A Review of AlfaNum Speech Technologies for Serbian, Croatian and Macedonian

Vlado Delić*, Milan Sečujski*, Darko Pekar[&], Nikša Jakovljević*, Dragiša Mišković[&]

* Faculty of Engineering, University of Novi Sad
Trg Dositeja Obradovića 6, Novi Sad, Serbia
{vdelic, secujski, jakovnik}@uns.ns.ac.yu

[&] AlfaNum – Speech Technologies
Trg Dositeja Obradovića 6, Novi Sad, Serbia
{darko.pekar, dragisa.miskovic}@alfanum.co.yu

Abstract

This paper gives a brief review of the development of systems for automatic speech recognition and text-to-speech synthesis in Serbian, Croatian and Macedonian language, at the Faculty of Engineering, University of Novi Sad, Serbia. The systems developed within this project enable two-way communication between humans and machines. These systems are predecessors to many commercial services such as voice portals and interactive voice response systems. Some original features of these systems, related to certain particularities of south-Slavic languages, will be illustrated as well. Available APIs and interfaces designed for using these software components in custom applications will also be described.

Pregled govornih tehnologij za srbščino, hrvaščino in makedonščino skupine AlfaNum

Članek poda kratek pregled razvoja sistemov samodejnega razpoznavanja govora in samodejnega tvorjenja govora iz besedil za srbski, hrvaški in makedonski jezik na Fakulteti za inženiring Univerze v Novem Sadu. Sistemi, ki so bili razvit pri tem projektu, omogočajo dvosmerno komunikacijo med človekom in računalnikom. Tovrstni sistemi so predhodniki mnogih komercialnih storitev, kot so glasovni portali in interaktivni govorni odzivniki. Ponazorjene bodo tudi nekatere izvirne značilnosti sistemov, ki so povezane s posebnostmi južnoslovanskih jezikov. Ob tem bodo opisani tudi razpoložljivi programski vmesniki (API) in vmesniki, ki smo jih razvili za uporabo programskih komponent pri uporabniških aplikacijah.

1. Introduction

Since speech is the most natural means of communication between humans, people have been trying to develop system that would enable them to extend this interface to communication with machines as well. Using automatic speech recognition (ASR) and text-to-speech synthesis (TTS), humans can talk to devices in their midst, such as household appliances, industry machines, cars, toys or remote computers, even via telephone. One of the important applications of these technologies is also providing independence in computer access to the physically impaired (particularly the visually impaired).

However, speech technologies are language dependent and in some regions they can hardly ever be imported from abroad as most other technologies. They have to be developed for each language separately, as is the case with languages such as Serbian, Croatian and Macedonian, having in mind all their peculiarities. It cannot be expected that some of the world's biggest companies dealing with speech technologies would decide to develop all resources needed for high-quality ASR and TTS for such a small and closed market in the near future.

A group at the Faculty of Engineering, University of Novi Sad, Serbia, called *AlfaNum* has been dedicated to the development of these technologies for ten years, and their results include phoneme-based automatic speech recognition over the telephone line with accuracy varying between 95% and 99% depending on vocabulary size and sound quality, as well as the first text-to-speech system in Serbian, Croatian and Macedonian taking into account linguistic information and thus greatly enhancing intelligibility and naturalness of synthesized speech. These systems are the basis of a number of applications, such as audio libraries and a speech-enabled web site for the visually impaired, as well as many commercial applications including interactive voice response systems and call centres.

2. AlfaNum Speech Synthesizer

During the development of the first high-quality TTS for Serbian language, this team has encountered many problems linked to bridging the gap between plain text and synthesized speech with all its typical features such as intelligibility and naturalness. There is no explicit information in a plain text concerning phone durations, pitch contours nor energy variations. These factors also depend on the meaning of the sentence, emotions and speaker characteristics, which further aggravates the task of attaining high naturalness of synthesized speech (Dutoit, 1997). While Serbian and Croatian are tonal languages, having high-low pitch patterns permanently associated with words, Macedonian is a pitch-accented language with antepenultimate stress on most words, excluding clitics, words of foreign origin as well as some other word groups. Nevertheless, a uniform dictionary-based strategy for lexical stress assignment has been successfully employed in all three cases.

