DEVELOPMENT OF ALGORITHM FOR COMBINATION OF CLOUD SERVICES FOR SPEECH CONTROL OF CYBER-PHYSICAL SYSTEMS[1];

Andrej Škraba[1], Vladimir Stanovov[2]
Eugene Semenkin[2], Andrej Koložvari[1], Davorin Kofjač[1]

[1] University of Maribor, Cybernetics & Decision Support Systems Laboratory
Faculty of Organizational Sciences, Kidričeva cesta 55a, 4000 Kranj
[2] Reshetnev Siberian State University of Science and Technology
Institute of Computer Science and Telecommunications
e-mails: {andrej.skraba, davorin.kofjac}@fov.uni-mb.si, andrej.kolozvari@gmail.com
{eugenesemenkin, vladimirstanovov}@yandex.ru
[1] Slovenia, [2] Russian Federation

Abstract: Paper describes the development of algorithm for efficient harvesting of speech recognition cloud services with application to the cyber physical systems. Theoretical consideration of improvement is provided as well as the mandatory condition for forming set of substitute tables for the limited set of commands. Experimental results of the speech recognition correctness for Google speech API and IBM Watson are provided on a small sample as well as the latency.

Key words: Speech recognition, Cloud, Internet of Things, Cyber-physical Systems

1. INTRODUCTION

Cloud speech recognition has been successfully applied in different application areas, for example, in control of the wheelchairs for disabled persons [1; 2] as well as in the combination of other novel technologies such as motion control [3]. Cloud speech recognition services have several benefits over classical standalone systems. The most important is a tight integration with Internet and its users, which continuously feed the database and provide corrections. On the other hand, there are several drawbacks, such as latency due to the network communication, word error rate (WER) as well as technical difficulty to use cloud Application Programming Interface (API). Since there are more new cloud speech recognition services available, there is a need to develop efficient algorithms, which will combine speech recognition results from different cloud APIs to improve WER as well as latency [4, 5, 6, 7,

[1] The paper has been presented on the sixth international workshop on mathematical models and their applications, 13-15.11.2017, Krasnoyarsk, Russian Federation and it is not published in the Conference Proceedings.

8, 9]. In the present paper, we will consider development of an algorithm to combine several cloud speech recognition services and compare two different services and the type of data that is available from API.

2. METHODOLOGY

WER is a common metric of the performance of a speech recognition or machine translation system defined as:

$$WER = \frac{\mathrm{S} + \mathrm{D} + \mathrm{I}}{\mathrm{N}} \quad (1)$$

where: S – the number of substitutions; D – the number of deletions; I – the number of insertions; N – number of words in the reference (N=S+D+C); C – the number of the corrects

There have been several different approaches developed recently [10, 11, 12, 13]. However, for the control of cyber-physical system, usually only the predefined smaller set of commands need to be recognized. In the case of the wheelchair [1, 2, 3], these commands are: W = {go, stop, left, right, back, spin left, spin right, go left, go right, back left, back right}. Therefore, we define CER as the ratio between the number of erroneous executions of given commands and the number of all issued commands:

$$CER = \frac{\mathrm{Ce}}{\mathrm{Nc}} \quad (2)$$

where C_e is the number of the erroneous commands and N_c is the number of all issued commands. CER for different commands is different as shown by the following test. The test was performed with 20 subjects which were issuing commands from the set W. Each subject had issued each command ten times. By each participants 120 commands were issued. Experimentally, we have determined the following distribution for the set W. (Fig. 1). Distribution is not symmetrical, indicating that the control commands could be selected also according to the lowest CER; for example, the command "stop" has a lower CER than "break"; therefore, we have declared additional "stop" word, i.e. it's "synonym" in the synonym table, to provide better CER in the case of the most critical command (to stop cyber-physical wheelchair).

