Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 711–716

ISBN 978-83-60810-22-4

Automatic Speech Recognition for Polish in a Computer Game Interface

Artur Janicki and Dariusz Wawer
Institute of Telecommunications, Warsaw University of Technology
ul. Nowowiejska 15/19, 00-665 Warsaw, Poland
Email: A.Janicki@tele.pw.edu.pl, Dariusz.Wawer@gmail.com

***Abstract*—The paper describes the process of designing a task-oriented continuous speech recognition system for Polish, based on CMU Sphinx4, to be used in the voice interface of a computer game called *Rally Navigator*. The concept of the game is presented, the stages of creating the acoustic model and the language model are described in details, taking into account the specificity of the Polish language. Results of initial experiments show that as little as 15 minutes of audio material is enough to produce a highly effective single-speaker command-and-control ASR system for the computer game, providing the sentence recognition accuracy of 97.6%. Results of the system adaptation for a new speaker are presented. It is also showed that the statistic trigram-based language model with negative trigrams yields the best recognition results.**

I. Introduction

AUTOMATIC speech recognition (ASR) systems gradually replace keyboards and touch pads in various applications - so it happens in word processors where dictation software is being introduced. But there are also trials to replace joysticks and buttons in computer games, thus making the games more interesting and enabling multimodal input. ASR systems can be successful in computer games, on condition that they provide high recognition accuracy and short processing time.

To ensure realistic conditions, an ASR system in the computer game should be able to recognize continuous speech, which is how people usually talk. Continuous speech recognition is much more difficult than recognition of isolated words. What is more, apart from a few examples ([1], [2]), such systems barely exist for the Polish language, which is highly inflective and thus hard to recognize.

This paper describes experiments with designing a small-vocabulary task-oriented automatic continuous speech recognition system for Polish, which can be used in the voice interface of a computer game.

II. Automatic Speech Recognition

A. Methods for Continuous Speech Recognition

Early works on ASR systems, starting in the 1950s, concerned recognition of isolated phonemes, or at best - a few words. One example is the isolated digits recognizer constructed at Bell Labs in 1952 [3]. The invention of Dynamic Time Warping (DTW) in the late 60s allowed for projects on larger-vocabulary word recognition and for processing of connected words [4]. In 1971 the ARPA SUR (Speech Understanding Research) program started, aiming at creating a reliable large vocabulary ASR for continuous speech. One of its results was HARPY - an ASR system developed at the Carnegie Mellon University, working with semantic accuracy of 95% at the processing speed of 80 times real-time [4]. Studies on continuous speech recognition continued intensively in the 1980s, when the usage of statistic acoustic modeling and statistic language modeling advanced, and they have continued till nowadays.

The key to successful continuous speech recognition is a combination of a highly accurate *acoustic modeling* (AM) and a proper *language modeling* (LM).

1) Acoustic modeling: its aim is to recognize as accurately as possible the phonetic content of the input speech signal, by comparing parameters of the speech signal (usually MFCC - mel-frequency cepstral coefficients or PLP - perceptual linear prediction parameters) with acoustic models stored in the ASR system. There were successful trials of using artificial neural networks for phonetic modeling, but contemporary systems almost exclusively use statistical acoustic modeling based on Hidden Markov Models (HMM). Context-dependent phonemes, called also triphones, are usually the base speech units used in acoustic modeling. Efficient acoustic models are trained on multi-speaker speech corpora containing hours of transcribed recordings, therefore their preparation for a new language is a very demanding and time-consuming task.

2) Language modeling: it is very important, because for continuous speech the word boundaries are difficult or impossible to detect. The language model enables the ASR system to decode the sequence of phonemes, recognized during acoustic recognition into the correct sequence of words. A proper language model makes it very likely that the recognized sequence will have correct syntax and will be semantically correct. One of the language modeling techniques is the use of N-grams [5]. In this statistical method sequences of N words are assigned various likelihoods. Such a language model is created based on statistical analysis of a given language or a given domain, depending on the ASR type (e.g. if it is a large-vocabulary one or task-oriented). N ranges usually between 2 to 4. Some probability mass is left for 2-grams (*bigrams*) and 3-grams (*trigrams*) unseen during the training procedure - such an operation is called *model smoothing*.

