rate went up to 80/100, which is a significant jump.

4. Conclusions

In this preliminary study we developed a grammar and a dictionary modification tool that modifies the parser of an existing ASR engine, Alexa using natural language interface. On evaluation the tool shows a significant improvement in the speech recognition tasks. In the future we plan to use this tool to other ASR engines such as Google Talk and Apple Siri.

Table 1. *Evaluation results – sample commands*

Commands (sample)	Experiment 1 Response	Experiment 2 Response
Domain:: <<**Camera**>>		
"Stop, forward, back, left, right, tilt, zoom camera"	Success	Success
"Stop, forward, back, left, right, tilt, zoom"	Failure	Success
"Take a picture"	Success	Success
"Capture image"	Success	Success
"I wanna take a pic"	Success	Success
"I wanna snap a pic"	Success	Success
"Can you open the camera"	Success	Success
"Capture image now"	Success	Success
"Can you capture this view?"	Success	Success
"Can you zoom to <thing not in camera's field of view>?	Failure	Failure
Domain:: <**Military**>		
"Arm Mark VIII"	Success	Success
"Arm Mark VIII torpedo"	Failure	Success
"Launch Fume Whitehead"	Failure	Success
"Launch Fume Whitehead torpedo"	Failure	Success
"Base 1, primary guidance looks bad, switching to SCS"	Failure	Failure
"Go active on ECM"	Failure	Success
"Jam 'em up"	Success	Success
"ECM active"	Failure	Success
"Switch from the secondary back to the primary control system"	Failure	Success
"Mayday"	Failure	Success

5. References

[1] H. Alam, A. Kumar, F. Rahman, Y. Tarnikova,. (2007). Spoken Language Understanding Software for Language Learning. *4th International Conference on Cybernetics and Information Technologies, Systems and Applications (CITSA 2007) jointly held with the 5th International Conference on Computing, Communications and Control Technologies (CCCT 2007)* 2007. Won the Best Paper Award!

[2] D. Duchier, & Y. Parmentier. High-level methodologies for grammar engineering, introduction to the special issue. *Journal of Language Modelling,* 3(1), 5-19. 2015.

[3] P. Grunwald. A tutorial introduction to the minimum description length principle. In: *Advances in Minimum Description Length: Theory and Applications, (Peter Grunwald, In Jae Myung, Mark Pitt, eds.), MIT Press*, pp. 23–81. 2005.

[4] Zadrozny et alAutomatic indexing and aligning of audio and text using speech recognition. Hamed A Ellozy, Dimitri Kanevsky, Michelle Y Kim, David Nahamoo, Michael Alan Picheny, Wlodek Wlodzimierz Zadrozny. . 1997