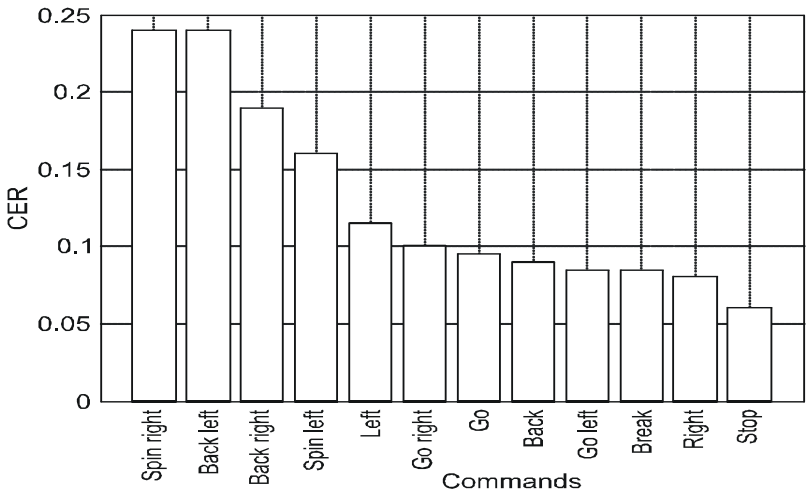

Fig. 1. Distribution of CER according to the command set W

To further improve the CER, we have examined the possibilities of joining several cloud speech recognition services. We have developed the program to combine Google speech API and IBM Watson speech API. These APIs can work in parallel, receiving inputs from two open browsers (on the client side) in real time from the same client. On the server side, we receive the speech recognition results and process them to execute the proper action. The code was developed in JavaScript applying nodejs and socket.io library.

3. RESULTS

To examine the type of data, that is available from two speech recognition services, we have gathered results for issuing the following set of commands: W_2 = {go, stop, left, right, back}. Each command was issued three times to both APIs.

Table 1 shows the results of speech recognition for the command "go". Each row represents the time of recognition, i.e. the lower the row, the more time is needed for the recognition. The results were gathered faster from Google API in most cases (marked with * if otherwise). We can also observe, that in the 3rd try the Watson was faster than Google (marked by *). Here it was not our intention to measure exact latency time as in [1]. In our case Table 1 shows the interim results, that were pushed to the server. Besides the set of interim results, the confidence value is also printed.

Table 1

1st try »GO«				2nd try »GO«				3rd try »GO«			
Google		Watson		Google		Watson		Google		Watson*	
	Conf.		Conf.		Conf.		Conf.		Conf.		Conf.
go	0.010	go	0.863	go	0.010	go	1.000	call	0.010	go	0.997
go	0.943	god	0.039	go	0.900			go	0.010	co	0.003
		got	0.026	go	0.920			go	0.863		
		because	0.009								
		good	0.008								
		gold	0.007								
		going	0.005								
		call	0.003								
		goes	0.002								
		goal	0.002								

*Results issuing command "GO", * marks, that Watson provided end result faster (in all other cases, Google API was faster)*

Table 2 shows the results for issuing command "stop". In this case the interim results sets are smaller. Only in one try the Watson was faster than Google (2nd try).

Table 2

1st try »STOP«				2nd try »STOP«				3rd try »STOP«			
Google		Watson		Google		Watson*		Google		Watson	
	Conf.		Conf.		Conf.		Conf.		Conf.		Conf.
stop	0.010	stop	0.999	stop	0.010	stop	0.996	star	0.010	stop	0.996
stop	0.900	step	0.001	stop	0.900	up	0.002	stop	0.010	stopping	0.002
stop	0.950			stop	0.947	stopped	0.001	stop	0.900	stopped	0.002
								stop	0.918		

Results issuing command "STOP"

Table 3 shows the results for issuing command "left". This time, Watson didn't provide proper result in the 3rd try (denoted with @). On the other hand, the Google was correct within the same try.

Table 3

1st try »LEFT«				2nd try »LEFT«				3rd try »LEFT«			
Google		Watson		Google		Watson*		Google		Watson*@	
	Conf.		Conf.		Conf.		Conf.		Conf.		Conf.
la	0.010	left	0.966	let	0.010	left	0.982	let	0.010	last	0.929
let	0.010	last	0.020	laugh	0.010	laughed	0.013	Les	0.010	left	0.024
laugh	0.010	left	0.005	less	0.010	let	0.002	left	0.010	lest	0.022
left	1.000	laughed	0.004	left	0.010			left	0.919	let	0.020
		let	0.004	left	0.946					less	0.003
										leftist	0.002

Results issuing command "LEFT", @ marks the error in recognition

Table 4 shows the results for issuing command "right". The Google was wrong in the 1st and the 3rd try; however, the Watson was correct in both cases (marked with @).

Table 4

1st try »RIGHT«				2nd try »RIGHT«				3rd try »RIGHT«			
Google@		Watson		Google		Watson*		Google@		Watson*	
	Conf.		Conf.	RIGHT	Conf.		Conf.	RIGHT	Conf.		Conf.
price	0.010	right	0.960	horror	0.010	right	0.676	price	0.010	right	0.842
pride	0.010	alright	0.023	right	0.010	allright	0.315	bright	0.010	alright	0.151
bright	0.010	bright	0.007	right	0.900	write	0.005	bright	0.416	write	0.003
Sprite	0.010	write	0.004	right	0.813					bright	0.003
Pride	0.909	price	0.001								

Results issuing command "RIGHT"

Table 5 shows the results for issuing command "back". Here, the interim result sets are provided by Watson and are larger in 2nd try.