Effective way of decoding the text is to perform a Viterbi search on a recognition network (it can be in the form of

a tree or word lattice), built out of the lexicon. Because of high calculation complexity, often the acoustic recognition (phoneme-based) and lexical one (word-based) are performed sequentially: as we proceed with the signal analysis, word sequence hypotheses are created, and at the same time acoustic and language scores are calculated, so that only the paths with highest scores are continued. The sequence with the highest score which reaches the end of the signal is the decoded sentence.

Another method of approaching language modeling is to build a grammar describing all possible phrases. To describe a grammar one can use Java Speech Grammar Format (*JSGF*) [6], which is a part of Java Speech API [7] and is designed for strict command and control systems. The grammar is defined by declaring rules which can contain either words, operators, or other previously declared objects. Such a language model based on grammar leaves no margin for any unforeseen word sequence.

B. The Sphinx Framework

CMU Sphinx framework, partly funded by DARPA, was created and has been continuously developed at Carnegie Mellon University. It consists of several subprojects. In our work we used *SphinxTrain* [8] and *Sphinx4* recognizer. SphinxTrain consists of tools written in C which can be used to train and adapt acoustic models. It also provides scripts which simplify acoustic model generation. Sphinx4 is written in Java and is a complete recognition system with modular architecture. It communicates with the application in two ways: the first one is the input, which acquires the audio data, and the output, which returns the recognition - the best match for the data. The other communication input is a recognizer control mechanism and the output are system state notifications.

Internally, Sphinx4 consists of the frontend, the decoder, the linguist and the configuration manager. The frontend receives audio data and may perform early signal modifications like removing long silences or amplifying the signal. After the signal leaves the frontend, the feature extraction occurs and the data reaches the decoder. Simultaneously the linguist generates the possible word sequences (possibly as word lattices or trees) based on the language model and sends them to the decoder. With the voice features and data from the acoustic model the decoder scores the possible sequences keeping only the most probable. After the signal finishes or a final state is reached, the result of the recognition is returned to the application. This corresponds to the speech recognition process described earlier in Section II.A.

Many of the above mentioned elements have several implementations, and each implementation can be separately configured. E.g. there are implementations for both N-gram and JSGF language models. The Sphinx framework was originally designed for English, but nowadays it supports also, among others, Spanish, French and Mandarin. However, there are no acoustic nor language models available for Polish.

To use it with Polish, the package required some modifications to work properly, e.g. several internal classes needed to be adapted to correctly read names containing non-ASCII characters, which were found in the language models and dictionaries.

C. Speech Recognition in Computer Games

Nowadays computer games are a large and well developing industry, generating high revenues. To be successful in the market, a game has to be easy to play, intuitive, must be fun and interesting and, last but not least, must not irritate the player.

While the first three of the requirements apply rather to the game design, the last one refers as well to the ASR system, and it places severe restrictions on the system. First of all, the system must recognize the command in a short time. The amount depends on how fast is the action in a game, but times greater than one second are definitely too long. Secondly, the accuracy of recognition must be really high. Should the system make an error and issue a wrong command, the player would be unhappy. Should such situation occur repeatedly, he or she would quit the game and never play it again.

An ideal game using speech recognition should work robustly for every speaker, including people with various accents, non-native speakers and even people with speech deficiencies.

There are many factors that can have negative impact on speech recognition robustness:

- usually a fairly low quality microphones being used;
- games are often played in noisy environments;
- highly-effective, fast ASRs consume large amounts of computing power, which may limit their usability on consoles and older PCs.

It is, however, advantageous that games usually require command-and-control (task-oriented) systems, which are easier to implement.

A computer game is a specific environment for a speech recognition system. It can be treated as a special type of a dialog system, in which in some cases the game may actually predict what the user might want to say, knowing the game scenario. This way the system may modify dynamically the language model to reflect that. Care must be taken while implementing such features and changes to the model must remain moderate.

The usage of speech recognition systems in computer games is yet unexplored. There have been a few attempts to utilize ASR systems, particularly in strategy games [9] and flight simulators [10]. ASR systems implemented in these games did work properly, but have not been enthusiastically accepted, mostly because they were tedious to use.

III. Designing the Recognition System for Polish

In this section we will describe specificity of the Polish language, we will present the concept of the computer game *Rally Navigator* which is going to use the ASR interface and we will describe the consecutive steps of designing the speech recognition system for this game.

