Development of a Novel Conversational Calculator Based on Remote Online Computation

Xiaohua Liu, Haoran Liang, Haiwei Dong*, and Nikolaos Mavridis

Department of Computer Engineering, New York University Abu Dhabi, Abu Dhabi, UAE
{xiaohua.liu,haoran.liang,haiwei.dong,nikolaos.mavridis}@nyu.edu

Abstract. This paper presents a way to improve the current situation by introducing a newly developed computational tool that is capable of speech recognition and voice output and includes a keypad-free user interface, resembling a Smartphone message box, to enhance human-machine interaction experience. Furthermore, it relies on effectively offloading computation to a remote online service. The main ideas behind this paper are the possibility of transforming tangible daily tools into specially designed interface agents that enable voice communication with users, and the possibility of utilizing available online information databases and well as online services that rely on remote machines instead of utilizing local computation. An important novelty of the presented work is also the fact that it was designed on the basis of empirical data arising from an appropriate Wizard-of-Oz-like experiment that was performed with almost one hundred people, and thus high-quality recognition of commonly occurring natural language queries was achieved.

Keywords: Human Computer Interface, Spoken Dialogue System, Language Model Design, Online Services.

1 Introduction

A traditional electronic calculator is a small electronic device used to perform the basic operations of arithmetic. In general, a basic electronic calculator consists of a power source, a keypad and a processor chip. Normal electronic calculators are capable of simple calculation such as addition and multiplication. However, the usability of a calculator is minimized, when the hands of their user are occupied or inconvenient to use. In daily life, calculators are mainly used to solve simple calculations, such as cost calculations or money conversions, in contrast to highly complex scientific problems. For all of the above reasons, and given avenues that have opened from recent advances in the state-of-the-art, one can come to the realization that the keypad of a calculator might indeed be substitutable, and human voice could be an alternative means of input as opposed to keypad.

There exists a wealth of existing research in *speech recognition* [1], yet still speech-enabled natural-language interfaces have not been deployed very widely, with

* Corresponding author.

T. Huang et al. (Eds.): ICONIP 2012, Part I, LNCS 7663, pp. 142–151, 2012.

the exception of specific application domains, such as automated call centers. Recently, though, speech recognition technology is expected to become more widely available to the average non-expert user, through products such as the Apple Siri. On the other hand, regarding *computational engines* that can accept natural language queries, there have recently been important developments, with Wolfram Alpha being a prime example, being able to provide answers to a wide range of natural language queries, including difficult symbolic math problems, such a symbolic integration etc. Human-Computer-Interaction is a large field with considerable history [2], [3], and multimodal interfaces have existed for quite a while – here he are presenting an empirical-data-driven tailored speech interface for an everyday usage calculator, which utilizes remote online computation through Wolfram Alpha as its computational engine.

But in order to make our system usable in the real world, we need to have sufficient accuracy in our speech recognition, as well as the sufficient comprehension by the language understanding module (which resides inside Wolfram Alpha in our case). The performance of a speech recognition system is usually determined by its accuracy. The accuracy can be affected by different factors, such as vocabulary size [4] and confusability [5].Situations can be that as the vocabulary size grows, the accuracy decreases, and that if the words are confusable, it is difficult for the machine to recognize. Besides vocabulary size, grammar can also influence the accuracy. Two widely used grammar-encoding formats are used in speech recognition are ABNF, and grammars encoded in XML. ABNF denotes Augmented Backus-Naur Form, and XML denotes eXtensible Markup Language. They are semantically mappable to allow automatic transformation between the two, and both representations essentially express a Context-Free Grammar (CFG) [6].The key to successful speech recognition in our system, is based on the fact that not all possibly words and phrases are usually spoken to a conversational calculator; thus, if we can estimate a suitable probabilistic grammar for what we expect our calculator to hear, we minimize the probabilities of misrecognition, as our effective hypothesis space becomes much smaller. In order to estimate this grammar, we performed a special wizard-of-oz experiment, where we asked people to ask the "calculator" whatever they want. Thus, through the statistical characteristics of the utterances of people in this context, we were able to derive a special grammar (language model) for our conversational calculator, which proved to be quite powerful and which significantly increased recognition accuracy within this context (language addressed towards a calculator). Thus, in comparison for example to general-purpose voice recognition systems, the vocabulary size of our calculator is much smaller, and our speech recognition is more accurate and specific. Besides, many calculators lack several functions that we equipped our program with, such as abilities for multi-step calculations, real-time updateable currency conversions etc.

