Journal of Computer Science 5 (2): 90-96, 2009
ISSN 1549-3636

A Software Agent for Speech Abiding Systems

[1]R. Manoharan, [2]K. Vivekanandan and [3]V. Sundaram
[1]Department of Master of Computer Applications, SNS College of Technology,
Sathy Main Road, Coimbatore-641035, Tamil Nadu, India
[2]Bharathiar University, Coimbatore-641046, Tamil Nadu, India
[3]Karpagam College of Engineering, Coimbatore-641032, Tamil Nadu, India

Abstract: Problem statement: In order to bring speech into the mainstream of business process an efficient digital signal processor is necessary. The Fast Fourier Transform (FFT) and the butter fly structure symmetry will enable the harwaring easier. With the DSP and software proposed, togetherly established by means of a system, named here as "Speech Abiding System (SAS)", a software agent, which involves the digital representation of speech signals and the use of digital processors to analyze, synthesize, or modify such signals. The proposed SAS addresses the issues in two parts. Part I: Capturing the Speaker and the Language independent error free Speech Content for speech applications processing and Part II: To accomplish the speech content as an input to the Speech User Applications/Interface (SUI). **Approach:** Discrete Fourier Transform (DFT) of the speech signal is the essential ingredient to evolve this SAS and Discrete-Time Fourier Transform (DTFT) links the discrete-time domain to the continuous-frequency domain. The direct computation of DFT is prohibitively expensive in terms of the required computer operations. Fortunately, a number of "fast" transforms have been developed that are mathematically equivalent to the DFT, but which require significantly a fewer computer operations for their implementation. **Results:** From Part-I, the SAS able to capture an error free Speech content to facilitate the speech as a good input in the main stream of business processing. Part-II provides an environment to implement the speech user applications at a primitive level. **Conclusion/Recommendations:** The SAS agent along with the required hardware architecture, a Finite State Automata (FSA) machine can be created to develop global oriented domain specific speech user applications easily. It will have a major impact on interoperability and disintermediation in the Information Technology Cycle (ITC) for computer program generating.

Key words: Speech abiding systems, speech science, speech technology, speech user interface, speech user applications, MS speech server, SALT, companding

INTRODUCTION

Speech and natural language understanding are the key technologies that will have the most impact in the next 15 years. Enabling users to speak and listen to a computer will greatly enhance the ability for users to access computers at any time from nearly any place. Speaking and listening is so fundamental that people take it for granted. Everyday people ask questions. They give instructions. Speaking and listening are necessary for learning and training, for selling and buying, for persuading and agreeing and for most social interactions. For the majority of people, speaking and understanding spoken speech is simply the most convenient and natural way of interacting with other people. So, is it possible to speak and listen to a computer? Yes.

Speech technologies allow companies to offer the option of a self-service interface to manage tasks like rate quotes or reservations, technical troubleshooting and customer support as well as the ability to handle complex customer dialogs, capturing all the information required to provide detailed responses.

Despite the significant progress that has been made in the areas of speech recognition and spoken-language processing, building a successful dialogue system still requires large amounts of development time and human expertise. In addition, spoken dialogue systems algorithms often have little generalization power and are not portable across application domain.

Corresponding Author: R. Manoharan, Department of Master of Computer Applications, SNS College of Technology, Sathy Main Road, Coimbatore-641035, Tamil Nadu, India Tel: 91-422-2666264/+919363158114

Motivation for speaking and listening to a computer to retrieve speech content: Despite physical impairments such as blindness or poor physical dexterity. Speaking enables impaired callers to access computers. Callers with poor physical dexterity (who cannot type) can use speech to enter requests to the computer. The sight-impaired can listen to the computer as it speaks. When visual and/or mechanical interfaces are not an option, callers can perform transactions by saying what they want done and supplying the appropriate information. If a person with impairments can speak and listen, that person can use a computer. To bypass the limitations of small keyboards and screens. As devices become smaller, our fingers do not. Keys on the keypad shrink-often to the point where people with thick fingers press two or more keys with one finger stroke. The small screens on some cell phones may be difficult to see, especially in extreme lighting conditions. By speaking and listening, callers can bypass the small screen of many handheld electronic devices.

