TUNING SPHINX TO OUTPERFORM GOOGLE'S SPEECH RECOGNITION API*

Patrick Lange[1,2,3] *and David Suendermann-Oeft*[1]

[1]*DHBW, Stuttgart, Germany*
[2]*Linguwerk GmbH, Dresden, Germany*
[3]*Staffordshire University, Stafford, UK*

patrick.lange@linguwerk.de david@suendermann.com

Abstract: In this paper, we investigate whether the open-source speech recognizer Sphinx can be tuned to outperform Google's cloud-based speech recognition API in a spoken dialog system task. According to this target domain, we use data from CMU's Let's Go bus information system comprising 258k utterances of telephony speech recorded in the bus information dialog system of Pittsburgh. By training a domain-specific language model on the aforementioned corpus and tuning a number of Sphinx's parameters, we achieve a WER of 51.2%. This result is significantly lower than the one produced by Google's speech recognition API whose language model is built on millions of times more training data.

1 Introduction

Cloud computing has become one of the most popular topics in the IT world during the past decade. One of the biggest benefits of a cloud-based service is the higher computational power of servers compared to customer devices. Among other applications, speech recognizers can greatly benefit from this advantage and are now available as cloud-based services offered by several vendors.

Mobile devices like smartphones usually have less computation power than non-mobile ones but are often equipped with wireless Internet connection. Since cloud computing shifts the requirement of devices from powerful hardware to reliable network connectivity, it is particularly appropriate to be used inside mobile applications. Further considering that user input on mobile devices is rather inconvenient due to the lack of a full hardware keyboard, it becomes obvious that mobile devices like smartphones are a good platform for voice-controlled applications utilizing cloud-based speech recognition.

Two of the biggest companies building voice-powered applications are Google and Microsoft. In the years 2007 to 2009, they have been providing toll-free alternatives to 4-1-1, a costly directory assistant by phone companies in the USA and Canada. These free services were commonly known as GOOG-411 [1] and BING-411. However, due to the above mentioned reasons, smartphones have proven to be better suited for voice search applications, and both GOOG-411 and BING-411 were discontinued a few years after their launch. Instead, both vendors are now focusing on providing mobile and voice-controlled versions of their search engines Google and Bing for various smartphone operating systems. Data collected during the operation of GOOG-411 and BING-411 served for bootstrapping acoustic and language models deployed in the voice search engines.

*This work was supported by a grant from the Baden-Wuerttemberg Ministry of Science and Arts as part of the research project OASIS—Open-Source Automatic Speech Recognition In Smart Devices

Although the recognizers of both companies are used by millions of users around the world, they come with some major drawbacks regarding 3rd-party applications. First of all, to the best of our knowledge, scientific evaluations on their performance are still outstanding. Secondly, the cloud-based recognizers cannot be customized regarding the used models or the internal parameters. Thirdly, no official APIs are provided to 3rd-party organizations to develop their own voice-powered applications. Finally, they are proprietary software and not open-source.

In this paper, we answer the question whether it is possible to achieve similar or even better performance with an open-source speech recognizer which is customizable and provides an API usable by 3rd-party application developers.

2 State of the Art

Besides the above mentioned vendors Google and Microsoft, there are many other companies developing speech recognizers. In this section, we present some of the most relevant commercial and open-source state-of-the-art products and motivate why we have chosen to compare Google's speech recognition API and Sphinx.

2.1 Commercial recognizers

One of the most popular speech-powered applications is Apple's personal assistant Siri [2]. First launched as a smartphone app, in October 2011, Siri was integrated as a key feature of the iPhone 4S. Ever since, Siri has regularly appeared in the media as a killer app, including New York Times and CNN. It is probably the most successful application combining speech recognition with understanding of natural language, context, and reasoning.

Another widely known application featuring highly sophisticated natural language and speech processing technology is the question answering system IBM Watson [3] famous for beating the two ultimate champions in the quiz show Jeopardy. IBM has announced that Watson will be made available as a customer service agent. The "Ask Watson" feature will be accessible through different channels such as web chat, email, smartphone apps, and SMS. Furthermore, speech recognition is to make Watson accessible by voice.