In short, through this paper, we also illustrate how an intelligent conversational computational tool can be designed to utilize vast resources both on personal computers and on the Internet, which other calculator programs hardly make use of [7]. Also, we present our empirical findings regarding a language model which corresponds to the probabilistic grammar of the utterances that people would like to say to a conversational calculator, derived through an extensive experiment with almost one hundred participants. We furthermore illustrate the effectiveness of this grammar towards

accurate speech recognition in the conversational calculator domain. Compared to traditional calculators that are pre-designed with specific functions and algorithms built in, our calculator makes use of the remote online Wolfram Alpha computational engine. Moreover, and our calculator is aimed at daily computation as opposed to mathematical education. Last but not least, our calculator can also serve as a helpful tool for users with visual or tactile special needs.

2 Method

Our method to build an intelligent conversational calculator includes three phases: a Wizard-of-Oz experiment, product design and product realization.

2.1 Experimentation

We conducted a Wizard-of-Oz experiment, of which the goal is to investigate human-computer interaction with a conversational calculator, and derive a context-specific language model [8].

During the experiment, we let a participant ask the computer any computational question he or she might think of. The question was recorded and sent to another computer, which one of our team members behind scene operated and gave the answers back accordingly via the computer. The purpose was to simulate human-computer interaction so that we could gather information about what people actually wanted to ask our program.

We managed to invite 100 people to our experiment, and we recorded their demography in an excel file and their questions and our responses in a separate log file (Fig.1). There were 68 males and32 females in our sample, the age range was 18 to 27 while the average age was 20.23, 99% of the respondents were college students, and we effectively covered 33 countries – with most participants coming from United Stated (24), China (14) and Ethiopia (6), while we also had participants from Mexico, Hungary, Korea, India, Britain, Australia and so on. Only 29% were native speakers of English; but almost all had TOEFL scores equal or higher than those required for enrolling in an English-Language University.

To analyze our results, we then used the CMU-Cambridge Statistical Language Modeling toolkit, a suite of UNIX software tools to facilitate the construction and testing of statistical language models [9].We first converted our text stream, including all questions and answers, into a list of vocabulary consisting of all words in our log file along with their numbers of occurrences using the text2wfreq tool inside the CMU-Cambridge Statistical Language Modeling toolkit. By using the text2idngram tool inside the toolkit, we then combined the text stream and the vocabulary list to construct an id 3-gram file, which was an ASCII file containing a numerically sorted list of 3-tuples of numbers, corresponding to the mapping of the word 3-gram relative to the vocabulary. Finally, we built a language model in ARPA format by using idngram2lm tool and got a 3-gram language model based on a vocabulary of 136 words.

2.2 Functionality Design

We analyzed the results from the Wizard-of-Oz experiment and define the functionality of the calculator.

Following the motivation to build an accurate calculator, we took the ARPA-format language model and studied the probability of all expressions. The probability was defined in the following manner:

Trigram:

$$\begin{aligned}&\mathrm{Prob}(w3 \mid w1, w2)\\ &= \begin{cases} \mathrm{Prob}\big(trigram(w1, w2, w3)\big) & \exists\ trigram(w1, w2, w3)\\ \mathrm{Prob}(w3 \mid w2) \cdot \text{Back-Off-}\mathrm{Prob}(w1, w2) & \exists\, bigram(w1, w2)\\ \mathrm{Prob}(w3 \mid w2) & otherwise \end{cases}\end{aligned} \tag{1}$$

Bigram:

$$\begin{aligned}&\mathrm{Prob}(w2 \mid w1)\\ &= \begin{cases} \mathrm{Prob}\big(bigram(w1, w2)\big) & \exists\ bigram(w1, w2)\\ \mathrm{Prob}(w2) \cdot \text{Back-Off-}\mathrm{Prob}(w1) & otherwise \end{cases}\end{aligned} \tag{2}$$

where $\mathrm{Prob}(\cdot)$ denotes probability [10].