Table 5

1st try »BACK«				2nd try »BACK«				3rd try »BACK«			
Google		Watson		Google		Watson*		Google		Watson*	
	Conf.		Conf.		Conf.		Conf.		Conf.		Conf.
back	0.010	back	0.996	back	0.010	back	0.637	back	0.010	back	0.981
back	0.900			back	0.361	pack	0.154	Beck	0.010	Beck	0.010
back	0.785					Peck	0.090	Beck	0.710	pack	0.004
						pac	0.061			Peck	0.003
						pak	0.022			thank	0.002
						Beck	0.018				
						pick	0.010				
						PAC	0.005				
						black	0.003				

Results issuing command "BACK"

It is important to note, that we have a situation where Google API provided right results and Watson API did not and vice versa. This provides the possibility to improve the CER by the combination of different cloud APIs.

3.1. Algorithm development

One of the possibilities to provide a lower CER is to form the substitution table of final recognized words as well as of interim results. Substitute table principle is shown in Fig. 2. For example, for the given command "go", several other words could be considered as a proper result, such as {gold, no, garrick, though, goat}. For the word "back", the set of acceptable results would be: {back, Beck, but, bag, garrick, meg}. However, there shouldn't be a common element in substitute tables represented as sets, if we compare all possible pairs; this case is represented by the dashed line and x at the word "garrick", which is present in two substitute tables at the same time. The sets should be formed by experimenting with several subjects. When forming the vocabulary, the correctness of the recognition should be manually determined. The gained sets should always be tested for consistency. Therefore, the n substitution tables which could be defined as sets $A_1, A_2, A_3, \dots, A_n$ should be pairwise disjoint:

$$A_i \bigcap A_j \equiv \emptyset \ ; \ i \neq j \tag{3}$$

At the process of automatic vocabulary substitute table generation, the pairwise disjoint condition should be checked at each new word entry.

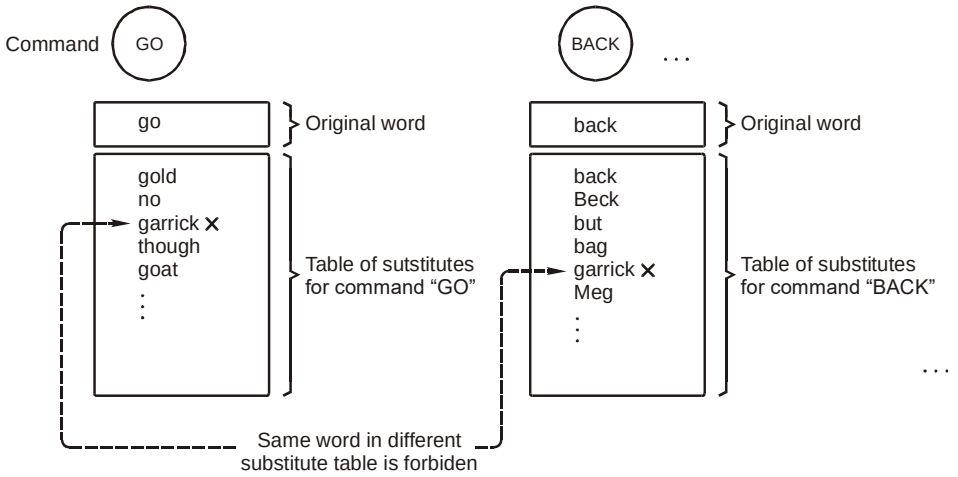

Fig. 2. Substitute's principle

According to the type of data provided by several speech APIs and considering substitution tables and timestamps of provided results, the following algorithm has been developed:

```
A1.  get user speech input
A2.  obtain set of interim transcripts and timestamps
A3.  confirm command
A4.  if Cw > Ct add interim transcript to the Cloudi command
     subset
A5.  create unique union set of words for particular command
     from Cloudi command subset
A6.  check for pairwise disjoint condition for all unique
     union sets: Ai ∩ Aj ≡ ∅ ; i ≠ j
A7.  if condition not met correct violations
A8.  order checked unique union set by interim transcript
     timestamp
A9.  if CER < CER_desired apply created system
```

Here C_w and C_t represent the confidence threshold, that is set by the user. Interim transcript timestamps determine which cloud service will be used mostly in a particular case. Even if no new word is put into the vocabulary, timestamps are used with sliding average window which determine the order in unique union set for a particular command. The algorithm needs the supervision of the user in initial phase while automatic generation could improve the performance [14, 15, 16, 17, 18, 19, 20, 21, 22].