As a reaction to the succes of Apple's Siri, in 2012, Google introduced the personal assistant Google Now in Android 4.1 [4]. It focuses less on understanding natural language, context, and reasoning but more on predicting search queries and preemptively showing results. Hence, the biggest difference between Google Now and Siri is that the former is to proactively offer information when appropriate. For example, in a train station, Google Now shows arrival and departure times without user request. Besides Google Now on smartphones, Google has released a web-based speech recognition API. This Javascript library provides speech recognition and synthesis functionality and is available starting in Chrome version 25. Google's speech recognition API allows developers to create voice-controlled Chrome add-ons and websites.

Another vendor of cloud-based as well as local speech recognizers is Nuance Communications. Besides developing the popular speech-to-text desktop software Dragon NaturallySpeaking, Nuance also offers a Siri alternative for Android and iOS called Vlingo Virtual Assistant [5]. Furthermore, recently, their CEO Paul Ricci disclosed that Nuance is powering the cloud-based speech recognition behind Apple's Siri [6].

2.2 Open-source speech recognizers

One of the most popular open-source speech recognizers is the Hidden Markov Model Toolkit (HTK) [7]. Although it can be used to train any type of hidden Markov model (HMM), HTK is primarily applied to speech recognition research. It consists of a set of libraries and tools

written in C. The tools provide facilities for speech analysis, HMM training, testing and result analysis. HTK can be used to build recognizers based on huge corpora including thousands of hours of speech [8].

Another open-source toolkit is SRI's Decipher [9]. It can recognize natural, continuous speech without requiring the user to train the system in advance. It can be distinguished from other speech recognizers by its detailed modeling of variations in pronunciation and its robustness to background noise and channel distortion. These features let Decipher excel in recognizing spontaneous speech of different dialects and makes recognition less dependent on the idiosyncrasies of different microphones and acoustic environments.

Kaldi is a free, open-source toolkit for speech recognition developed at Microsoft Research [10]. It provides a speech recognition system based on finite-state transducers together with detailed documentation and scripts for building complete recognition systems. Kaldi is written in C++ and is released under the Apache License v2.0 which is intrinsically nonrestrictive, making it suitable for a wide community of users.

MIT's WAMI toolkit provides a framework for developing, deploying, and evaluating web-accessible multimodal interfaces in which users interact using speech, mouse, pen, or touch [11]. The toolkit makes use of modern web-programming techniques, enabling the development of browser-based applications which rival the quality of traditional native interfaces, yet are available on a wide array of Internet-connected devices. Additionally, it provides resources for collecting, transcribing, and annotating data from multimodal user interactions.

Sphinx is the name of a group of speech recognition systems developed at Carnegie Mellon University. Among these systems are Sphinx 3 [12], a decoder for speech recognition written in C, Sphinx 4 [13], a modified recognizer written in Java, and Pocketsphinx [14], a lightweight recognizer library written in C.

RWTH Automatic Speech Recognition (RASR) is written in C++ containing a speech recognizer and tools for the development of acoustic models [15]. Speech recognition systems developed using this framework have shown high performance in multiple international research projects and at regular evaluations, e.g. in the scope of the DARPA project GALE [16].

2.3 Experimental systems

Our aim is to show that open-source speech recognizers can be an alternative to cloud-based commercial speech recognizers that are trained on humongous amounts of data. From the list of available open-source speech recognizers provided in Section 2.2, we chose CMU's Sphinx. Main reason for this choice was the ultimate goal of this research to optimize recognition performance in the scope of distributed industry-standard-compliant open-source dialog systems [17, 18, 19]. In addition to Sphinx's support of the ARPA language model format [20] and the Java Speech Grammar Format (JSGF), there exists also a Media Resource Control Protocol (MRCP) port [21] which we are using within a complete VoiceXML and SIP-based spoken dialog system. The architecture of this system is depicted in Figure 1. Moreover, Sphinx's license is least restrictive and permits even commercial use of the code which makes it very appealing for industrial partners of the authors. For this paper's experiments, we used Pocketsphinx Version 0.8. Pocketshinx is still in an experimental state and under development.

We chose Google's speech recognition API because it is the only industrial cloud-based speech recognizer with a freely accessible API.

