Wise home non-touch control. Brain-computer interface and fuzzy voice input as an advanced voice input handler

Zuev E. (Russian Federation)

Бессенсорный Умный дом. BCI и нечеткие голосовые команды как эффективный способ управления голосом

Зуев Е. Д. (Российская Федерация)

Зуев Егор Дмитриевич / Zuev Egor Dmitrievich – студент,
кафедра информационных и телекоммуникационных технологий,
Московский авиационный институт (национальный исследовательский университет)
г. Москва

Abstract: *this paper outlines the way in which human can interact with wise things. It looks at different approaches of implementing the partial integration of human brain with smart playground; the ways of processing the voice input; fuzzy logic in text analyzing and non-strict commands input; what methods could be used to achieve full non-touch control.*

Аннотация: *в данной статье рассматривается возможность управления умными вещами через оператора. Здесь будут рассмотрены различные методы и подходы к частичной интеграции мозга оператора с умной площадкой; методы обработки голосовой информации; ленивые вычисления в анализе текста и нечетких голосовых команд; какие подходы позволят достичь полного бессенсорного контроля.*

Keywords*: levenshtein distance, Naïve Bayes, Wise home, automatization, voice input, phoneme – grapheme, EEG, neurosky, brain waves, neurology, R language, fuzzy logic, machine learning, wise things, classification.*

Ключевые слова: *дистанция Левенштейна, наивный Байес, умный дом, автоматизация, голосовой ввод, фонема-графема, ЭЭГ, нейроскай, мозговые волны, нейрология, язык R, ленивые вычисления, машинное обучение, умные вещи, классификация.*

1. INTRODUCTION

Today, building a non-touch control wise home system is a task of many years of effort. Thanks to the latest breakthrough in neurology and data mining, new approaches appeared which can cope with this problem.

In this research, we have introduced a brand new way, in which wise home can be partially integrated with human brain using latest technologies in a field of brain-computer interface (BCI) and latest semantic recognition approaches based on fuzzy logic.

The proposed model consists of two main parts – voice input and BCI input. Here, BCI is used as a trigger, which handles voice input. Voice input is divided into two step model, in order to reduce errors, which could happen during the recognition process.

The rest of the paper is organized as follows: Section 2

gives a general idea of wise home (playground) in conception of our model. Section 3 dives into a problem of controlling wise things with non-strict commands based on voice input. Section 4 describes a part of proposed model – semantic processing and recognition of non-strict voice commands. Section 5 introduce the second part of our model - an idea of using BCI in conception of our entire system and some integration notes.

2. WISE PLAYGROUND

By a «wise playground», we mean a mechanism, which represents a many-to-one interface for most smart things (Of course interface must not be abstract and have realization). The idea of embedding all things in ‘one point’ originally came from the rise of the internet and web2.0.It’s also known as IoT (internet-of-things).

However, in original conception, there is no semantic rule or filter presented, and the main aim of such ground is nothing than exchange data between things and do simple automatization. Moreover, nowadays there is no point in such computer-based system, as most processes seem to be difficult to implement in a non-strict way. By saying “non-strict,” I mean flexible input and control. As a result, an operator must trigger some event-driven actions with exact command directly or thanks to special algorithms, which implements some self-learning based on operator input. It is also known as neural networks or expert systems (depends on the difficulty of the problem). General solution scheme is presented below (Fig.1)

