HALEF: an open-source standard-compliant telephony-based modular spoken dialog system – A review and an outlook

David Suendermann-Oeft[†], Vikram Ramanarayanan[†], Moritz Teckenbrock[‡], Felix Neutatz[‡] and Dennis Schmidt[‡]

Abstract We describe completed and ongoing research on HALEF, a telephony-based open-source spoken dialog system that can be used with different plug-and-play back-end modules. We present two examples of such a module, one which classifies whether the person calling into the system is intoxicated or not, and the other a question answering application. The system is compliant with World Wide Web Consortium and related industry standards while maintaining an open codebase to encourage progressive development and a common standard testbed for spoken dialog system development and benchmarking. The system can be deployed toward a versatile range of potential applications, including intelligent tutoring, language learning and assessment.

1 Introduction

Spoken dialog systems (SDSs) have witnessed a steep increase in usage over the last five years thanks to improvements in speech recognition performance, the availability of smart devices, ubiquitous high-speed internet and cloud computing, progress in developing standards, and the emergence of crowdsourcing for speech applications, among other factors [26]. While commercially deployed industrial vendors (such as Cisco, Nuance, Avaya, Genesys, Microsoft, Voxeo, etc.) tend to concentrate on dialog managers with finite-state call flows, rule-based grammars for speech recognition, large volumes of call data and more or less standardized interfaces and protocols, academic research (see for example [3, 17, 28, 2, 30, 1]) has adopted a more long-term approach, focusing on statistically-trained dialog managers and spoken language understanding modules, smaller-sized datasets and proprietary interfaces [15, 25]. Academia has also been more open to publishing codebases, software and research results as compared to industry. Having said that, a large percentage of practical, deployed solutions are industry-based and are, as such, proprietary. Although there are standard protocols in place to develop SDS solutions which many industrial systems adhere to, their system implementations and software components are often different, which makes benchmarking of systems relative to each

[†] Educational Testing Service (ETS) Research, San Francisco, CA
`Email: <suendermann-oeft,vramanarayanan>@ets.org`

[‡] DHBW, Stuttgart, Germany

Part of the work described here was completed when the first author was at DHBW Stuttgart.

other a difficult task. An open-source implementation that is compliant with W3C standards would be a positive step towards a working solution to this issue.

It is further important to note the utility of having a telephony-based, modular SDS architecture. Although there exist many open-source SDS implementations in the academic world, most of these are not telephony-based. This means that most of these systems typically require installation of a software interface on a local workstation or computer. A telephony-based SDS setup would allow people to call into and access the SDS without any software installation overhead. Furthermore, by making such a system modular, we can individually optimize the different components of the system – telephony server, speech server, voice browser and web server. However, there are currently almost no systems that offer all of the above advantages [25] to our knowledge (the CMU Olympus [2] and the ALEX dialog frameworks [11] are two systems that come close to being exceptions, but that they are standard-compliant is not clear).

To address these shortcomings, we developed HALEF – a telephony-based, modular, open-source, standard-compliant spoken dialog system. The primary objective of this paper is to describe the current state of the HALEF system and discuss how various back-end applications can be integrated within the SDS framework. HALEF is written primarily in Java and leverages a number of open-source tools in a distributed framework for scalability.

SDS frameworks can be deployed to suit a wide range of applications such as directory services [7], technical troubleshooting [20], intelligent tutoring [8] or computer-assisted language learning [24, 29]. In this paper, we present a couple of such applications, including that of a plug-and-play module for alcoholic state classification and a question answering service. Note however that the purpose of this paper is *not* to further the state of the art in alcohol classification or question answering applications, but to demonstate how a working classifier/application can be incorporated into the HALEF framework as an independent plug-and-play module.

The rest of the paper is organized as follows: Section 2 describes the basic architecture and components of the HALEF spoken dialog system. We then describe example applications of the HALEF SDS to alcoholic state classification and question answering in Section 3. Finally we conclude with a discussion of ongoing and future research into the system in Section 4.