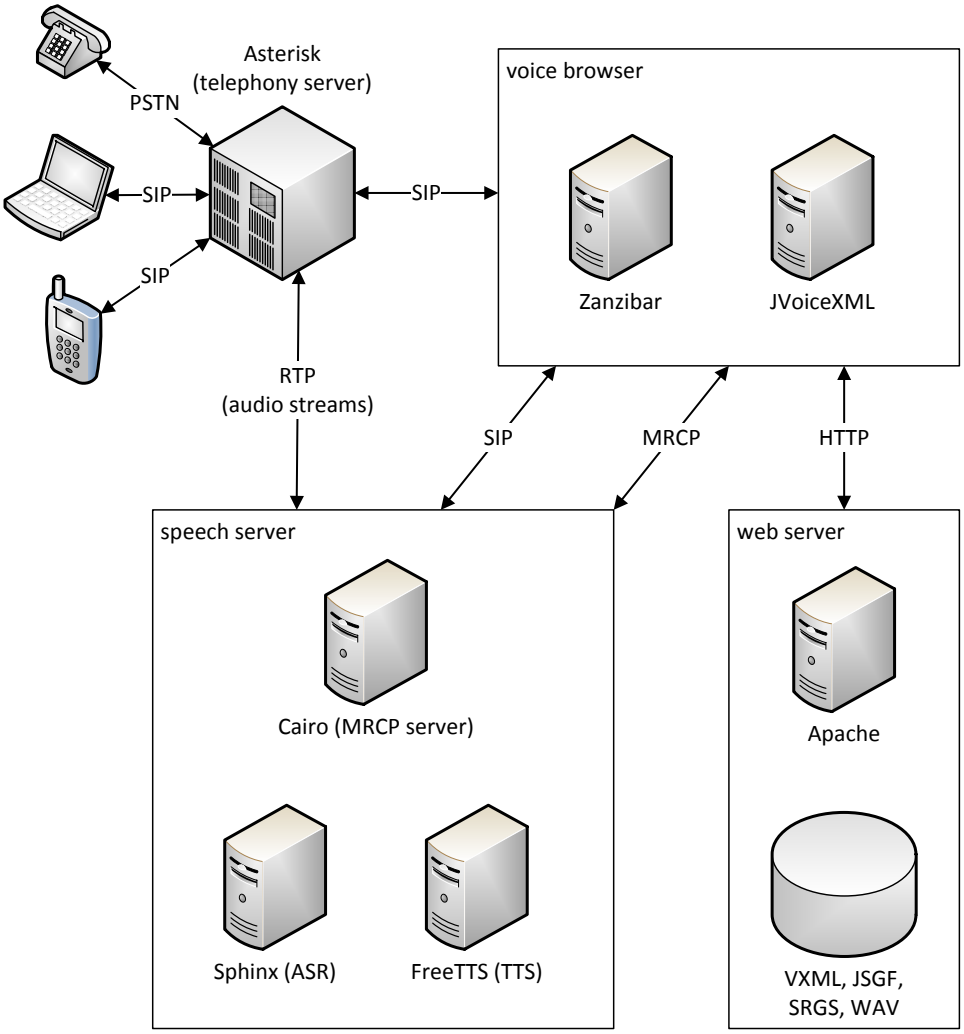

Figure 1 - Architecture of a Distributed Industry-Standard-Compliant Open-Source Spoken Dialog System

3 Corpus

Our experiments are based on data collected within the Let's Go bus information system in the years 2008 and 2009 [22]. This spoken dialog system created at Carnegie Mellon University provides bus schedule information to the Pittsburgh population during off-peak times.

The entire corpus consists of about 250k utterances which have been fully transcribed. We randomly selected 3000 sentences as test and 3000 sentences as development data. The remaining sentences were used to train a domain-specific language model.

To estimate the difficulty of the recognition task, we counted OOV words and computed the 3-gram perplexity for each set. Utterances which could not be transcribed are tagged as NON_UNDERSTANDABLE in the corpus. For our experiments, we replaced the NON_UNDERSTANDABLE tags by empty transcriptions. See Table 1 for detailed corpus statistics.