We got rid of the uncommon expressions with a probability cutoff of -0.1 (in log10 scale). For instance, the 3-gram expression "the partial derivative" had a probability of -0.1761 on a log10 scale. With engineering experience, we decided to discard such an expression based on its low probability. We then further adjusted our language model by adding and deleting expression according to our common sense. For example, the number 19 happened not to appear in our log file. We thought that it was due to a probability issue and that in fact the number 19 is as likely to appear as any other number in its neighborhood. As a result, we added the number 19 to our language model and determined its probability based on the probabilities of the numbers 18 and 20.

Also, we analyzed the questions and comments from the participants in the Wizard-of-Oz experiment. Since few participants asked our program to do integrals, for example, we decided to make our program capable of doing questions on differentiation but not on integration. Based on the comments from the participants, we also decided that our program needed to have memory features so that multi-step calculation was possible. Other functions, for example, include unit and currency conversion. At the end of this stage, we decided on our language model and the functional features of our program.

2.3 Product Realization

To build the program, we split it into three processing sections: speech recognition, central computation and speech synthesis (Fig.1).

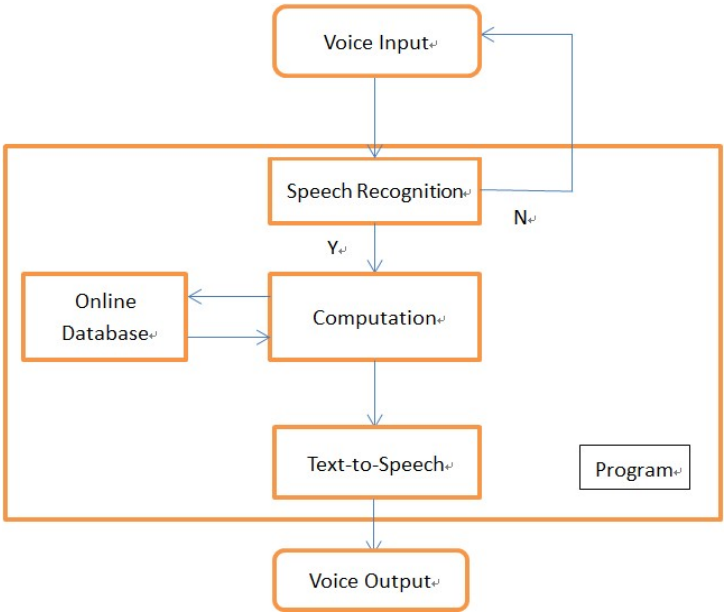

Fig. 1. Flow chart of the program

First, we used the speech recognition engine in Windows to convert the voice input into recordable text. To make this possible, we designed a grammar file in the XML form. Inside our XML grammar file, we used four types of common grammars (Table 1). Within each grammar, every element consisted of several categories. Every category contained a certain number of vocabularies (Table 2). We assigned to all expressions probabilities proportional to the results from our adjusted language model.

Table 1. Repeat times of individual categories inside four types of grammars

	Expression	Number	Action	Unit
Expression+Number+Action+Number+Action	0-1	1-6	1-3, 0-3	N/A
Expression+Action+Number+Action	0-1	1-6	1-3, 0-3	N/A
Expression+Number+Action	0-1	1-6	1-3	N/A
Expression+Number+Unit+Unit	0-1	1-6	N/A	1

Table 1 presents four different types of common grammars we built inside our XML file. Every grammar consisted of different categories. In our file, certain combinations and sequences of the categories made up four different grammars. Each category was composed of a certain number of vocabularies. Meanwhile, each word in the category was assigned a value, called a weight, which represented the probability that the word would arise during conversation. These values were determined by the result of Wizard-of-Oz experiment.