TABLE I
POLISH PHONEMES INVENTORY, BASED ON [11]

phoneme type	voiceness	phonemes (in SAMPA)
vowels	voiced	a e o u i I
nasals	voiced	m n n' N
plosives	unvoiced voiced	p t k k' b d g g'
fricatives	unvoiced voiced	f s s' S x v z z' Z
affricates	unvoiced voiced	ts ts' tS dz dz' dZ
laterals	voiced	l
liquids	voiced	j w
trills	voiced	r

A. The Polish Language

The Polish language belongs to West-Slavic language family. Spoken language contains 38 phonemes: 6 vowels and 32 consonants [11], out of which fricatives are the most numerous (9 phonemes). Most of the phonemes (plosives, fricatives and affricates) exist in pairs: unvoiced-voiced, e.g. [ts'] - [dz']. Some of these voiced phonemes become unvoiced in an unvoiced context (so called devoicing), and opposite: unvoiced phonemes can become voiced in certain circumstances. This results in pronunciation variation. As for the prosodic features: a melody (pitch contour) of the word is irrelevant to the word's meaning, however the sentence melody sometimes carries semantic information (e.g. can make a question or add an emotional flavor).

¿From the point of view of grammar, the Polish language is complex. It is highly inflective - nouns are inflected according to 7 cases and 2 numbers, verbs are inflected according to gender, tense and number, adjectives and numerals are inflected, too [12]. There are 3 genders in singular and 2 genders in plural. Thanks to the high inflection, the word order in a Polish sentence is rather free, as the function of the word (e.g. whether a noun is a subject or an object) is determined by the form of the word, and not by the position of the word within the sentence. Subjects are often dropped, because they can be deducted from the form of the verb. There are no articles preceding nouns or any other parts of speech.

These features make continuous speech recognition for Polish quite a demanding challenge. Lack of articles makes detecting nouns difficult. High inflection sometimes causes a single word to have dozens of forms, which often sound similarly. Loose word order makes it very difficult to create a good language model. Pronunciation variation, which can be helpful in speech synthesis [13], disturbs speech recognition. Luckily these problems are less severe if we consider a small vocabulary recognition, which is usually the case for a computer game.

B. Concept of the Computer Game

Our main aim while inventing the game was to make the ASR system its integral part, not an additional or alternative way of controlling it. We also wanted the game to be original and innovative, possibly giving the player an opportunity to experience something he has not experienced before. We settled on a game we called *Rally Navigator* in which the player would compete in races - not as a driver, but as a navigator. The player's task would be to provide the driver with information about the route and track elements like curves and straights. To make the game more difficult (and the ASR system more complex) we also decided to include speed control and gear switching. The aim of the game is to win the rally. The more precise and well-timed the information supplied by the player, the quicker the car reaches the finish line.

C. Developing the Acoustic Model

The following elements were required to train a new acoustic model:

- audio data with recorded speech;
- transcription of each audio file;
- dictionary with phonetic representations of all words appearing in the transcriptions;
- list of phonemes (and sounds) appearing in the transcriptions.

The amount of audio data required to properly train the model depends on the type of the model. For a simple command-and-control one-speaker system (the one we began from) the amount of data can be fairly low. For multi-speaker systems the amount of required audio increases, and increases even further for dictation purposes.

To reflect the conditions in which the system will be used, the audio signal was recorded in a home environment, using 16 kHz sampling. The speaker read the following:

- 114 specially designed, phonetically balanced CORPORA [14] sentences, which contain all phonemes and all diphones (pairs of phonemes) appearing in the Polish language;
- the same set of sentences, but spoken faster. The reason behind the faster set of data is that we predict that in our game the players will sometimes speak hastily, for example for a sequence of tight curves. The Corpora sentences in total formed 11 minutes of our training data;
- sample commands, which will be used in game, and numbers, which must be correctly recognized for the game to work (curve angles are described with numbers, so are gears and lengths of straight road).

In total, we prepared 25 minutes of audio.

The audio files were then transcribed, including special silence marks for all silent moments appearing in the file. Such marks allow the training algorithm to better align the speech and, in turn, produce better models.