2 HALEF System Description

The HALEF (Help Assistant–Language-Enabled and Free) framework leverages different open-source components to form an SDS framework that is modular and industry-standard-compliant: Asterisk, a SIP- (Session Initiation Protocol) and PSTN- (Public Switched Telephone Network) compatible telephony server [13]; JVoiceXML, an open-source voice browser that can process SIP traffic [21] via a voice browser interface called Zanzibar [16]; Cairo, an MRCP (Media Resource Control Protocol) speech server, which allows the voice browser to initiate SIP or RTP (Real-time Transport Protocol) connections from/to the telephony server [16]; the Sphinx automatic speech recognizer [12]; Festival [27] and Mary [22] – text-to-speech synthesis engines; and an Apache Tomcat-based web server that can host

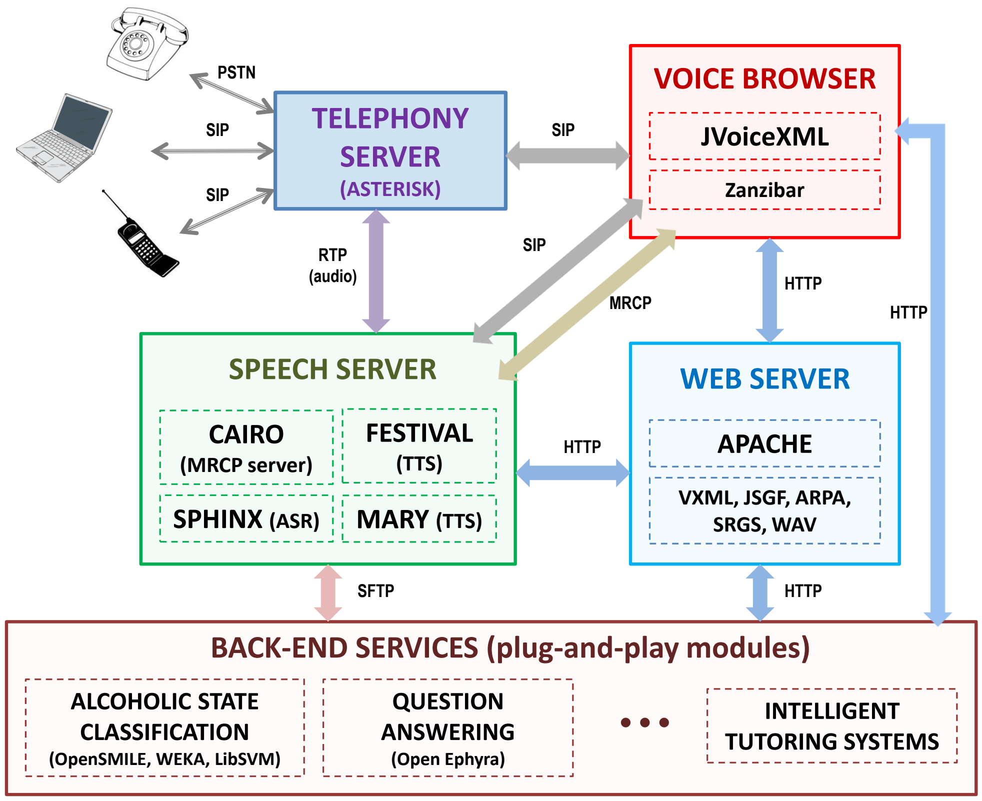

Fig. 1 System architecture of the HALEF spoken dialog system depicting the various modular open-source components.

dynamic VoiceXML (VXML) pages and serve media files such as grammars[2] and audio files to the voice browser. Figure 1 schematically depicts the main components of the HALEF system. Note that unlike a typical SDS, which consists of sequentially-connected modules for speech recognition, language understanding, dialog management, language generation and speech synthesis, in HALEF some of these are grouped together forming independent blocks which are hosted on different virtual machines in a distributed architecture. For further details on the individual blocks as well as design choices, please refer to [14]. In this framework, one can serve different back-end applications as standalone web services on a separate server. Incorporating the appropriate start URL (Universal Resource Locator) of the web service in the VXML input code that the voice browser interprets will then allow the voice browser to trigger the web application at the appropriate point in the callflow. The web services in our case typically take as input any valid HTTP-based

[2] Popular grammar formats include JSGF (Java Speech Grammar Format), SRGS (speech recognition grammar specification) and ARPA (Advanced Research Projects Agency) formats.