Examples: 1. (what is) twenty (two) plus thirty (one) (squared) 2. (calculate) the square root of twenty (five) 3. (what is) forty (two) cubed 4. (what is) eighty (five) dollars in dirham

Table 2. Vocabularies of four categories inside the grammar file

Categories	Vocabularies
Expression	what is, what is the value of, how much is, calculate
Number	pi, negative, half, third, fourth, fifth, sixth, seventh, eighth, ninth, tenth, zero, one, two, three, four, five, six, seven, eight, nine, ten, eleven, twelve, thirteen, fourteen, fifteen, sixteen, seventeen, eighteen, nineteen, twenty, thirty, forty, fifty, sixty, seventy, eighty, ninety, hundred, thousand, million, billion, answer, point, x
Action	plus, minus, multiplied by, divided by, times, mod, the sine of, the cosine of, the tangent of, the square root of, squared, the cube root of, cubed, to the power of, the derivative of
Unit	Dollars, Pounds, Euros, Dirhams, degrees, radians, kilometers, meters, miles, feet, kilometers per hour, meters per second, grams per centimeter, kilograms per liter

To realize the speech recognition, we used the following startrecognizer() method in C# programming language. Inside recognizer_SpeechRecognized(object sender, SpeechRecognizedEventArgs e), we appended every recognized word to our question recorded for submission to Wolfram Alpha.

Once the voice input was converted into recordable text, we sent the request to Wolfram Alpha. We conducted a series of research on the mechanism of Wolfram Alpha online calculation. Noticing the patterns of Uniform Resources Locators (URLs) of Wolfram Alpha, we simply appended the recorded text to the URL http://www.wolframalpha.com/input/?i=.

We analyzed the response from the website by carefully examining the source code and located our desired result. Then we checked the patterns of Wolfram Alpha source code and found out that each result box corresponds to a specific sequence of characteristic expressions. We thus located our desired result by first finding that expression and then trimming out unnecessary information by the substring() method.

Finally, we fetched the desired result and recorded it in our program. Our search algorithm, however, did not always give the correct result. The above code only

looked for answers in the second result box, which in reality does not always give the most needed answer. As a result, we kept improving our algorithm by searching for other result boxes and adding more special-case solutions for particular expressions (Fig.2). For example, for trigonometry questions, we searched the third result box instead of the second one.

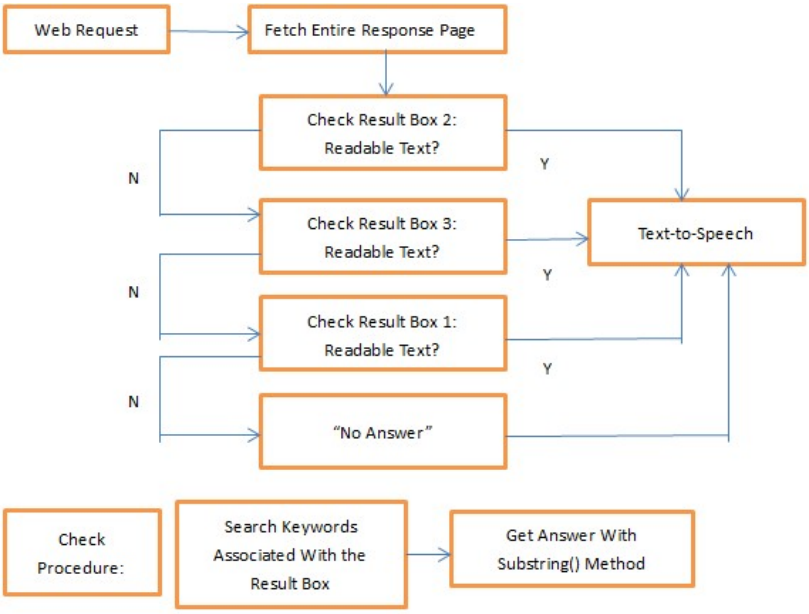

Fig. 2. Computational Engine output information processing and Search algorithm

Once we had the result recorded, we used Windows Speech Synthesizer to speak out the result to the user.