4 Experiments

As motivated in the introduction, the ultimate goal of our experiments was to find out whether the open-source speech recognizer Sphinx is able to outperform Google's speech recognition API (the "baseline").

Table 1 - Let's Go Bus Information System Corpus Statistics

Train	Sentences	251 653
	Words	466 695
	Vocabulary Words	4 107
	Singletons	1819
	NON_UNDERSTANDABLE	43 834 (17%)
Development	Sentences	3 000
	Words	5 852
	OOVs	25
	Perplexity	126.57
	NON_UNDERSTANDABLE	502 (16%)
Test	Sentences	3 000
	Words	5885
	OOVs	16
	Perplexity	131.19
	NON_UNDERSTANDABLE	489 (16%)

As a first step, we determined the baseline performance on the test set. To this end, we converted the raw audio files into the *flac* format supported by the Google speech recognition API and achieved a word error rate (WER) of 54.6% (3211 errors out of 5885 words). Short utterances (like *yes* or *no*) or times or bus numbers were recognized very well, but Google's speech recognition API struggled with grammatically wrong phrases and addresses. For example, Google recognized

oakland knox avenue in homewood knoxville last bus

as

i cannot have another else like knoxville stuff.

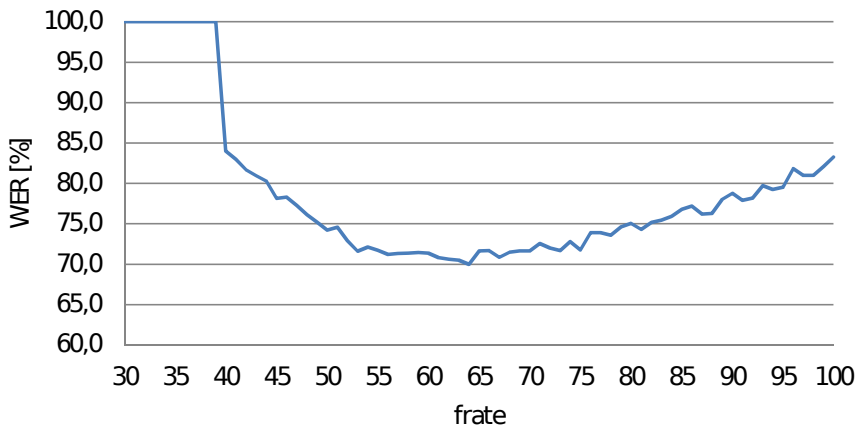

Figure 2 - Results: Frate Optimization First Iteration

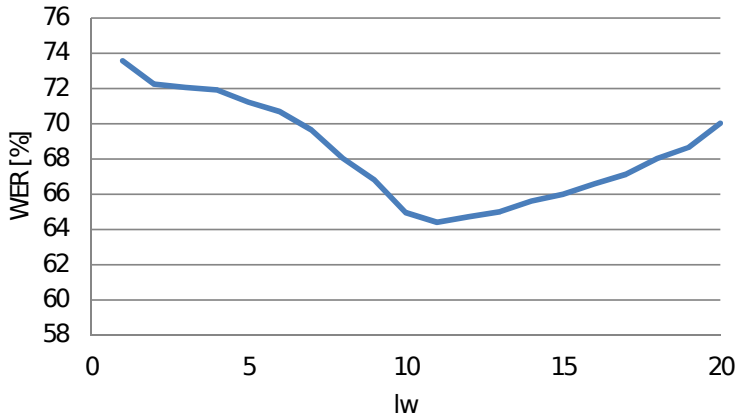

Figure 3 - Results: Lw Optimization First Iteration

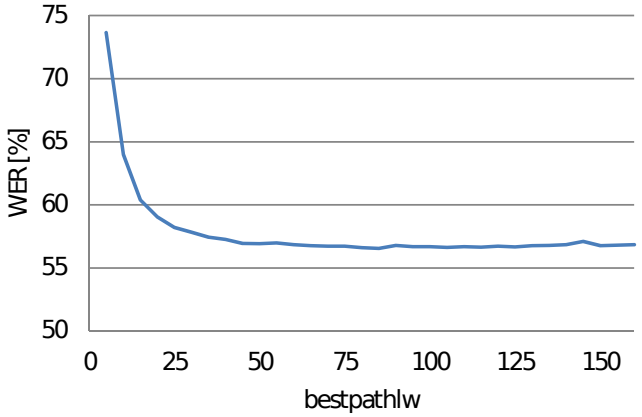

Figure 4 - Results: Bestpathlw Optimization First Iteration

After establishing the baseline, we started experimentation with Sphinx using a set of default models and settings. In all the following experiments, we used the Hub-4 acoustic model and dictionary [23]. All tuning experiments were run exclusively on the development set to prevent optimization on the test set. The Hub-4 language model resulted in a WER of 171% on the development set. Also WSJ-5k and WSJ-20k language models produced WERs over 100%. The reason was obviously that vocabulary and n-gram distributions did not match those of the Let's Go bus information system domain.

To overcome this issue, we created a domain-specific language model using the language model toolkit CMUclmtk and the data of the training set (466k word tokens). This customized language model improved the WER to 83.3%.