A specific problem in speech input is the room acoustics, where environmental noise may prevail, so that the frequency-dependant reflections overlay a sound wave along walls and objects with the primary sound waves. Also, word boundaries have to be defined, which is not easy, because most speakers or most human languages do not emphasize the end of one and beginning of that next word. A kind of time standardization is required to be able to compare a speech unit with existing samples. The same word can be spoken fast or slow. However, we cannot simply clench or stretch the time axis, because elongation factors are not proportional to the total duration.

Speech recognition and understanding called (abiding), by machine or a system is still a difficult and largely unsolved problem and there are a number of areas of active research that are being explored in the attempt to conquer the remaining serious problems. It is not possible to understand speech and audio signal processing in any depth without having a solid background in the mathematical underpinnings of signal processing and pattern recognition. So, advances in speech processing owe much to advancing computer technology; but, in addition, this progress has been dependent on the mathematical disciplines of digital signal processing. The connection between speech and Digital Signal Processing (DSP) is straightforward. Speech depends greatly on filtering, both production and perception.

The objective of this study to capture an error free Speech Content after having considered all the characteristics of NL. There is a viable solution to solve these issues through a well defined algorithm with the use of Fast Fourier Transform and also filtering through Decimation[1]. However let us touch upon this issue here, by giving the Signal Flow Graph (SFG) after the FFT and decimation process, forming a "Butter-fly" symmetry[2], which is really feasible to have a solution through the hardware and/or through the software. Here we have attempted to solve through software. The detail study and design and required tools and software are given in the following paragraphs.

MATERIALS AND METHODS

- Part I: Mathematical tool Fast Fourier Transform (FFT) is used
- Part II: Microsoft Speech Server (MSS Server), Speech Application Language Tags (SALT), C#, NET Frame work, Widows XP O/S

Part I: Capturing Zero-error speech content for speech user applications.

Why to focus on the FFT: The DFT[3] computational yields the spectrum of a finite sequence and hence its great importance in signal processes applications. In analyzing speech signal variations with the Discrete-Time Fourier Transform (DTFT), we encounter the problem that a single Fourier transform cannot characterize changes in speech content over time such as time-varying formants and harmonics. In contrast, the Fast Fourier Transform (FFT) consists of separate Fourier Transform for each instant in time. In particular, we associate with each instant the Fourier transform of the signal in the neighborhood of the instant, so that spectral evolution of the signal can be traced in time.

There are a great variety of FFT algorithms. They can all derive from successive applications of a single operation, by representing a one-dimensional string of numbers as a two-dimensional array. If we have N-point sequence, the integer N is either a prime or a composite number. If N is composite, it can be expressed as the product $N_1 N_2$. If either or both N_1 and N_2 are composite, further reduction is permissible. For example, we can express the number 60 as (12×5) or (3×4×5) or (2×2×5×3) and so on. The term radix is commonly used to describe this decomposition. If N can be expressed as a product of the same integer 'r', the FFT algorithm is called a radix 'r' algorithm[2]. If N is a power of 2 then DFT can be computed in $N \log_2 N$ operations. Thus for example, if N = 1024, $\log_2 N = 10$, then the number of operations = 1024×10 = 10240. This

is in contract with the "brutal force" DFT computation which takes $N^2 = 1024 \times 1024$ operations.

Therefore:

$$\text{Savings} = \frac{1024 \times 1024}{10240} = \frac{1024}{10} = 102.4$$

The savings is a factor lf 100 (ignoring the details). Therefore FFT is 100 times better than direct DFT in terms of operations alone.

The motivation for the Fourier transforms comes from the study of Fourier series. In the study, of Fourier series, complicated periodic functions are written as the sum of simple wave that are mathematically represented by Sine and Cosine. Due to properties of Sine and Cosine it is possible to recover the amount of each wave in the sum by an integral. In many cases, it is desirable to use Euler's Formula which states that $e^{\pi j\Theta}$ = Cos $2\pi\Theta$ + i Sin $2\pi\Theta$, to write Fourier series in terms of the basic waves $e^{2\pi j\Theta}$ This has the advantage of simplifying many of the formulae involved in Fourier analysis. This passage from Sine and Cosine to complex exponentials makes it necessary for the Fourier co-efficient to be complex valued. The usual interpretation of this complex number is that it gives both the amplitude (and size) of the waves present in the function and the phase (or the initial angle) of the wave.