3 Results

We successfully conducted the Wizard-of-Oz experiment and built a language model solely designed for later use in our computational program. We then use the results to build the conversational calculator program that was able to understand the voice input about computational questions and to give voice output of the correct answers, and finally we carried out a user test to evaluate performance.

We designed a user interface that resembled the message box of a Smartphone in order to realize our motivation to provide users with a simple, friendly and helpful calculator that enabled an intelligent conversation about daily computational problems (Fig.3).

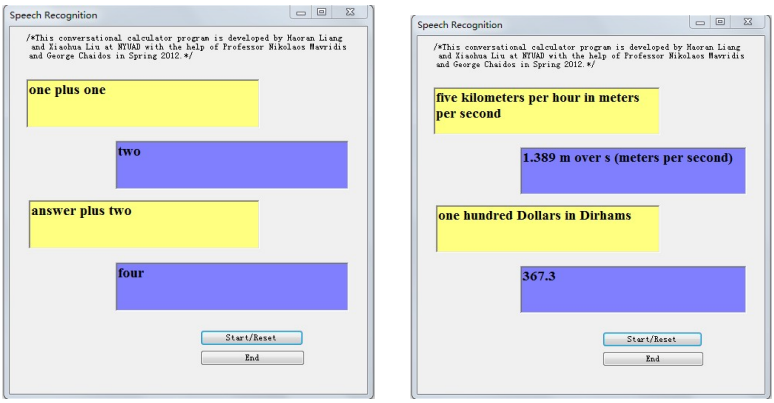

Fig. 3. Demonstration program showing the multi-step calculation and unit/currency conversion features

For the purpose of evaluating system performance and receiving helpful feedback, we made a questionnaire and randomly delivered it to 76 participants. We designed another experiment, in which our program gave answers to various questions from the participants in our experiment. Each participant tried our demonstration program with questions he/she would like to ask, and then gave evaluation to the trial. Meanwhile, we recorded our program performance, either accurate or inaccurate. The results turned out that 193 out of 284 questions asked by 76 participants were accurately answered by our program, approximately 67.96% accuracy. 58 out of 76 participants gave "good" comments to the trial experience (Table 3).

Table 3. Feedback surveyforourdemonstrationprogram

	NumberofParticipants	Percentage
Good	58	76.3%
Okay	15	19.7%
Poor	3	4.0%
Total	76	100%

During the process, we encountered several problems. We first programmed in JAVA language and tried CMU Sphinx 4 for speech recognition. However, the speed and accuracy was not satisfactory enough. As a result, we switched to Windows Speech Recognizer, and the speed and accuracy improved by more than 20%. Meanwhile, since we wrote the grammar file in the BNF format, which CMU Sphinx 4 could read directly, we had to change the grammar file into XML format so that Windows Speech System could read more easily. To decide the repeat times of each category of expression in our grammar file, we actually had to make assumptions based on our Wizard-of-Oz experiment result and our own experience. Chances were that people would ask questions that did not fall under any type of our grammars. In this case, the computer would not recognize the question, and the computation process

would not start. We kept improving our grammar so that it could have a more powerful speech recognition capacity. When deciding vocabularies in each category, we also used common sense to manually adjust the dictionary even after the first round of elimination based on -0.1 probability cutoff. We considered bigrams and trigrams with high probabilities as individual expression in our vocabulary. We assigned them weights that were generally proportional to those inside the language model but made round-off changes for easier programming purposes.

We managed to hand the computation over to the Wolfram Alpha computational engine and to retrieve correct information from there by using our search algorithm. Most of time we got the desired answers, but there were quite a few times we did not get what we expected. For example, when we did currency conversion between US Dollars and UAE Dirhams, the answers included Arabic language, which the Windows Speech Synthesizer could not recognize. As a result, we added into our search algorithm a special case operation to retrieve only the numerical values.