To further increase the performance of Sphinx, we now looked into optimizing recognizer parameters. After running a number of informal tests, we chose to include the following parameters in formal experiments: frate, lw, bestpathlw, fwdflat, wip, pip, and uw. See Table 2 for more information about these parameters.

Although the most rewarding way to optimize parameters would be to brute-force-test every possible combination of parameter values, this approach would be too time consuming regarding the wide range of the values. Therefore, we assumed independence of the parameters and, hence, that parameters can be optimized individually. As this assumption is probably not entirely valid, we decided to repeat the cycle of individual parameter optimization twice.

In the first iteration, we reduced the WER to 54.4% which is an improvement by 37.4% com-

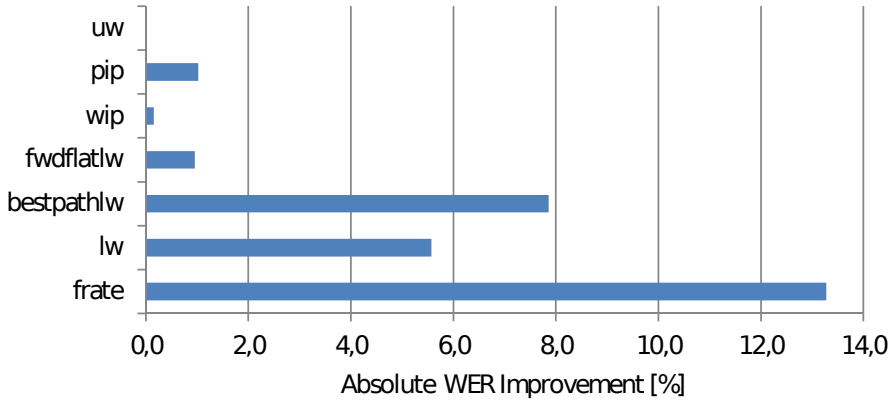

Figure 5 - Absolute WER Improvement First Iteration

Table 2 - Parameters Optimized During The Experiment

Name	Default value	Description
frate	100	Frame rate
lw	6.5	Language model probability weigth
bestpathlw	9.5	Language model probability weight for DAG best path search
fwdflatlw	8.5	Language model probability weight for flat lexicon decoding
wip	0.65	Word insertion penalty
pip	1	Phone insertion penalty
uw	1	Unigram weigth

pared to the custom language model with default parameter values. Figure 5 shows that the main improvement was gained through optimizing frate, lw, and bestpathlw. Figures 2, 3, and 4 show how the WER depends on the values of these three parameters during the first optimization cycle. While frate and lw exhibit clearly visible local optima, the local minima of bestpathlw are not very obvious. Rather, it looks like that any sufficiently high value is acceptable meaning that the language model score is overly important during the directed acyclic graph (DAG) best path search.

In the second iteration, we were able to decrease the WER by another 2.8% which is relatively small compared to the first iteration. Again, frate was the parameter with the highest impact followed by lw, bestpathlw, and wip.