Using compression techniques to reduce the number of windows through the use of the powerful mathematical tools like Discrete Fourier Transform

Discrete Fourier Transform (DFT)[3] and the Discrete Cosine Transform (DCT)[3] suitably adapted. In other words, capturing speech content for the signal-based audio data through discrete transformations.

Most new systems for the processing of the speech are now digital and as such are based on the fundamental mathematical tools, namely Z-Transform, Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT) that are briefly reviewed here. The transforms are the mathematical bridges that connect the time and frequency properties. The computational complexity []of the DFT is overcome by decimation process, both in time and frequency. The computational savings are quite significant and are shown herein with small examples.

Let x(n) be a signal defined for $n \geq o$ and if Z is a complex variable, X (z) is a function of a complex variable, then the Z – transform X (z) is given by:

$$\chi(z) = \sum_{n=0}^{\infty} x(n) Z^{-n} \qquad (1)$$

When the Z-transform is evaluated on the unit circle in the Z-plane, then $Z = e^{j\theta}$ and $\theta = wt$.

The Z-transform becomes:

$$\chi\left(e^{jwT}\right) = \sum_{n=o}^{\infty} x(n) e^{-jwnT} \qquad (2)$$

The basic form of the DFT given by:

$$\chi(m) = \sum_{n=o}^{N-1} x(n) e^{-j2\pi mn/N} \qquad (3)$$

It is readily apparent that computation of $\chi(m)$ for any single value of m will require (in general) N complex multiplications and N complex additions. Therefore, Computing a Complete set of N values for $\chi(m)$ will entail N^2 Complex multiplications and N^2 complex additions. Furthermore, values of $e^{-j2\pi mn\,N^{-1}}$ need to be computed for various combinations of m and n. This is overcome by indirectly using DFT using the process of decimation explained here.

Starting with the 'usual' DFT for an N-point sequence:

$$\chi(m) = \sum_{n=o}^{N-1} x(n) W_{N,}^{mn} m = 0,1.....N-1$$

Where:

$$W_N = \exp\left(\frac{-2\pi j}{N}\right)$$

Breaking the Summation into two separate summations-one for the even-indexed samples of x[n] and one for the odd-indexed samples of x[n]:

$$\chi(m) = \sum_{n=o}^{N-1} x(n) W_{N,}^{mn} m + \sum_{n=0}^{N-1} x[n] W_N^{mn}$$

$$= \sum_{n=o}^{N/2-1} x(2n) W_{N,}^{2mn} + \sum_{n=0}^{N/2-1} x[2n+1] W_N^{(2n+1)m}$$

$$\chi(m) = \sum_{n=o}^{N/2-1} x(2n) W_{N/2,}^{mn} + W_N^m \sum_{n=0}^{N/2-1} x[2n+1] W_{N/2}^{nm}$$

The factor W_N^{2mn} is equal to $W_{N/2}^{2mn}$ and W_N is called the twiddle factor.

The decimation process for an 8-point DFT and its Signal Flow Graph (SFG) is illustrated here below:

An angle-point DFT is given by:

$$\chi(m) = \sum_{n=o}^{N-1} x(n) W^{mn} m = 0,1.....7 \qquad [I]$$

(The subscript N has been omitted from W for convenience).

Splitting into separate DFTs for even and odd n:

$$\chi(m) = \sum_{n=o}^{3} x(2n)W^{2mn} + W^{m}\sum_{n=0}^{3} x[2n+1]W^{2nm} \qquad [II]$$

If we let A[m] denote the DFT of the even-indexed samples and B[m] denote the DFT of the odd-indexed samples, then we can write the above equation as:

$$\chi(m) = A[m]+W^{m}B[m] \qquad [III]$$

$\chi(m)$ is an 8-point frequency sequence, but A[m] and B[m] are only 4-point frequency sequences. Since an N-pint DFT is periodic with a period of N samples i.e.:

A[4] = A[0], B[4] = B[0], A[5] = A[1] , B[5] = B[1]

Therefore:

$\chi[4] = A[0]+W^{4}B[0]$
$\chi[5] = A[1]+W^{5}B[1]$
$\chi[6] = A[2]+W^{6}B[2]$
$\chi[7] = A[3]+W^{7}B[3]$

The operations represented by the above equations are depicted in the Signal Flow Graph (SFG) (Fig. 1). In an SFG there are nodes and edges. Each node represents a signal that is obtained by summing together all of the signals represented by the edges directed into the node. Each edge represents the multiplication of a weight times the signal that is represented by the edge's source node. An edge's weight is indicated by an annotation near the arrowhead used to indicate the edge's direction.