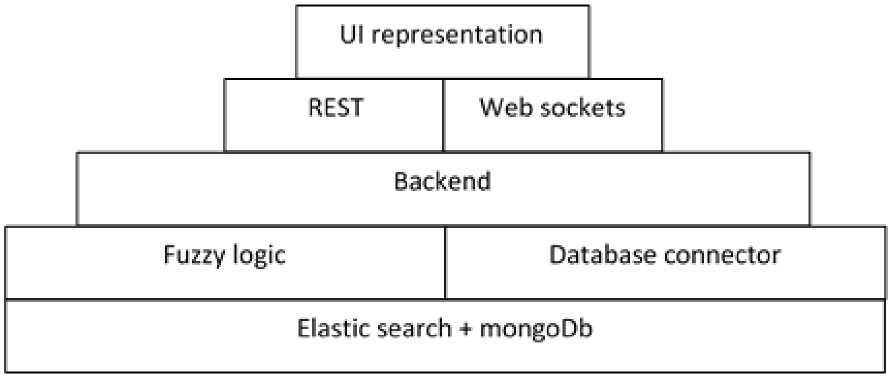

Fig. 1. General scheme of solution

3. SEMANTIC PROBLEMS IN "WISE HOME CONTROLLING CONCEPTION"

It is not a secret, that modern approach in controlling home is based on some non-strict input. For instance, it can be virtual reality, neuro interface, or voice input. Right now, the virtual reality seems to be the best choice. However, right now you need a special device to do such stuff. Therefore, using voice does not require any special interface.

As you may have figured out, there are several problems with voice input. The first one is converting the voice record to text. The second one is semantic rules. The problem is that all our things have special commands which trigger a specific event (For example command "toggle" will trigger lamp).Therefore, your voice input must be exact to command name. And the last one is to recognize the command in a general speech. In further chapters, we will discuss this issues and methods to come up with it.

Fuzzy logic in text recognition

Before we will talk about text processing itself, it is also important to understand the origin of characters manipulation. So, every character in the word is presented as a binary number system. For instance, the word "ride" in the binary system is 01110010 01101001 01100100 01100101. Then, if you try to equal this word with word "bird", the computer will check this equality in the binary system. It is important to understand the behavior of word storing and representation, as you already know, that all words are processed as equality. But what, if I do not want to use a strict word in my recognition system? For solving this problem, we will use fuzzy logic.

Fuzzy logic is an analytic approach that extends classic Boolean algebra. According to its rules, it can handle the partial truth, where the result is presented as a percentage. For instance, if I have 1100 and 1010 in my dictionary, I want to know the correlation between this dictionary and 1011. The system result may be 0.16 for 1100 and 0.84 for 1011.

As you see, it is a rather flexible system. However, more difficult word sequences (or just sentences) in dictionary – more problems in getting good results. For this purpose, I am using in my system only single words and phraseologisms. Therefore, it seems as a piece of cake to cope with such data. But every thing may have several commands and what if there are more than 10 things? Remember command for every trigger for every thing seems to be a great problem. To resolve this issue, we will use synonyms for every command and classification algorithm.

Some notes on classification

A classification is an analysis approach that consists of predicting a certain outcome based on a given input. In order to predict the outcome, the algorithm processes a training set containing a set of attributes and the respective outcome, usually called goal or prediction attribute. The algorithm tries to discover relationships between the attributes that would make it possible to predict the outcome. Next, the algorithm is given a data set not seen before, called prediction set, which contains the same set of attributes, except for the prediction attribute – not yet known. The algorithm analyses the input and produces a prediction. The prediction accuracy defines how "good" the algorithm is. The sample from the previous chapter nicely covers this topic.

Choosing algorithm

There are two approaches that could be useful in solving a classification problem in our case: the first one is Naïve Bayes, and the second one is Levenshtein distance.

Naïve Bayes itself represent a probabilistic model. The general type of Bayes model (1) is:

$$p(C_k|x) = \frac{p(C_k)\,p(x|C_k)}{p(x)} \qquad (1)$$

While Levenshtein (2) is a math approach which search distance between two strings. The general form is: (of length $|a|$ and $|b|$ respectively) is given by $Lev_{a,b}(|a|,|b|)$ where

$$(2) \qquad Lev_{a,b}(i,j) = \begin{cases} \max(i,j) \\ \min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_i)} \end{cases} \end{cases}$$

Where $1_{(a_i \neq b_i)}$ is the indicator function equal to 0 when $a_i = b_j$and equal to 1 otherwise, and $Lev_{a,b}(i,j)$ is the distance between the first i characters of a and the first j characters of b.