The phonetic dictionary was prepared in such a way that it contained all words with all possible variants of their pronunciation, to take into account pronunciation variability, caused by various speaking manners and the specificity of Polish, described earlier. Careful preparation of phonetic dictionary prevents from incorrect association of a phoneme with audio parameters of a different phoneme which would effect in decreasing the model's accuracy.

The list of sounds contained mostly phonemes, but also sounds like clicking a mouse button or breathing. It is important to include such sounds in the transcriptions (if they occur in audio files) for two reasons: they will not be mistaken with other phonemes during training and the working ASR system may successfully recognize and omit them.

To train our acoustic model we used applications and scripts from SphinxTrain [8], which is a part of CMU Sphinx.

D. Training the Language Model

We decided to use a tool supplied by CMU Sphinx called *Sphinx Knowledge Base Tool* [15], which generates the language model from a list of sentences. The quality of the model therefore depends on how well the list is prepared or, in other words, how well it reflects the commands which will be issued in the system. The generation process itself is simple, the tool notes and counts all appearing n-grams and then converts the results to n-gram language model format compatible with Sphinx. The harder part is creating a good set of sentences. For example, creating a file with all possible commands is actually not a good idea. Longer commands with more parameters would dominate the data making the less complex commands less probable and the model would then not reflect the real probabilities of N-grams in the system. Furthermore, the amount of possible commands is very large, especially if we want to make the system flexible and allow different variations of the same command. One solution would be to repeat the less complex commands in the training file, so that their amount would be similar to the more complex ones. This would make the file enormously big. But what if we wanted to include two commands in a single sentence? We could then make all possible combinations of these commands, but that would cause the file to grow exponentially.

For our model we have decided to take a different approach. We split our commands into at least three word long parts, each part with at most two parameters and generated all possible variations of each of such fragments. Some of these fragments overlapped, so that all possible bigrams and trigrams were included. This way also allowed us to repeat the less-parametrized fragments, since the amount of repetitions did not have to be large. This whole operation caused our training file to become very small, simple and quick to generate. After using the Sphinx Knowledge Base Tool on this file, the resulting model required some fine-tuning. Most importantly all impossible silence-starting and silence-ending N-grams were removed.

The last stage of building the model was the inclusion of, as we called them, *negative n-grams*, which are artificial *n-grams* with near-zero probability. All sequences built from unigrams, even if they are not included in bi- or trigrams, have a greater than zero probability of being recognized. Negative N-grams can be used to disallow word sequences which we consider invalid in our systems, but are sometimes incorrectly recognized by the system. We have identified several such sequences using our tests and included appropriate negative N-grams in our language model.

Our *JSGF* grammar was created manually. First we defined groups of words describing parameters in our commands, like distances, directions and angles. Then we defined constant fragments of commands using these groups, some of these in a few variations, each correct from polish languages point of view. Next step was grouping these fragments into commands, and some commands into sequences of commands. It is important to note, that we have tried to relax the strict commands by making some words optional, and providing alternatives to some of the words. After the grammar was complete we ran the tests, and then repeated the whole process for sentences from the tests which did not match any of the grammars rules.

IV. Results

This section presents the most significant results of the carried out experiments.

The ASR system was tested using a set of 85 recordings, containing commands which are likely to occur in the *Rally Navigator* game. It contained phrases 2-16 words long, total number of tested words was 422. 2 male speakers were recorded, speaker A whose voice was used the acoustic model training, and speaker B used only for testing. Total time of test recordings was 489 s.

Usually when testing ASR system tests, the following metrics are evaluated:

- *substitutions* (Sub) - the number of words which were recognized as other words
- *insertions* (Ins) - the number of words which were wrongly added to the recognized words
- *deletions* (Del) - the number of words which omitted in recognition
- *word error rate* (WER) - the ratio of word recognition errors (such as substitutions, insertions and deletions) against the total number of words;
- *word accuracy* (WA) - the ratio of correctly recognized words against the total number of words; it is strongly related to WER;
- *sentence accuracy* (SA) - the ratio of correctly recognized sentences against the total number of sentences.

Figure 1 shows the recognition performance for various language models. The JSGF grammar yielded the worst results ($WA = 79.15\%, SA = 61.2\%$ for speaker A). More detailed analysis of the recognition logs showed that it worked very well only for short sentences. However, it had trouble correctly interpreting long word sequences, especially if they consisted of more than one short command. The recognizer would in such cases completely ignore the data after the first command, in spite of the used grammar definition which allowed compound commands.