Since the WER improved only by 2.8% during the second iteration, we decided to stop tuning parameters at this point. We believed it to be unlikely that further (slim) improvements would populate to the test set but would rather overfit the development set.

In the last step, we applied the optimized parameter configuration to the test set achieving a WER of 51.2% (3015 errors out of 5885 words). As aforementioned, Google's speech recognition API achieved a WER of 54.6% (3211 errors out of 5885 words). With a p-value of 0.0003 as determined by a two-sample pooled t-test, this improvement is highly statistically significant.

Table 3 - Parameter Optimization Results

	WER	Total error	Parameter	Value
default lm	171%	10007		
custom lm	83.3%	4872		
first iteration	70%	4 095	frate	64
	64.4%	3 769	lw	11
	56.6%	3 309	bestpathlw	85
	55.6%	3 253	fwdflatlw	13
	55.4%	3 244	wip	0.7
	54.4%	3 184	pip	28
	54.4%	3 184	uw	1
second iteration	53.1%	3106	frate	78
	52.6%	3078	lw	10
	52.2%	3053	bestpathlw	390
	52.2%	3053	fwdflatlw	13
	51.9%	3037	wip	0.07
	51.9%	3037	pip	28
	51.6%	3021	uw	2

5 Conclusion and future work

Using the Let's Go bus information system corpus, we were able to show that the open-source speech recognizer Sphinx is capable of outperforming Google's speech recognition API. The former achieved 51.2% WER on a test set comprising 3000 sentences. This result outperformed Google by 3.3%, being highly statistically significant. The most effective parameters tuned during Sphinx's optimization cycles were frate, lw, and bestpathlw.

Even though improvements in the Let's Go bus information system domain were significant, the overall WER is still very high. One of the reasons is that we did not distinguish individual contexts within the spoken dialog system when training the language model. For instance, in a confirmation context, the user input (mainly *yes* or *no*) looks entirely different from a bus destination collection context which is dominated by names of bus stops. We have shown in [24] that dividing data into micro-contexts has a strongly positive impact on performance.

Another weak point of our current setting is that we were using default acoustic and lexical models. Building, or at least tuning, the acoustic model on the collected telephone speech considering the typical Pittsburghian caller and updating the lexicon with relevant vocabulary should improve performance even more, as should the application of speaker adaptation and normalization techniques [25]. We also suspect that a brute-force approach to parameter tuning is more likely to find the global optimum than the greedy two-iteration approach.

As motivated before, we are presently applying the knowledge gained in this tuning experiment to building open-source industry-standard-compliant spoken dialog systems [19]. In order to supply the best possible performance to our systems, in addition to the aforementioned measures, we are also considering to compare Sphinx with other free speech recognizers and possibly apply system combination techniques (see examples in Section 2.2).

References

[1] M. Bacchiani, F. Beaufays, J. Schalkwyk, M. Schuster, and B. Strope, "Deploying GOOG-411: Early Lessons in Data, Measurement, and Testing," in *Proc. of the ICASSP*, Las Vegas, USA, 2008.

[2] N. Winarsky, B. Mark, and H. Kressel, "The Development of Siri and the SRI Venture Creation Process," SRI International, Menlo Park, USA, Tech. Rep., 2012.

[3] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty, "Building Watson: An Overview of the DeepQA Project," *AI Magazine*, vol. 31, no. 3, 2010.

[4] G. de Melo and K. Hose, *Advances in Information Retrieval*. New York, USA: Springer, 2013.

[5] Z. Liu and M. Bacchiani, "TechWare: Mobile Media Search Resources," *Signal Processing Magazine*, vol. 28, no. 4, 2011.

[6] P. Ricci, "Nuance CEO Paul Ricci: The Full D11 Interview (Video)," `http://allthingsd.com/20130530/nuance-ceo-paul-ricci-the-full-d11-interview-video/`, May 2013.

[7] S. Young, G. Evermann, M. Gales, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book, Version 3.4*. Cambridge, UK: Cambridge University, 2006.

[8] G. Evermann, H. Chan, M. Gales, B. Jia, D. Mrva, P. Woodland, and K. Yu, "Training LVCSR Systems on Thousands of Hours of Data," in *Proc. of the ICASSP*, Philadelphia, USA, 2005.