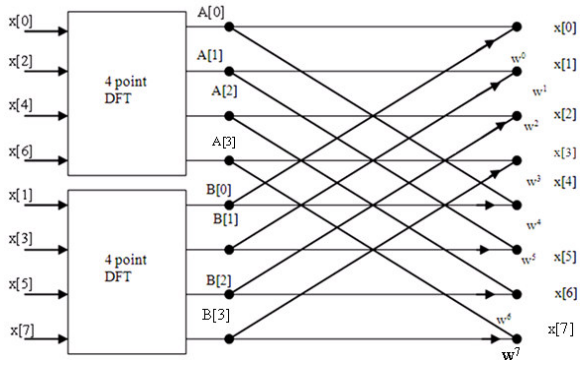

Fig. 1: Signal flow graph illustrating how an 8-point DFT can be formed in terms of two 4-point

By decimation, "Butter-fly" symmetry is achieved which would be congenial for hardware[2] implementation, if necessary and as well software implementation in Part-II. Sometime DFT, after decimation process may be called as Decimated Fourier Transform (DFT).

Part II: Providing an environment for developing and deploying a Speech User Application/Interface

A software study was conducted[4]. It is realized that an effective languages and tools are required for speech processing[5]. Based on the study these software are proposed: MS Speech Server, SALT/C#, .NET Framework, SDK, under Windows environment.

Why to Focus on SALT: VoiceXML and SALT are both markup languages that describe a speech interface. However, they work in very different ways, largely due to two reasons: (i) They have different goals; (ii) They have different Web heritages. VoiceXML is designed for telephony applications. It was developed to allow the specification of Interactive Voice Response (IVR) applications in a markup language that leveraged the benefits of the World Wide Web. It is a simple, high-level dialog markup language that facilitates the authoring of system-driven and mixed-initiative voice dialogs over telephones and cell phones. SALT targets speech applications across a whole spectrum of devices, including telephones, PDAs, tablet computers and desktop PCs. Since many devices also contain displays, multimodal interactions are a key focus. Developers use SALT with existing Web programming standards to author system-driven, user driven and mixed-initiative voice dialogs and multimodal applications. These differences are manifested mainly in (i) The form of the markup, (ii) The programming and execution model and (iii) The level of the programming interface available to the developer.

Scope: VoiceXML incorporates speech interface, data and control flow, SALT focuses on the speech interface Programming model VoiceXML has a built-in, form-filling algorithm, SALT enables application developers to write customized dialog flow. Level of API: VoiceXML has a high-level API and SALT has a lower level API. Other standards: VoiceXML and SALT both use W3C standards. Both VoiceXML and SALT recommend the use of SRGS and SSML as grammar and speech output formats, respectively. In addition, SALT also recommends the use of NLSML as a recognition result format and CCXML as a telephony call control language (or a call control object closely modeled on CCXML as an alternative). Licensing VoiceXML may be subject to royalty payments and

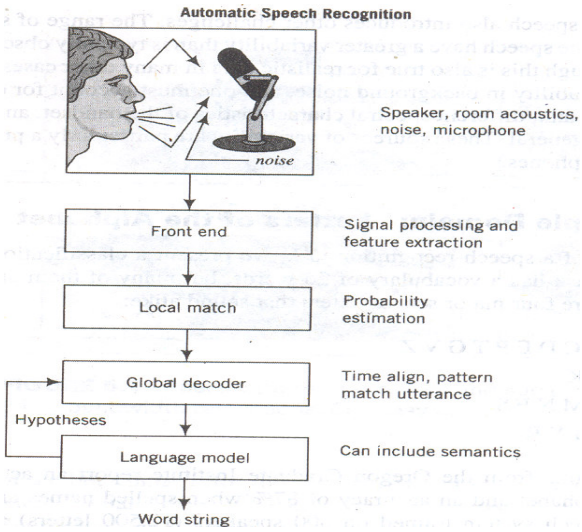

Fig. 2: Block Diagram of agent SAS

SALT will be royalty-free. The future: Although SALT and VoiceXML were developed to solve different problems, these problem spaces are beginning to converge. Some VoiceXML developers are asking for a stripped-down version of VoiceXML, without the FIA, so they can write their own turn taking strategies for complex speech applications. Other VoiceXML developers are asking that VoiceXML be modularized so that its tags can be embedded into other languages. SALT already applies a model that fits these roles.