Vocabularies of interim transcripts of each cloud system are different, for example, given the Table 1, the vocabulary of Google is: V_G(go) = {go, call}, while the vocabular of Watson is V_W(go) = {go, god, got, because, good, gold, going, call, goes, goal, co}. In this case the union is: V_G U V_W = {go, call, god, got, because, good, gold, going, goes, goal, co}. As mentioned, the order of words in the union set is important for efficient command algorithm and low CER.

To avoid the error in the recognition, certain threshold should be put in the confidence value. A particular word would then be added to the union only if the confidence value would be greater than critical confidence threshold $C_w > C_t$.

Table 6 shows possible output combinations for the three cloud APIs (for example Google, Watson, and the third, hypothetical cloud, which we name Cloud3). In the first three columns "0" represents error when recognizing a command and "1" represents correctly recognized command. In the last column, the output is represented, where "0" represents error in overall recognition and "1" correct overall recognition. In this case, only one of the APIs should be correct in order to output correct command.

Table 6

Google	**W**atson	**C**loud3	**OUTPUT**
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

Possible output combinations for three cloud APIs

According to the presented example of cloud result combination, the CER harvesting multiple cloud platforms (CER_m) in ideal case is defined as:

$$CER_m = \prod_{i=1}^{n} CER_i \tag{4}$$

where CER_i is command error rate of *i*-th cloud platform and *n* is the number of harvested platforms. Here, the condition for proper functioning of the algorithm is that $CER_i > 0$.

4. CONCLUSION

Careful examination of the cloud speech API results enabled us to develop an algorithm which combines interim and final results in order to improve CER. The development of an efficient algorithm should include users' interaction as well as the programmer, which should continuously check for the correctness of interpretation. This is rather inevitable, since we control a physical object, in our case, the wheelchair. According to the theoretical considerations, the combination of multiple cloud platforms should contribute to the improved CER as long as $CER_i > 0$. This is promising for the development of the multiple cloud harvesting algorithms, which will improve the accuracy in many different areas of application.

ACKNOWLEDGEMENT

This research was supported by the Slovenian Research Agency (Programme No. P5-0018 & Proj. No. RU/16-18-040).

REFERENCES

[1] Škraba A., Stojanović R., Zupan A., Kološvari A., Kofjač D., Speech-Controlled Cloud-Based Wheelchair Platform for Disabled Persons. *Microprocessors and Microsystems*, Elsevier, 2015.

[2] Škraba A., Kološvari A., Kofjač D., Stojanović R., Prototype of speech controlled cloud based wheelchair platform for disabled persons. *Embedded Computing (MECO) 3rd Mediterranean Conference on, Budva, Montenegro*. 15-19 June 2014, pp. 162 – 165, 2014.

[3] Škraba A., Kološvari A., Kofjač D., Stojanović R., Wheelchair maneuvering using leap motion controller and cloud based speech control: *Prototype realization. Embedded Computing (MECO) 4th Mediterranean Conference on, Budva, Montenegro. 14-18 June 2015*, pp. 391 – 394, 2015.

[4] Wessel F., Schlüter R., Macherey K., Ney H., Confidence Measures for Large Vocabulary Continuous Speech Recognition. *IEEE Transactions on Speech and Audio Processing*, Vol. 9, No. 3, March 2001, pp. 288 – 298, 2001.

[5] Jiang H., Confidence measures for speech recognition: A survey. *Speech Communication*, pp. 455 – 470, 2005.

[6] Assefi, M., Izurieta, C., Liu, G., & Wittie, M.P., An Experimental Evaluation of Apple Siri and Google Speech Recognition. *Conf. SEDE'15*, 2015.

[7] Twiefel J., Baumann T., Heinrich S., Wermter S., Improving domain-independent cloud-based speech recognition with domain-dependent phonetic post-processing. *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI'14)*. AAAI Press 1529-1535, 2014.

[8] Satyanarayanan M., Bahl P., Caceres R., Davies N. "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14-23, Oct.-Dec. 2009.