Let us check which method would better fit our problem. We will do this stuff by building up simple models with this two methods using R language.

```
install.packages('e1071', dependencies = TRUE)
install.packages('RTextTools', dependencies = TRUE)
install.packages('vwr', dependencies = TRUE)

library(e1071)
library(RTextTools)
library(vwr)

#feed with patterns
ride_data =  rbind(c('ride', 'ride'), c('beride', 'ride'), c('rider', 'ride') )
raid_data = rbind(c('turn', 'turn'), c('turner', 'turn'), c('turn on', 'turn') )

data = rbind(ride_data, raid_data)

matrix= create_matrix(data[,1], language="english",
removeStopwords=FALSE, removeNumbers=TRUE,  # we can also removeSparseTerms
stemWords=FALSE)


# train the model
mat = as.matrix(matrix)
classifier = naiveBayes(mat[1:6,], as.factor(data[1:6,2]) )


# test the validity
test_case <- c('prize', 'ride')
predicted = predict(classifier, test_case); predicted
table(test_case, predicted)
recall_accuracy(test_case, predicted)

####levenshtein#####
lev_perc <- levenshtein.distance(test_case[1], data)
lev_perc <- 1 - lev_perc / 7
lev_perc
#get template name
data[which(lev_perc == max(lev_perc))[[1]] ,2]
#get accuracy
max(lev_perc)[1]
```

The output will be the following:

```
> recall_accuracy(test_case, predicted)
[1] 0.5

#get template name
[1] "ride"
> max(lev_perc)[1]#get accuracy
[1] 0.7142857
```

The first one is Naïve Bayes – as you see, the less dictionary, the more inconvenienced results.

The second – is Levenshtein. According to our algorithm, we are checking our origin input with every string in the dictionary and then get the most accuracy pattern.

The conclusion is simple: Bayes is good in the case when we have an input, which is close to origin pattern. Furthermore, in our example, we use a word, which have a similar part to first origin pattern ("ride"), and beside this, the result is 0.5 and this is the unsatisfying result. Levenshtein instead shows the better result, which I supposed to get from the very beginning. So, for our purpose, Levenshtein seems to be the best choice.

4. VOICE INPUT PROCESSING

As we have already pointed out, voice recognition in some way may give an accurate result due some reasons: accent, noise, several voice sources and etc. Therefore, there is no point in recognition of command directly. As some triggers of different things may be similar or even equal, it is suggested to choose a thing first, and after a trigger for command of chosen thing. Therefore, there would be a two-step voice input (Fig. 2). For first input, we need a strict phoneme-based dictionary, which will help us to reduce output results. Let us assume that every phoneme item in our dictionary is a thing's name. As for the second step – there would be a general google voice processing system. You may be interested, why we will not use the same idea, as in the first step. So, you command for a trigger may be a custom word, which goes out of general semantic rules. As a result, there is no phoneme rule. In addition, there is no general algorithm for processing word to phoneme (right now, thing's name parsed to phoneme from prepared static dictionary, as a number of supported vendors is strict, such dictionary is a useful component in our system).

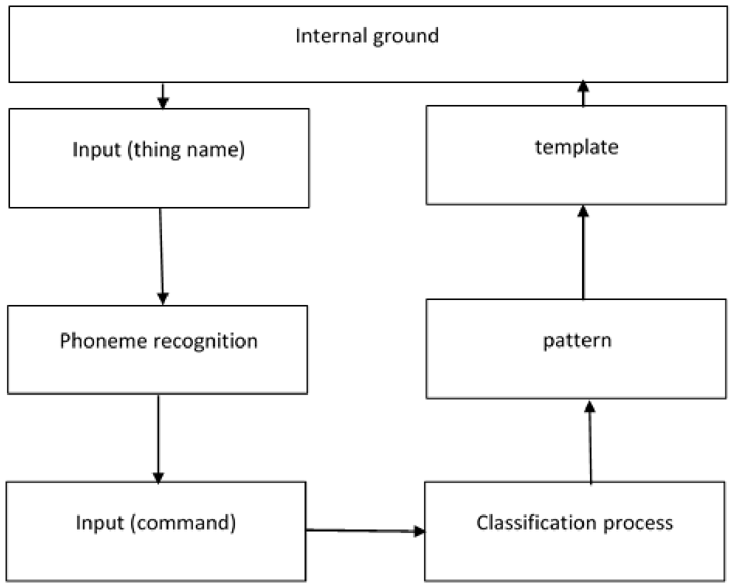

Fig. 2. Two step voice input

Word-phoneme based speech recognition

Natural language processing is a complex problem, which involves different approaches, methods and methodologies to discover. In our deals, we do not need natural processing as we split system to two steps – command for choosing thing (which is strict) and second for the command for chosen thing in the previous step.