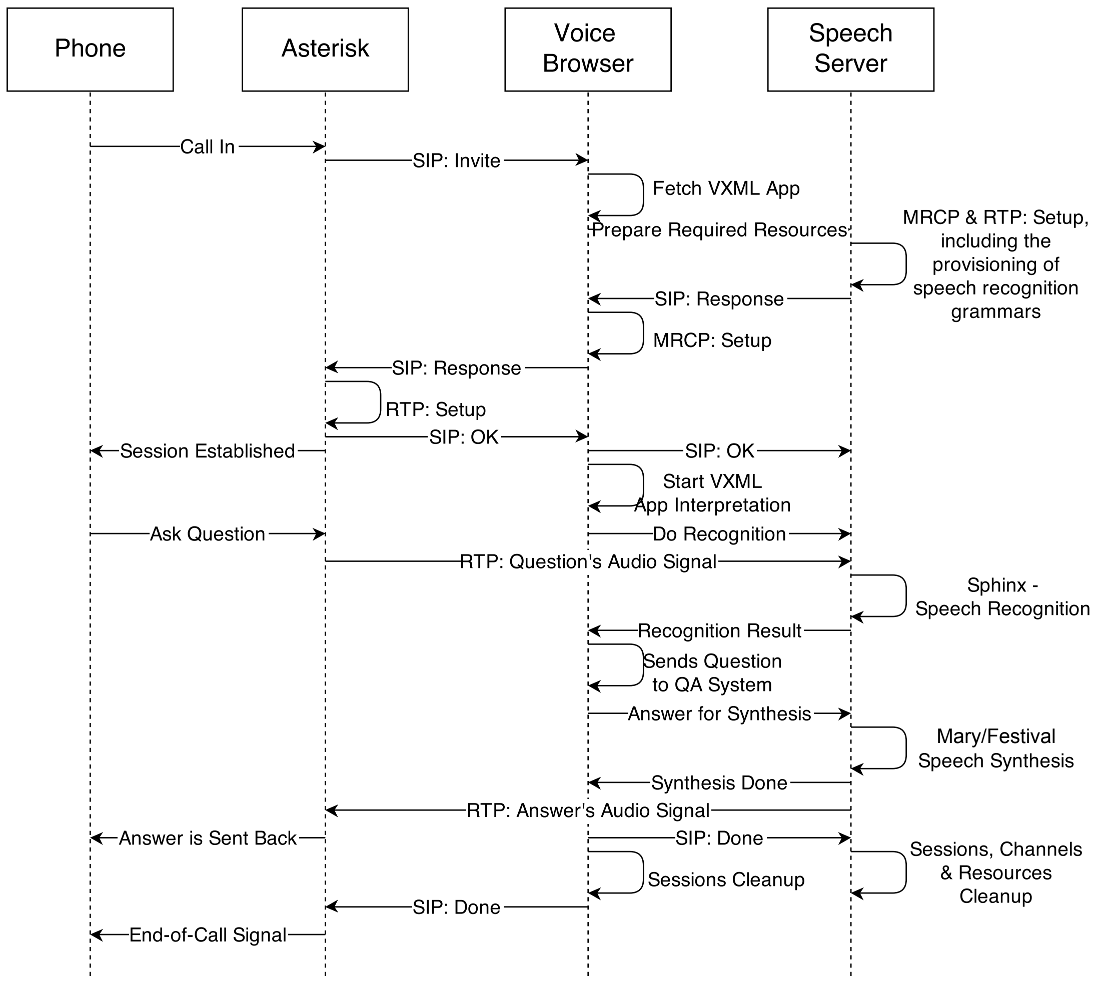
Phone
Asterisk
Voice Browser
Speech Server
Call In
SIP: Invite
Fetch VXML App
Prepare Required Resources
MRCP & RTP: Setup, including the provisioning of speech recognition grammars
SIP: Response
MRCP: Setup
SIP: Response
RTP: Setup
SIP: OK
Session Established
SIP: OK
Start VXML App Interpretation
Ask Question
Do Recognition
RTP: Question's Audio Signal
Sphinx - Speech Recognition
Recognition Result
Sends Question to QA System
Answer for Synthesis
Mary/Festival Speech Synthesis
Synthesis Done
RTP: Answer's Audio Signal
Answer is Sent Back
SIP: Done
Sessions, Channels & Resources Cleanup
Sessions Cleanup
SIP: Done
End-of-Call Signal