Figure 2 shows the flow of processing of SAS. The user will interact with help of a microphone and in the front end signal processing and extraction is done by a DSP, after the analogue to digital conversion process is done. This captured error free speech content are then matched with predefined recorded sample which have gone through the same process and stored. Through the function global decoder an automatic intelligent construct is made through a subroutine language model, to response an unorganized query into a disciplined query and response.

Figure 3 shows the proposed architecture that can support the proposed agent SAS. The development tools are Speech application SDK and Microsoft Visual Studio. The SALT clients are through telephones, pocket PC and desktops. The Web Server could be Microsoft Speech server having the interface ASP.net or C# which could do the job of prompts and grammatical disciplines. If we attempt to do through web IIS (Internet Information Server) need to be configured. Through the SALT with HTML along with any scripts can interface with server to fetch the speech content from the speech core.

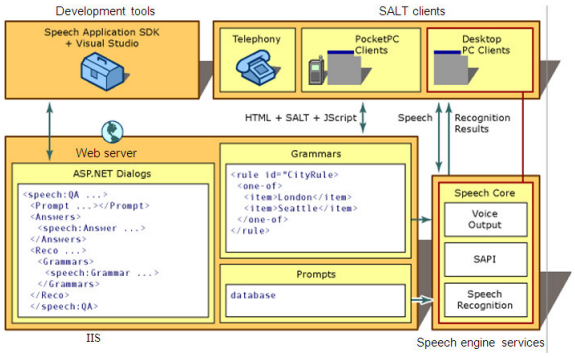

Fig. 3: Proposed architecture of an intelligent software agent

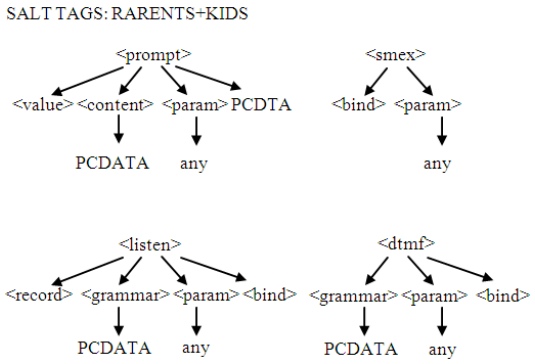

Fig. 4: The basic structure of SALT

Sample coding: The following example shows a simple form of dialog flow management using client-side script. The SALT primitives <listen> and <prompt> are activated accordingly to the RunAsk() script which examines the values inside the input fields and executes the relevant prompts and recognitions until the values of the both fields are obtained. In order to illustrate programmatic result processing, the binding of the recognition results into the relevant input fields is accomplished below by the script functions procOriginCity() and procDestCity(), which are triggered by onReco events of the relevant <listen> elements. The handler for an unrecognized speech event onNoReco is used to play an appropriate message - the SayDidntUnderstand prompt, which restarts the cycle on its completion.

A sample program (part of the main program) is shown as per the SALT structure shown in the Fig. 4.

```
<!—- HTML -->
<html xmlns:salt="urn:saltforum.org/schemas/020124">
<body onload="RunAsk()">
<form id="travelForm">
 <input name="txtBoxOriginCity" type="text" />
```

```
 <input name="txtBoxDestCity" type="text" />
 </form>
<!—- Speech Application Language Tags -->
<salt:prompt id="askOriginCity"> Where would you like to leave from? </salt:prompt>
<salt:prompt id="askDestCity"> Where would you like to go to? </salt:prompt>
<salt:prompt id="sayDidntUnderstand" onComplete="runAsk()">
Sorry, I didn't understand.
</salt:prompt>
<salt:listen id="recoOriginCity"
onReco="procOriginCity()"
onNoReco="sayDidntUnderstand.Start()">
<salt:grammar src="city.xml" />
</salt:listen>
<salt:listen id="recoDestCity"
onReco="procDestCity()"
onNoReco="sayDidntUnderstand.Start()">
<salt:grammar src="city.xml" />
</salt:listen>
<!—- script -->
<script>
function RunAsk() {
if (travelForm.txtBoxOriginCity.value=="") {
askOriginCity.Start();
recoOriginCity.Start();
} else if (travelForm.txtBoxDestCity.value=="") {
askDestCity.Start();
recoDestCity.Start(); }
 }
function procOriginCity() {
travelForm.txtBoxOriginCity.value = recoOriginCity.text;
RunAsk();
}
function procDestCity() {
travelForm.txtBoxDestCity.value = recoDestCity.text;
travelForm.submit();
}
</script>
</body>
</html>
```