There are two approaches on which I paid attention mostly. The first one – is a mapping between real physical word and my pattern. The goal is clear – we have preloaded dictionary, so there is no need to implement any services for converting waves to text. The problem is that it have n-templates for our unique pattern, which means that results of your input will be decreased in as 1/n, where n – amount of templates for your pattern. As you see, the more difficult your keyword, the fewer chance you get the right result. Nevertheless, the second approach is more flexible as it uses phoneme data to construct the pattern. It is great, as we do not have any bind to the real word, which gives us the ability to use acronyms (custom words, which do not exist in a real dictionary). Thus, a text-to-speech approach using phones provides flexibility but cannot produce intelligible and natural speech, while a word level concatenation produces intelligible and natural speech but is not flexible. In order to balance between flexibility and intelligibility/naturalness, subword units such as diphones which capture essential coarticulation between adjacent phones are used as suitable units in a text-to-speech system.

Given the sequence of words, the next step is to generate a sequence of phones. For languages such as English where the relationship between the orthography and pronunciation is complex, a standard pronunciation dictionary such as CMU-DICT is used. To handle unseen words, a grapheme-to-phoneme generator is built using machine learning techniques [Black et al., 1998].

Fetch template by phoneme recognition

Here we are going to trigger events in a fluent style. In other words, next function (or step) will be executed as soon as previous will end.

The following block-scheme below (Fig. 3) will describe the logic of our code (which is present above)

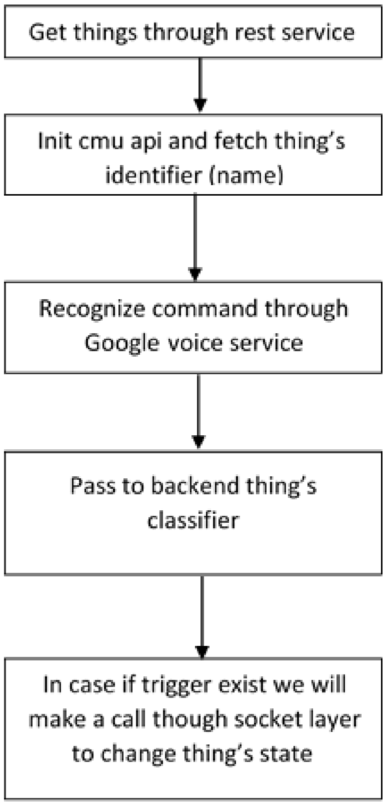
Get things through rest service
Init cmu api and fetch thing's identifier (name)
Recognize command through Google voice service
Pass to backend thing's classifier
In case if trigger exist we will make a call though socket layer to change thing's state

Fig. 3. Phoneme recognition

5. NEURO INTERFACE, BARIN WAVES AND EEG

In the second chapter, we have already mentioned about neuro interface. In our conception, neuro interface is an EEG device, which will handle the certain state of an operator's brain for the initialization of voice input purpose. Thanks to latest technologies, EEG has become available to the general audience. Don't expect of recognition your thoughts. Right now, it's impossible, because most artefacts are unique, and others depend on a specific brain. In Neurology, the term "artefacts" means a set of special algorithms, where main arguments are states of brain waves (alpha, beta, gamma, theta, delta). Nevertheless, there are few artefacts, which are general for most humanity. We will use some of them to handle an input event. For instance, when I am concentrating, the voice interface appear. This let us avoid sensor control, and for some social groups, it could be the only chance to control the wise home.