Fig. 2 Flow diagram of an example HALEF call flow for the question answering application.

GET or POST request and output a VXML page that the voice browser can process next.

In order to understand how HALEF works in a better manner, let us consider an example. Figure 2 illustrates the step-by-step flow of operations that are executed in the case of a question answering (QA) back-end application. Once the Asterisk server receives a call, it sends a notification to the voice browser to fetch the VXML code from the web server. The voice browser in turn identifies the resources that the speech server will need to prepare for this application. It then notifies the MRCP server and starts sessions and channels for all required resources including the provisioning of speech recognition grammars. Finally, the speech server sends a SIP response back to the voice browser and Asterisk to confirm session initiation. Completion of this process successfully establishes a communication channel between the user and Halef's components.

Now that the session is established, Asterisk streams audio via RTP to the speech server. When the caller starts speaking, the Sphinx engine's voice activity detector fires and identifies voiced portions of the speech, and starts decoding these portions. When the voice activity detector finds that the caller has finished speaking, Sphinx sends the recognition result back to the voice browser, which passes it on to the standalone QA web application (which is served on another server) via HTTP and waits for an answer. It then sends this answer to the dialog manager which evaluates and generates VXML code with the final response to be spoken out by the speech synthesizer (either Festival or Mary). The voice browser then interprets this VXML code and sends a synthesis request to the speech server with the response. Festival/Mary synthesizes the response and passes the result back via RTP to Asterisk, which forwards the audio signal to the user. At the same time, Cairo sends a confirmation signal to the voice browser. After receiving this signal, the voice browser sends a cleanup request to close all open channels and resources. This ends the SIP session with Asterisk, which finally triggers Asterisk to send an end-of-call signal to the user.

Note that HALEF makes no assumptions on the specifics of the dialog management system used. One could choose to use a specific rule-based call flow management routine (in which case one would have to generate VXML pages corresponding to actions for each rule branch of the routine) or a more statistical system, such as one based on Partially Observable Markov Decision Processes (which one could implement as a separate web service that returns an appropriate VXML page detailing the next action to be taken by the SDS). There is similar flexibility in designing aspects of the spoken language understanding and language models for speech recognition (or grammars). In case of the latter, one could imagine wanting to use different grammars depending on the language or the domain in question. Currently HALEF supports the use of either JSGF (Java Speech Grammar Format) and ARPA (Advanced Research Projects Agency) formats to specify grammars. This modularity in design is intended to allow users more flexibility and ease of use in adapting HALEF to different use cases and environments.

3 Specific back-end use case examples

3.1 Case study I: a question answering application

The flow diagram in Figure 2 depicts the sequence of operations executed in the case of a back-end interface that allows HALEF to interact with a question answering (QA) web application called OpenEphyra [31], which was developed by researchers working on the IBM Watson DeepQA initiative [6]. We shall only briefly mention the key features here – for further details please see [14]. The application is a combination of several components including question analysis, query generation, pattern matching, answer extraction and answer selection. As this system has already been elucidated in the publications cited above, we only provide a brief description of the steps involved in answering a question here. First, the spoken input question is normalized for punctuation, abbreviations, etc. and then stemmed for nouns and verbs. Next, keywords, question type and named entities are extracted to form queries that are subsequently used to search the available knowledge base. After matching possible candidates in the database, an n-best list is returned, following which the answer with the highest confidence is chosen as the output.