The TTS engine has two main functions: text analysis and synthesis of the speech signal. Text analysis includes text processing such as expanding abbreviations, as well as resolution of morphological and syntactic ambiguities based on a comprehensive accentuation dictionary as well as rule-based syntax analysis. A separate dictionary and syntax analysis techniques were required for each language. Lexical stress assignment, as one of the most important factors influencing the shape of the pitch contour, is performed using a rule-based algorithm described in (Sečujski, Delić, 2006). This approach has proved to produce reasonably correct stress pattern, with word error rate 2,8% for Serbian language. No equivalent tests have been carried out for either Croatian or Macedonian so far. The synthesized speech in all three languages is highly intelligible and reasonably natural-sounding, much more than any other attempts at speech synthesis in Serbian, Croatian and Macedonian so far. An objective test showing the improvement in TTS quality introduced by accent-based prosody is described in (Sečujski et al., 2002).

For the development of a multilingual TTS, separate speech databases are generally required for each language, although the Serbian database is at the moment used for Macedonian. This leads to a quite insignificant decrease in speech quality, due to the fundamental similarity between phonetic inventories of these two languages. The Macedonian speech database is expected to be recorded soon. If phonetic inventories are similar enough, as is the case for Serbian, Croatian and Macedonian language, it is appropriate accentuation and appropriate pitch contours that will make synthesized speech sound naturally, almost regardless of the original language of the database.

As to speech signal synthesis, the concatenative approach has been selected as the most promising. The AlfaNum R&D team has recorded a large speech database and labeled it using visual software tools specially designed for that purpose. By keeping score of every phone in the database and its relevant characteristics, use of phones in less than appropriate contexts was avoided, which further contributed to overall synthesized speech quality. This synthesizer is not diphone-based as almost all other speech synthesizers developed for related languages are. The TTS engine can use larger speech segments from the database, according to both phonetic and prosodic requirements, and select them at runtime in order to produce the most intelligible and natural-sounding utterance for a given plain text (Beutnagel et al., 1999).

The most significant application of this system so far is *anReader*, a speech synthesizer for the visually impaired that, combined with software known as *screen-readers*, offers them complete independence in computer access. The number of the visually impaired computer users in Serbia has increased significantly since *anReader* was presented for the first time, and its popularity in Croatia and FYR Macedonia is also growing.

3. AlfaNum System for Automatic Speech Recognition

The goal of ASR is to recognize spoken words in a speech signal, independently of the speaker, the input device, or the environment. A recognized sequence of words W_{ASR} for a given acoustic observation sequence X and all expected word sequences W is usually estimated using Bayes rule:

$$W_{\mathrm{ASR}} = \arg_W \max P(W|X) = \arg_W \max P(W)\cdot P(X|W)$$

where $P(W)$ is the *language model* estimated using *n*-gram statistics and $P(X|W)$ is the *acoustic model* represented by a Hidden Markov Model (HMM), trained using maximum likelihood estimation. HMM encodes the acoustic realisation of speech and its temporal behaviour, while prior probabilities for word sequences $P(W)$ lead to a choice of the word sequence hypothesis with the maximum posterior probability given the models and observed acoustic data. The best word sequence W_{ASR} is computed using a pattern recogniser based on a standard Viterbi decoder. A conventional approach to front-end signal processing of 30 *ms* frames, every 10 *ms*, results in a feature vector X that captures primarily spectral features of the speech signal estimated as cepstrum and energy, along with their first- and second-order time derivatives. A finite vocabulary defines the set of words (sequences of phone units) and phrases that can be recognised by the speech recogniser. The size of the recognition vocabulary plays a key role in determining the accuracy of a system, typically measured in Word Error Rate (WER), including insertion, deletion, and substitution errors.