[9] Zembrzuski M. et al. (2017) Automatic Speech Recognition Adaptation to the IoT Domain Dialogue System. In: Kryszkiewicz M., Appice A., Ślęzak D., Rybinski H., Skowron A., Raś Z. (eds) Foundations of Intelligent Systems. *ISMIS 2017. Lecture Notes in Computer Science*, vol 10352. Springer, Cham

[10] Traum, D., Georgila, K., Artstein, R., Leuski, A.: Evaluating spoken dialogue processing for time-offset interaction. *Proceedings of the 16th SIGDIAL Conference*, Praha, Czech Republic (2015)

[11] Byambakhishig, E., Tanaka, K., Aihara, R., Nakashika, T., Takiguchi, T., Ariki, Y.: Error correction of automatic speech recognition based on normalized web distance. *Proceedings of the INTERSPEECH 2014*, pp. 2852–2856 (2014)

[12] Sarma, A., Palmer, D.D.: Context-based speech recognition error detection and correction. *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 85–88 (2004)

[13] Razavi, M., Magimai Doss, M.: On recognition of non-native speech using probabilistic lexical model. *Proceedings of the 15th Annual Conference of the International Speech Communication Association*. Interspeech 2014 (2014)

[14] Drexler J., Glass J. 2017. Analysis of audio-visual features for unsupervised speech recognition. *Proc. GLU*.

[15] K. Gupta and D. Gupta, "An analysis on LPC, RASTA and MFCC techniques in Automatic Speech recognition system," 2016 *6th International Conference - Cloud System and Big Data Engineering (Confluence)*, Noida, 2016, pp. 493-497.

[16] Ochi K, Ono N, Miyabe S, Makino S. Multi-talker Speech Recognition Based on Blind Source Separation with Ad hoc Microphone Array Using Smartphones and Cloud Storage. *INTERSPEECH 2016*.

[17] Yazdani R., Segura A., Arnau J. M., Gonzalez A., "An ultra low-power hardware accelerator for automatic speech recognition," 2016 *49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Taipei, 2016, pp. 1-12.

[18] Talib M. R., Hanif M. K., Nabi Z., Sarwar M. U., Ayub N. Text mining of judicial system's corpora via clause elements. *International Journal on Information Technologies & Security*, № 3, 2017

[19] Sharma A. S. and R. Bhalley, "ASR — A real-time speech recognition on portable devices," 2016 *2nd International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Fall)*, Bareilly, 2016, pp. 1-4.

[21] Brester C., Ryzhikov I., Semenkin E. Restart operator for multi-objective genetic algorithms: implementation, choice of control parameters and ways of improvement. *International Journal on Information Technologies & Security*, № 4 (vol. 9), 2017

[21] Nakadai K., T. Mizumoto and K. Nakamura, "Robot-Audition-based Human-Machine Interface for a Car," 2015 IEEE/RSJ *International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, 2015, pp. 6129-6136.

[22] Bhavsar M., P. Kosaraju, G. Ananthakrishnan, G. S. Shet and S. Anand, "Dynamic Improvements in a Cloud-Based Speech Recognition Engine by Incorporating Trending Data," 2016 4th IEEE International *Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, Oxford, 2016, pp. 60-66.

Information about the authors:

Andrej Škraba is a Professor habiltated in the field Support Systems Laboratory at the Faculty of Organizational Sciences. A. Škraba obtained his Ph.D. in the field of Organizational Sciences – Informatics from the University of Maribor. His research interests cover systems theory, modeling and simulation, cyber-physical systems and decision processes. He is a member of System Dynamics Society, INFORMS and SLOSIM

Vladimir Stanovov received the B.S. and M.S. and PhD degrees in system analysis and control from Reshetnev Siberian State Aerospace University, Krasnoyarsk, Russia, in 2012, 2014 and 2016, respectively. His research interests include genetic fuzzy systems, self-configured evolutionary algorithms and machine learning. He is currently a PhD student at the Siberian State Aerospace University. Mr. Stanovov received the Best Student Paper Award from the 4th International Congress on Advanced Applied Informatics in 2015.

Eugene Semenkin is a Professor at the Department of System Analysis and Operations Research, Siberian State University of Science and Technologies. He received his PhD in Computer Science from Leningrad State University (Leningrad, USSR) in 1989 and his Dr. Sc. In Engineering and Habilitation from the Siberian State Aerospace University (Krasnoyarsk, Russia) in 1997. His areas of research include the modelling and optimization of complex systems, computational intelligence and data mining.

Andrej Kološvari obtained his M.Sc. in the field of Organizational Sciences – Informatics from the University of Maribor. He has extensive industrial experience in the field or measurement, monitoring and control from Iskra Kibernetika Inc. He is currently a Ph.D. student at the University of Maribor, Faculty of Organizational Sciences working on his thesis in the field of Cyber-Physical systems in education

Davorin Kofjač received his Ph.D. from the University of Maribor in the field of Management of Information Systems. He has published several papers in international conferences and journals, and has been involved in many national research projects. Currently, he is working as a researcher at the same university in the Cybernetics & Decision Support Systems Laboratory. His research interests include modeling and simulation, artificial intelligence and operational research.

Manuscript received on 20 January 2018