3.2 Case study II: alcoholic state classification

In this section we present an example of a plug-and-play alcoholic state classification module that can be used with HALEF. The problem of alcoholic state classification has recently gained popularity in the pattern recognition community, leading to the proposal of competitions at academic conferences such as the Interspeech 2011 Speaker State Challenge [23]. That being said, recall that the main goal of the paper is *not* to present state-of-the-art classification results, but to present a working classification module (which can be optimized for performance independent of the HALEF system).

Similar to the previously described case of the question answering application, we served the alcohol language classifier as a *standalone* web service – or more specifically, a Java servlet that is served by Apache Tomcat. The speech server ships the incoming audio file to this web service, which then performs three operations. First, it preprocesses the incoming audio file and extracts features using OpenSMILE. Then it uses Weka to perform the classification using a previously-trained model. Finally, it extracts the result and generates a corresponding VXML page that contains information to be processed by the voice browser regarding how it should proceed further.

3.2.1 Data

We used the Alcohol Language Corpus (ALC) collected at the Ludwig Maximilians University of Munich to train the classifier. The dataset contains audio recordings of people in sober and alcohol-intoxicated state [19, 18], comprising 39 hours of speech from 77 female and 85 male speakers. Out of this, we performed experiments on a reduced data set[3] that was introduced by the Interspeech 2011 Speaker State challenge [23]. We further converted all audio instances of the ALC from 44.1 kHz to 8 kHz sample rate, to ensure compatibility with HALEF.

Classification tasks that leverage speech collected using a spoken dialog system are bound to certain constraints. For example, speaker turns cannot be arbitrarily long in duration in a practical setting. This is even more so when one is testing for alcohol intoxication. Therefore we only considered experimental trials during which speakers spoke prompts that were short in duration. We chose five speech prompts from the ALC that met these requirements – see Table 1 for a list of these prompts.

[3] Since the data collected during different ALC experiments are not balanced in terms of class and gender, we removed all speakers that were recorded in only one of the classication states. We then discarded as many male speakers (selected at random) as necessary to achieve gender balance.

Table 1 List of speech prompts used from the Alcohol Language Corpus [19, 18] and their corresponding test classification performance (represented as unweighted average recall, UAR).

Exp.	Command	# Samples	Test UAR
1	Sportplatzweg 27, Marktgraitz	228	68%
2	Temperatur 23°C	268	78%
3	Nächster Titel	268	73%
4	Frequenz 92.2 MHz	268	60%
5	Autobahn meiden	268	63%

3.2.2 Classification paradigm

In order to classify as sober or alcohol intoxicated, the test person dials into the HALEF system and is prompted to repeat one of the prompts, for example: "*Temperatur 23°C*" . As mentioned earlier, after the user input has been recorded, a web service is triggered to run openSMILE [5] to generate a sequence of feature vectors. The acoustic feature set used corresponds to the configuration of the Interspeech 2011 Speaker State Challenge – 4368 features comprising a multitude of low-level descriptors (such as spectral features, F0, etc.) and their applied functionals; see Schuller *et al.* (2011) [23] for more details.
We used support vector machine (SVM) classifiers to perform the classification. We ran all experiments with the Weka machine learning toolkit (version 3.7) [10, 9] in combination with LibSVM, an open-source implementation of support vector machines [4]. For evaluation we selected 10 male/female speaker pairs as test set. We tuned the complexity parameter of the linear kernel by using leave-one-*speaker pair*-out cross-validation on the remaining speaker pairs. Table 1 lists the unweighted average recall (UAR) for each test prompt. We observe that although the system performs consistently better than chance, there is scope for improvement. However, we deemed it to be sufficient in order to set up a working prototype spoken dialog interface for our purposes.

4 Conclusions and Outlook

We have presented the current state of the art of the HALEF system – a fully open-source, modular, telephony-based industry-standard-compliant spoken dialog system that can be interfaced with a number of potential back-end applications. We illustrated this capability with two example applications, that of alcoholic state classification and a question answering application. HALEF can be accessed online at the following URL: `http://halef.org`. One can also call into HALEF for a demo at the following US-based telephone number: `(206) 203-5276 (Ext. 2000: QA demo; 2001: ALC demo)`. Another back-end application that we are currently developing is a system for English language learning and assessment tailored to address the conversational competency of a user.