As noted above, further event handlers are available in the <listen> and <prompt>[11] elements to manage user silences, errors and other situations requiring some form of dialog recovery.

By deploying SAS, a primitive level of application is generated for creating a domain specific speaker and language independent processing.

RESULTS

A software agent is created to capture zero-error speech content instantly and is stored in the speech database as samples for that particular domain. These samples are speaker independent and language independent. Further this is be retrieved for actions and also being queried for intelligence and intellisense response automatically. In essence, SAS agent facilitates for developing a primitive level of a global oriented domain specific, speaker and language independent applications, for any speech users interface business process systems. The SAS agent will enhance the accessibility of the users those who know natural language by 85% in number and will reduce the cost of transactions processing[5] in business applications, projected for about 50%.

DISCUSSION

The same level of agents, components and products will have more impact and be the ingredients of key technologies of business processing in the years to come. Even the blind and visually impaired people can also be supported by these technologies[6]. The first commercial implementation of a free speech speaker verification system is implemented in a call center[7]. A single source formatting[4] is also possible by the combinations of Graphic User Interface (GUI) and Voice User Interface (VUI) by leveraging the existing infrastructure. People who can able to speak will interact with computer in any language the know with help of the intelligent agent SAS proposed.

CONCLUSION

The Speech Abiding System (SAS) agent along with required hardware architecture can be made as a product instead of component, to accomplish all speech users' applications. It might emphasize naturalness in the mechanism of interoperability and reduce disintermediation (known delays), in the Information Technology Cycle (ITC), in the spectrum of computer application programming. And as Natural Language (NL) speech applications move into the mainstream, chances are your enterprise will install the technology during the next 12-18 months[8].

ACKNOWLEDGEMENT

The researcher would like to thank Dr. S.N. Subbramanian, Director cum Secretary, Dr. S. Rajalakshmi, Correspondent, SNS College of Technology, Coimbatore, India for their motivation and continual constant encouragement.

REFERENCES

1. Britton Rorabaugh, C., 2005. DSP Primer. Tata McGraw-Hill Publishing Company Limited, New Delhi, ISBN: 0-07-060349-9, pp: 163-164.
2. Hsu, Y.P. and S.Y. Lin, 2008. Implementation of low memory reference FFT on digital signal processor. Int. J. Comput. Sci., 4: 545-549. http://www.scipub.org/fulltext/jcs/jcs47545-549.pdf
3. Subramanian, V.S., 2001. Principles of Multimedia Database Systems. Morgan Kaufmann Publishers, Inc., San Francisco, CA., USA., ISBN: 81-7867-041-0, pp: 223.
4. Plomp, C.J. and O. Mayora-Ibarra, 2002. A generic widget vocabulary for the generation of graphical and speech-driven user interfaces. Int. J. Speech Technol., 5: 39-47. DOI: 10.1023/A:1013678514806
5. Suhm, B. and P. Peterson, 2002. A data-driven methodology for evaluating and optimizing call center IVRs. Int. J. Speech Technol., 5: 23-37. DOI: 10.1023/A:1013674413897
6. Olaszi, P., I. Koutny and S.L. Kálmán, 2002. From bliss symbols to grammatically correct voice output: A communication tool for people with disabilities. Int. J. Speech Technol., 5: 49-56. DOI: 10.1023/A:1013682632553
7. Fogel, N., 2002, A commercial implementation of free-speech speaker verification system in a call center. Int. J. Speech Technol., 5: 313-320. DOI: 10.1023/A:1020956924277
8. Aaron Fisher, 2009. A natural part of natural language. http://www.speechtechmag.com/Articles/Column/Inside-Outsourcing/A-Natural-Part-of-Natural-Language-52492.aspx