R&D for Serbian, Croatian and Macedonian ASR has been concentrated on four aspects that define the quality of a speech recognition technique (Gilbert et al., 2005):

§ *Accuracy* – WER is less than 5% for small and medium-sized vocabulary continuous ASR; it is achieved by developing acoustic models trained with 40 hours of speech database; good results for large vocabulary continuous ASR in these languages are expected when a more complex language model and more comprehensive post-processing are implemented.

§ *Robustness* – channel distortions are compensated by CMS (Cepstral Mean Subtraction), background noise spectrum is subtracted and speaker variations are treated by gender separation and speaker adaptation based on VTN (Vocal Tract Normalization).

§ *Efficiency* – long work on software code optimization has resulted in fast decoder and small memory footprint. The ASR engine consumes 2% or more of CPU time on a Pentium IV PC, depending on vocabulary size.

§ *Operational performance* – The ASR engine gives a useful confidence scoring and implements barge-in capability, improving operational performance. On the other hand, features such as rejection of out-of-vocabulary speech have not yet been enabled.

Due to the complexity of the problem, a system for isolated word recognition in Serbian language was developed initially. It was later upgraded into a system for connected word recognition. Eventually a system for continuous speech recognition (CASR) was developed, based on recognition of phonemes in particular contexts. An elementary HMM model is a triphone model, representing a phoneme in a particular left and right context. In case there are too few instances of a triphone in the database, model-tying procedures are performed (Pekar, 2002). The advantage of phoneme-based approach is that users can define an arbitrary set of words (vocabulary) for each recognition at initialization time. The system takes into account lexical stress (particularly vowel length), assigning greater significance to stressed vowels at recognition time.

This system can be used for speech recognition in all three aforementioned languages because it is phoneme-based and because of the similarity of phonetic inventories of these three languages. No significant drop in performance for languages other than Serbian has been observed, but actual experiments will be carried out as soon as adequate ASR speech databases in Croatian and Macedonian are available.

Even state-of-the-art ASR systems cannot be successful enough if they are based on acoustic features only. In order to achieve natural dialogs in speech applications, AlfaNum ASR has to apply some post-processing such as Spoken Language Understanding (SLU), as well as a lot of experience in both machine learning and design of front-end technology. The goal of SLU is to extract the meaning of recognized speech in order to identify a user's request. Dialog Manager (DM) evaluates the SLU output in context of the call flow specifications, which results in dynamic generation of the next dialog turn. The DM may apply a range of strategies to control dialog flow according to different application tasks. To provide a successful dialog progress, intelligent speech applications have to handle problematic situations caused by system failures or absence of concise or accurate information in a speech utterance. Post-processing makes it viable to adopt natural language dialog applications without having to achieve perfect recognition accuracy and without dictating what a user should say.

4. API for AlfaNum ASR and TTS

So far we have elaborated some general features of recognizer and synthesizer (ASR and TTS engine) and their capabilities. However, in order to be able to integrate ASR and TTS engine into a specific application, it is necessary to implement appropriate interfaces. Depending on the application and programming language in which it was designed, one of the forms of the application programming interface (API) is chosen.

For that reason, several versions of interfaces to the software component have been implemented, and a potential application designer can decide which one to use. Among other interfaces, standard MS SAPI4 and MS SAPI5 interface (speech APIs proposed by Microsoft) have been implemented. The full compatibility of ASR and TTS engines with MS SAPI4 and MS SAPI5 means that any application can access ASR and TTS engines via SAPI functions. Other interfaces implemented include a custom C++ library, socket communications and COM interface.

4.1. API for AlfaNum ASR

As mentioned above, AlfaNum ASR works with small and medium vocabularies. In order to make the system recognize specific words, a grammar must be defined. This is accomplished using regular expressions, which will be explained later. It is clearly possible to define several grammars and to decide which one to use for recognition at each moment. A specific way how to do this depends on the interface used. The input of the recognizer always consists of the speech signal and the name of the grammar. The output consists of two arrays. The first one is the array of recognized words (strings), such as ['DAJTE', 'MI', 'LOKAL', 'TRI', 'PET', 'DVA']. The second one is the array of numeric values, each of them defining the reliability of recognition of the corresponding word, i.e.: [93.1, 73.2, 90.0, 86.7, 91.2, 93.5]. Reliability values lie in the range 0-100. The exact format of these arrays depends on the interface applied.