4 Conclusion

We have built a prototype of an intelligent conversational calculator designed on the basis of an empirically derived language model through a Wizard-of-Oz experiment, and utilizing the combination of voice recognition, Text-to-Speech, and a remote online computational engine, namely Wolfram Alpha. Furthermore, we have evaluated our system through a user test. The Wizard-of-Oz experiment aimed to collect data, information and suggestion from participants, and its result was used to create and improve the language model of our system. We carefully designed the language model through the analysis of the experiment data, providing generalizations where required, in order to obtain an appropriate vocabulary, which is general enough for our purpose, but small enough so that recognition accuracy remains high. Our central computational engine, Wolfram Alpha, which was accessed as a remote online service, performed language understanding and question answering towards our purpose. Our system has many advantages over other approaches, such as for connecting Siri to the Apple calculator: for example, we have much better recognition accuracy through the context-specific language model, and greater functionality. Also, if one had tried to connect directly to Wolfram Alpha without appropriate post-processing, then the result would be a list of all possible results instead of speaking out the best answer directly to the user. Our prototype, instead, is capable of selecting the most appropriate result sent back by Wolfram Alpha, and converting into speech. So far, our program has been improved with multi-step calculation, unit/currency conversion and background processing features. With further development, our prototype can be modified to a highly intelligent conversational calculator, which might also serve as a component of larger systems.

Most importantly, through our approach, we have illustrated the power of empirically-derived context-specific language models, and derived, presented, and evaluated such a model for the context of conversational calculators. We have furthermore illustrated the power of remote online computation in building real-world tools with natu-

ral and fluid interfaces that can simplify tasks and assist humans. Last but least, we foresee that with the further advancement of online services and cloud technologies, such tools will be able to utilize a multitude of interconnected online services in the future, and exhibit much greater levels of intelligence and transparency-in-use, becoming an integral part of our everyday lives.

Acknowledgment. The authors would like to thank George Chaidos for his great help in coding, and to thank Leonard Helmrich and Mo Ogrodnik for their huge support.

References

1. Varile, G.B., Zampolli, A.: Survey of the State of the Art in Human Language Technology. Cambridge University Press, Cambridge (1997)
2. Myers, B.: A Brief History of Human-Computer Interaction Technology. Interact 5, 44–54 (1998)
3. Baecker, R.M., Grudin, J., Buxton, W.A.S., Greenberg, S.: Readings in Human-Computer Interaction. Morgan Kaufmann (1995)
4. Casali, S.P., Williges, B.H., Dryden, R.D.: Effects Recognition Accuracy and Vocabulary Size of a Speech Recognition System on Task Performance and User Acceptance. Journal of Human Factors 32, 183–196 (1990)
5. Hernandez-Abrego, G., Olorenshaw, L., Tato, R., Schaaf, T.: Dictionary Refinemets Based on Phonetic Consensus and Non-uniform Pronunciation Reduction. In: 8th International Conference on Spoken Language Processing, pp. 1697–1700. IEEE Press, New York (2004)
6. Schwartz, E.: Speech Recognition Grammar Specification. W3C Recommendation (2004)
7. Karat, C.M., Vergo, J., Nahamoo, D.: Conversational Interface Technologies. In: The Human-Computer Interaction Handbook, pp. 169–186. Lawrence Erlbaum Associates (2003)
8. Bernsen, N.O., Dybkjaer, H., Dybkjaer, L.: Designing Interactive Speech Systems – From First Idea to User Testing. Springer (1998)
9. Clarkson, P., Rosenfeld, R.: Statistical Language Modeling Using the CMU-Cambridge Toolkit. In: 5th European Conference on Speech Communication and Technology, pp. 2707–2710 (1997)
10. Katz, S.M.: Estimation of Probability from Sparse Data for the Language Model Component of a Speech Recognizer. IEEE Transactions on Acoustics, Speech, and Signal Processing 35, 400–401 (1987)