After switching to the N-gram language models, the recognition improved. Even unigram models enabled accuracy better than the one for JSFG, but after adding information about probabilities of bigrams and trigrams, the results improved significantly, yielding word accuracy of 98.58% and sentence accuracy of 92.9%. The use of negative trigrams turned out to

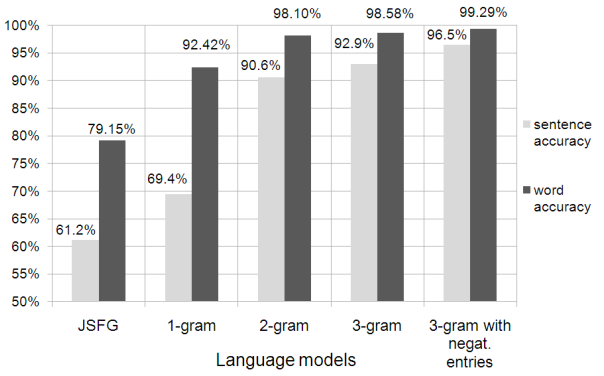

Fig. 1. Word accuracy and sentence accuracy for recognition using various language modeling.

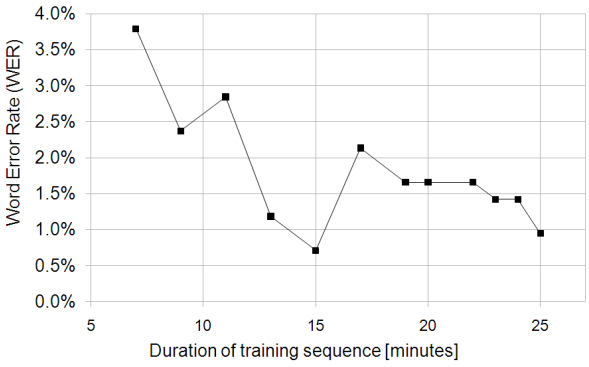

Fig. 2. Word error rate against the duration of audio signal used in the training process.

be a successful move, giving for speaker A the final result of $WA = 99.29\%$ and $SA = 96.5\%$.

Figure 2 displays the performance of recognition when the acoustic models were trained with different size of audio data. The word error rate for speaker A after training the acoustic model with 114 CORPORA sentences (i.e. 7 minutes of recording) was almost 4%, what is considered high, taking into account that it is a small-vocabulary task-oriented recognition. After adding 4 minutes more, containing the same sentences, but uttered faster, WER became slightly below 3% and the sentence accuracy increased from 88.2% to 90.6%. When adding recordings containing numerals and control commands, the performance continued to improve until the training set contained 15 minutes of recordings, where WER equaled 0.7%. Sentence accuracy at this moment reached 97.6%, what was considered a satisfactory result. Further enlargement of audio data resulted in worsening of WER up to 2.1%, due to slight increase of substitutions and deletions. Training using 25 minutes of recordings yielded again a low value of WER - 0.9%.

It is noteworthy that not every word deletion, substitution or insertion resulted in a wrong command. E.g. omitting *w* (here meaning 'to') in a sentence *skrec w prawo* ('turn to the right') caused the command change into 'turn right', being actually the same. So the semantic accuracy was even higher than the sentence accuracy.

Table II gives information about the recognition performance both for speakers A and B. When challenging the system with the voice of speaker B, who was not used to create the acoustic model, the ASR system was able to recognize correctly 72.9% sentences at the WER rate of 8.3%. After adapting the models with 10 CORPORA sentences of the speaker B, WER even slightly increased, but adaptation using 20 sentences decreased WER down to 6.62%. Further enlargement of the adaptation session was steadily improving the recognition performance, but 80 sentences were required to make WER as low as 1.42%, with sentence accuracy of almost 93%.