Grammars are defined using Backus-Naur form which will be explained through an example.

Let us consider a grammar designed for recognition of a telephone extension number. In this grammar a user can, but does not have to, say the word "lokal" (extension), and after that a sequence of digits is expected. Before and after any speech activity some noise can occur, and the entire spoken sequence is optional (i.e. the user can stay silent).

```
digit = NULA | JEDAN | DVA | TRI | CHETIRI
| PET | SHEST | SEDAM | OSAM | DEVET;
lokal = LOKAL;
gr = <gar>;
main = [$gr] [[$lokal] <$digit>] [$gr];
```

Several elements can be observed:

variables – digit, lokal, gr, main. Variable "main" is the only reserved word and denotes the main sequence, i.e. what is to be recognized. For that reason it is defined at the end. Other variables can be referenced in any of the following definitions, which can be accomplished by using the prefix "$".

- **mark "|" –** denotes a choice. The recognizer will choose one of all given words.
- **angle brackets "<>" –** surrounded sequence can occur once or several times.
- **square brackets "[]" –** denotes an optional sequence. The recognizer can pass through this word (or the whole rule), or skip it.
- **reserved word "gar"** – denotes noise model used for noise itself as well as some sounds that could be produced by the speaker but are not qualified as orthographic words.

4.1.1. AlfaNum ASR Server

All interfaces which will be mentioned here rely on ASR server, which contains the recognition engine. All client applications communicate with the server over the IP protocol. This enables remote access to the server and distribution of multiple ASR servers (implemented on several computers) in case there is a need for large number of simultaneous recognitions. All interfaces first connect to the server, then send a command and wait for a response. Commands are usually recognition requests, although the protocol supports many other commands as well.

An application designer can always use low level IP communication with server. However, we have developed a higher level library in C++, enabling fully functional communication with the server using a very small number of functions, with no need for low level protocol details. The library contains the following functions and fields:

- **void AddHost (const string &host_name, float host_load) –** adds computer **host_name** to the list of computers having ASR server capabilities. **host_load** represents the load coefficient of a particular server. Servers will be used according to these coefficients.
- **void Connect () –** connects to ASR server.
- **void Disconnect () –** disconnects from ASR server.
- **void RecognizeFromFile (const string &grammar, const string &file_name, float timeout_s)** – Recognizes the utterance recorded in the file **file_name** in accordance with **grammar**. The result is stored in fields **results** and **reliabilities**.
- **void AddGrammarAsync (const string &name, const string &grammar_file_name, const string &transcriptor, const string &pronunciation, const string &postprocessor, int timeout_s = -1)**

– Starts a process of adding a new grammar named **name**, defined in the file **file_name**. Since this operation can be time-consuming, the process is asynchronous in order to avoid the client being blocked during initialization.

- **vector <string> results –** string vector with results of the last recognition.
- **vector <float> reliabilities –** float vector with values denoting reliability of the latest recognition.

The COM interface is similar to the C++ interface, but it contains a set of methods and properties which can be used in virtually any programming language. Most descriptions of functions and properties are the same as the ones given for the C++ interface:

- **AddHost(host_name As String, host_load As Single)**
- **Connect ()**
- **Disconnect ()**
- **RecognizeFromFile(grammar As String, file_name As String , timeout_s As Double)**
- **GetRecoResults As RecoResults –** returns an object of **RecoResults** type, with a string array containing spoken words.
- **GetResultsReliabilities As ResultsReliabilities** – returns an object of type **ResultsReliabilities** which contains reliabilities of recognized words.