For my aim, I have chosen the neurosky device. The main reason is that it have an only frontal sensor and it has a general headsets size. The idea lies behind a mobility, which means, that there is no need in special equipment, nor in the special gel for sensors. Under the hood, most dev companies pay most attention to device's size, as there is no need in obtaining full brain map. So, working on the field of recognition using only frontal sensor seems to be demanded.

eSence notes

Neurosky provides their concentration and meditation algorithms out of the box. This technology is known as eSence. In this project, we use special settings for detecting a trigger event, so you need manually set concentration and meditation levels.

6. CONCLUSIONS

In this paper, we proposed a new modeling strategy based on voice input and EEG trigger like system. The proposed method solves problems with controlling wise home via voice and introduce new approaches in integration brain-computer interface with wise things. Also, we have outlined problems in fuzzy logic recognition voice systems in context of BCI systems, which haven't been discussed before. Furthermore, we have done some research in a field of semantic and phoneme-grapheme voice-pattern recognition. For future work, we plan to extend our work by modeling and predicting some EEG artefacts, which are responsible for triggering voice controller in our complex system. We will do some research in a field of using brain waves and issues, which may appear due recognition process (such as building deep neural network, finding general artefacts and adopting neural network for different users).

Finally, I would like to admit, semantics and voice control are two difficult problems, which involve not only «strict» algorithms and certain answer, but also a fuzzy logic, or even a complex expert system. Thereby, in future research, it is planning to pay most attention to pos-tagging, natural language processing and Neurology (especial artefacts).

References

1. *Hwang M-Y*: Subphonetic Acoustic Modeling for Speaker-Independent Continuous
2. Speech Recognition. Ph.D. thesis, Carnegie Mellon University, 1993.
3. *Hieronymus* L., J.: ASCII Phonetic Symbols for World's Languages: worldbet. Technical report, Bell Labs, 1993.
4. *Clarkson P., and Rosenfeld R.:* Statistical Language Modeling Using the CMUCambridge Toolkit. In the proceedings of Eurospeech, Rodhes, Greece, 1997, 2707 – 2710.
5. *Abu-Mostafa Y. S.* (1990). Learning from hints in neural networks, J. Complexity 6, 192–198.
6. *Akaike H.* (1970). Statistical predictor identification, Ann. Inst. Statist. Math. 22, 203–217.
7. *Angel J. R. P., Wizinowich P., M. Lloyd-Hart, and D. Sandler* (1990). Adaptive optics for array telescopes using neural–network techniques, Nature 348, 221–224.
8. *Baum E. B.* (1991). Neural net algorithms that learn in polynomial time from examples and queries, IEEE Trans. on neural networks 2 1, 5–19.
9. *Bayes T.* (1763). An essay towards solving a problem in the doctrine of chances, Philos.Trans. R. Soc. London 53, 370–418, reprinted in Biometrika (1958) 45, 293–315.
10. *Haykin S.* Neural networks: A comprehensive foundation. Prentice-Hall Inc., New Jersey, 1999. 97, 98
11. *Hermansky H. and Malayath N.* Speaker verification using speaker-specific mappings. In Speaker Recognition and its Commercial and Forensic Applications, France, 1998. 72
12. *Huang X. D., Acero A, and Hon H-W.* Spoken Language Processing: A Guide to Theory, Algorithm and System Development. Prentice Hall, 2001. 1

A. *Hunt and. Black A. W.* Unit selection in a concatenative speech synthesis system using a large speech database. Proceedings of ICASSP-96, 1:373–376, 1996. 5

13. *Ikbal M. S., Misra H. and Yegnanarayana B.* Analysis of autoassociative mapping neural networks. In IEEE Proceedings of the International Joint Conference on Neural Networks, Washington, USA, 1999. 70 Angell R. C., Freund G. E. and Willett P. 1983. Automatic spelling correction using a trigram similarity measure. Inf. Process. Manage. 19,255 261.
14. *Atwell E., and Elliott S.* 1987. Dealing with ill-formed English text (Chapter 10). In The Computational Analysis of English: A Corpus- Based Approach. R. Garside, G. Leach, G. Sampson, Ed. Longman, Inc. New York.
15. *Schimke S., Vielhauer C., Dittmann J.* Using Adapted Levenshtein Distance for On-Line Signature Authentication. Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04), 2004.