TABLE II
RECOGNITION RESULTS FOR SPEAKERS A AND B WITH VARIOUS NUMBERS OF SENTENCES USED FOR ACOUSTIC MODEL ADAPTATION

speaker / adaptation	WER	WA	SA	Sub	Ins	Del
A	0.9%	99.29%	96.5%	2	1	1
B, no adapt.	8.3%	92.91%	72.9%	20	5	10
B, 10 sentences	9.69%	90.78%	69.4%	28	2	11
B, 20 sentences	6.62%	94.09%	77.6%	18	3	7
B, 30 sentences	5.44%	95.27%	81.2%	13	3	7
B, 40 sentences	4.49%	95.98%	82.4%	9	2	8
B, 60 sentences	3.55%	96.93%	84.7%	6	2	7
B, 80 sentences	1.42%	98.58%	92.9%	3	0	3

V. CONCLUSION AND FUTURE WORKS

This paper described the process of designing a task-oriented continuous speech recognition system for Polish, based on CMU Sphinx4, to be used in a computer game called *Rally Navigator*. We presented the steps undertaken to create the acoustic model and the language model, using both the grammar and the statistic N-gram model.

As for the language model, we showed that the best results were achieved if the statistic trigram model was used. We improved it by adding negative trigrams, what decreased the number of misrecognized words.

Initial experiments showed that the audio material as short as 15 minutes is enough to produce a highly effective single-speaker command-and-control ASR system, providing the sentence recognition accuracy of 97.6%. What was expected, such a model required adaptation for another speaker. 20 sentences of the new speaker enabled partial adaptation of ASR, so that it reached word accuracy of 94.09%, but better results (WER below 4%) were obtained if the model was adapted with 60 or 80 sentences. Obviously using such a long audio material for adaptation of each new user would be impractical, so the acoustic model needs to be improved. Training the acoustic model on a large multi-speaker speech corpus of the Polish language is planned as the next step.

REFERENCES

[1] B. Ziolko, S. Manandhar, R. C. Wilson, M. Ziolko, J. Galka, *Application of HTK to the Polish Language*, In Proceedings of IEEE International Conference on Audio, Language and Image Processing ICALIP2008, Shanghai, 2008, pp.1759-1764.

[2] M. Szymanski, J. Ogorkiewicz, M. Lange, K. Klessa, S. Grocholewski, G. Demenko, *First evaluation of Polish LVCSR acoustic models obtained from the JURISDIC database*, Speech and Language Technology, vol. 11, 2008.

[3] S. Furui, *Selected topics from 40years of research in speech and speaker recognition*, Interspeech 2009, Brighton UK, 2009.

[4] L. Rabiner and B.-H. Juang, *Historical Perspective of the Filed of ASR/NLU*, Springer Handbook of Speech Processing, ed. J.Benesty et al.,Berlin Heidelberg: Springer-Verlag, 2008.

[5] S. Young, *HMMs and Related Speech Recognition Technologies*, Springer Handbook of Speech Processing, ed. J.Benesty et al.,Berlin Heidelberg: Springer-Verlag, 2008.

[6] Sun Microsystems, *Java Speech Grammar Format*, version 1, 1998, http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/

[7] Sun Microsystems, *Java Speech API*, 1998, http://java.sun.com/products/java-media/speech/

[8] Carnegie Mellon University, *SphinxTrain*, http://cmusphinx.sourceforge.net/wiki/sphinxtrainwalkthrough

[9] Ubisoft Shanghai, *Tom Clancy's Endwar*, 2009, http://endwargame.us.ubi.com/

[10] Ubisoft Romania, *Tom Clancy's H.A.W.X*, 2009, http://www.hawxgame.com/

[11] W. Jassem, *Podstawy fonetyki akustycznej* (in Polish), Warsaw, Poland : PWN, 1973.

[12] Ron F. Feldstein, *A Concise Polish Grammar*, Duke University, Durham, USA: Slavic and Eurasian Language Resource Center, 2001.

[13] A. Janicki, P. Meus, M. Topczewski, *Taking Advantage of Pronunciation Variation in Unit Selection Speech Synthesis for Polish*, 3rd International Symposium on Communications, Control and Signal Processing (ISCCSP 2008), St.Julians, Malta, 2008

[14] S. Grocholewski, *CORPORA—Speech Database for Polish Diphones*, 5th European Conference on Speech Communication and Technology Eurospeech '97, Rhodes, Greece, 1997.

[15] Alex Rudnicky, *Sphinx Knowledge Base Tool*, 2010, http://www.speech.cs.cmu.edu/tools/lmtool.html