Microsoft SAPI5 interface also relies on the ASR server, and on the client side contains methods and properties defined in MS SAPI5 standard. All details on this standard can be found at www.microsoft.com/speech.

4.2. API for AlfaNum TTS

The basic functionality of the AlfaNum TTS system is to transform an input text into speech. Textual input is usually an unicode string, while the output is usually a wav file. Beside this basic functionality, it is possible to set other input parameters such as pitch, speed, speaker, accentuation manner, etc.

It is possible to obtain an audio stream from the synthesizer, which means that the client side can get parts of audio signal even before the whole text has been processed. In this way, delays are much shorter than in case the signal is available only after the whole synthesis has been completed.

All interfaces which will be mentioned rely on TTS server. It contains text-to-speech engine, and client applications communicate with the server over the IP protocol, in a way rather similar to the ASR server.

An efficient C++ library for communication with the server was developed, similar to the one developed for communication with ASR server. Examples of TTS-specific functions and fields are given below:

- **void Synth (const string &file_name, const wstring &text, int timeout_s)** – synthesizes sentence given in unicode string text and puts it in the file file_name.
- **float speed** – defines the speed of the synthesized speech.
- **float pitch** – defines the pitch of the synthesized speech.
- **string speaker** – defines the speaker to be used for synthesis.
- **bool read_punctuation** – defines if punctuation marks should be read out.
- **bool read_abbreviations** – defines if abbreviations should be spelled.
- **bool letter_by_letter** – defines if the entire text should be spelled.
- **bool prosody_override –** defines if the user is allowed to specify his/her own preferred accentuation of a specific word.
- **bool character_substitution –** defines if, in the absence of letters with diacritics (č, ć, š, ž...), there should be an attempt to replace them with appropriate letters with diacritics.

Examples of methods and properties of the COM TTS interface as well as the Microsoft SAPI4 and SAPI5 interface are the same as for the ASR server, with addition of TTS-specific methods and properties related to the functions and fields given above.

5. Conclusion

Two speech technologies developed for the Serbian, Croatian and Macedonian language have enabled two-way communication between humans and machines in these languages. This communication can be direct or remote (e.g. via telephone), which introduces the possibility of building speech-enabled intelligent systems. This is a step of a human-to-machine interface in the regions where these languages are spoken from touch-tone prompts toward multimedia and multimodal interface. While both these technologies are still under development, they are already implemented in many commercial applications, and are also in wide use as aid for people with visual disabilities in Serbia, Croatia, Bosnia-Herzegovina and FYR Macedonia. Owing to a number of interfaces developed as well as appropriate reference manuals, these software components can be used by third party programmers who want to develop their own speech-enabled applications.

6. References

Beutnagel, M., Mohri, M., Riley, M., 1999. Rapid Unit Selection from a Large Speech Corpus for Concatenative Speech Synthesis. In *Proc. of EUROSPEECH'99*, Budapest, 607-610.

Dutoit, T., 1997. *An Introduction to Text-to-Speech Synthesis*. Dordrecht: Kluwer academic publishers, 149-152.

Gilbert, M.; Wilpon, J. G.; Stern, B.; Di Fabbrizio, G., 2005. Intelligent Virtual Agents for Contact Center Automation, *IEEE Signal Processing Magazine*, Vol. 22, 5:32-41.

Pekar, D.; Obradović, R.; Delić, V., 2002. AlfaNumCASR – a system for continuous speech recognition. In *Proc. of 3rd Conference DOGS*, Bečej, Serbia, 49-56.

Sečujski, M., Obradović, R., Pekar, D., Jovanov, Lj., and Delić, V., 2002. AlfaNum System for Speech Synthesis in Serbian Language. In *Proc. of 5th Conf. Text, Speech and Dialogue*, Brno, 8-16.

Sečujski, M., Delić V., 2006. A software tool for semi-automatic part-of-speech tagging and sentence accentuation in Serbian language. In *Proc. of IS-LTC*, Ljubljana.

www.microsoft.com/speech

www.microsoft.com/downloads, Speech SDK 5.1