[9] H. Murveit, M. Weintraub, and M. Cohen, "Training Set Issues in SRI's DECIPHER Speech Recognition System," in *Proc. of the DARPA Speech and Natural Language Workshop*, Hidden Valley, USA, 1990.

[10] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The KALDI Speech Recognition Toolkit," in *Proc. of ASRU*, Hawaii, USA, 2011.

[11] A. Gruenstein, I. McGraw, and I. Badr, "The WAMI Toolkit for Developing, Deploying, and Evaluating Web-Accessible Multimodal Interfaces," in *Proc. of the ICMI*, Chania, Greece, 2008.

[12] K. Seymore, S. Chen, S. Doh, M. Eskenazi, E. Gouvea, B. Raj, M. Ravishankar, R. Rosenfeld, M. Siegler, R. Stern, and E. Thayer, "The 1997 CMU Sphinx-3 English Broadcast News Transcription System," in *Proc. of the DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, USA, 1998.

[13] P. Lamere, P. Kwok, E. Gouvea, B. Raj, R. Singh, W. Walker, M. Warmuth, and P. Wolf, "The CMU SPHINX-4 Speech Recognition System," in *Proc. of the ICASSP'03*, Hong Kong, China, 2003.

[14] D. Huggins-Daines, M. Kumar, A. Chan, A. Black, M. Ravishankar, and A. Rudnicky, "Pocketsphinx: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices," in *Proc. of the ICASSP*, Toulouse, France, 2006.

[15] D. Rybach, S. Hahn, P. Lehnen, D. Nolden, M. Sundermeyer, Z. Tüske, S. Wiesler, R. Schlüter, and H. Ney, "RASR–The RWTH Aachen University Open Source Speech Recognition Toolkit," in *Proc. of ASRU*, Hawaii, USA, 2011.

[16] D. Rybach, S. Hahn, C. Gollan, R. Schlüter, and H. Ney, "Advances in Arabic Broadcast News Transcription at RWTH," in *Proc. of the ASRU*, Kyoto, Japan, 2007.

[17] D. Suendermann, *Advances in Commercial Deployment of Spoken Dialog Systems*. New York, USA: Springer, 2011.

[18] A. Schmitt, M. Scholz, W. Minker, J. Liscombe, and D. Suendermann, "Is It Possible to Predict Task Completion in Automated Troubleshooters?" in *Proc. of the Interspeech*, Makuhari, Japan, 2010.

[19] T. von Oldenburg, J. Grupp, and D. Suendermann, "Towards a Distributed Open-Source Spoken Dialog System Following Industry Standards," in *Proc. of the ITG*, Braunschweig, Germany, 2012.

[20] D. Gibbon, R. Moore, and R. Winski, *Handbook of Standards and Resources for Spoken Language Systems*. New York, USA: Mouton de Gruyter, 1997.

[21] D. Prylipko, D. Schnelle-Walka, S. Lord, and A. Wendemuth, "Zanzibar OpenIVR: An Open-Source Framework for Development of Spoken Dialog Systems," in *Proc. of the TSD*, Pilsen, Czech Republic, 2011.

[22] A. Raux, D. Bohus, B. Langner, A. Black, and M. Eskenazi, "Doing Research on a Deployed Spoken Dialogue System: One Year of Let's Go! Experience," in *Proc. of the Interspeech*, Pittsburgh, USA, 2006.

[23] K. Seymore, S. Chen, M. Eskenazi, and R. Rosenfeld, "Experiments in Evaluating Interactive Spoken Language Systems," in *Proc. of the DARPA Speech Recognition Workshop*, Washington, D.C., USA, 1997.

[24] D. Suendermann, J. Liscombe, K. Evanini, K. Dayanidhi, and R. Pieraccini, "From Rule-Based to Statistical Grammars: Continuous Improvement of Large-Scale Spoken Dialog Systems," in *Proc. of the ICASSP*, Taipei, Taiwan, 2009.

[25] D. Sündermann, A. Bonafonte, H. Ney, and H. Höge, "Time Domain Vocal Tract Length Normalization," in *Proc. of the ISSPIT*, Rome, Italy, 2004.