References

1. Black, A.W., Burger, S., Conkie, A., Hastie, H., Keizer, S., Lemon, O., Merigaud, N., Parent, G., Schubiner, G., Thomson, B., Williams, J., Yu, K., Young, S., Eskenazi, M.: Spoken Dialog Challenge 2010: Comparison of live and control test results. In: Proceedings of the SIGDIAL 2011 Conference, pp. 2–7. Association for Computational Linguistics (2011)
2. Bohus, D., Raux, A., Harris, T., Eskenazi, M., Rudnicky, A.: Olympus: An Open-Source Framework for Conversational Spoken Language Interface Research. In: Proc. of the HLT-NAACL. Rochester, USA (2007)
3. Bos, J., Klein, E., Lemon, O., Oka, T.: Dipper: Description and formalisation of an information-state update dialogue system architecture. In: 4th SIGdial Workshop on Discourse and Dialogue, pp. 115–124 (2003)
4. Chang, C.C., Lin, C.J.: LIBSVM: a Library for Support Vector Machines. ACM Trans. on Intelligent Systems and Technology **2**(3) (2011)
5. Eyben, F., Wöllmer, M., Schuller, B.: Opensmile: the Munich versatile and fast open-source audio feature extractor. In: Proc. of the MM. Florence, Italy (2010)
6. Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A., Lally, A., Murdock, W., Nyberg, E., Prager, J., Schlaefer, N., Welty, C.: Building Watson: An Overview of the DeepQA Project. AI Magazine **31**(3) (2010)
7. Gorin, A., Riccardi, G., Wright, J.: How May I Help You? Speech Communication **23**(1/2) (1997)
8. Graesser, A.C., Chipman, P., Haynes, B.C., Olney, A.: Autotutor: An intelligent tutoring system with mixed-initiative dialogue. Education, IEEE Transactions on **48**(4), 612–618 (2005)
9. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. ACM SIGKDD explorations newsletter **11**(1), 10–18 (2009)
10. Holmes, G., Donkin, A., Witten, I.H.: Weka: A machine learning workbench. In: Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on, pp. 357–361. IEEE (1994)
11. Jurčíček, F., Dušek, O., Plátek, O., Žilka, L.: Alex: A statistical dialogue systems framework. In: Text, Speech and Dialogue, pp. 587–594. Springer (2014)
12. Lamere, P., Kwok, P., Gouvea, E., Raj, B., Singh, R., Walker, W., Warmuth, M., Wolf, P.: The CMU SPHINX-4 Speech Recognition System. In: Proc. of the ICASSP'03. Hong Kong, China (2003)
13. van Meggelen, J., Smith, J., Madsen, L.: Asterisk: The Future of Telephony. O'Reilly, Sebastopol, USA (2009)
14. Mehrez, T., Abdelkawy, A., Heikal, Y., Lange, P., Nabil, H., Suendermann-Oeft, D.: Who Discovered the Electron Neutrino? A Telephony-Based Distributed Open-Source Standard-Compliant Spoken Dialog System for Question Answering. In: Proc. of the GSCL. Darmstadt, Germany (2013)
15. Pieraccini, R., Huerta, J.: Where Do We Go from Here? Research and Commercial Spoken Dialog Systems. In: Proc. of the SIGdial. Lisbon, Portugal (2005)
16. Prylipko, D., Schnelle-Walka, D., Lord, S., Wendemuth, A.: Zanzibar OpenIVR: An Open-Source Framework for Development of Spoken Dialog Systems. In: Proc. of the TSD. Pilsen, Czech Republic (2011)
17. Raux, A., Langner, B., Bohus, D., Black, A., Eskenazi, M.: Let's Go Public! Taking a Spoken Dialog System to the Real World. In: Proc. of the Interspeech. Lisbon, Portugal (2005)
18. Schiel, F., Heinrich, C.: Laying the Foundation for In-Car Alcohol Detection by Speech. In: Proc. of the Interspeech. Brighton, UK (2009)
19. Schiel, F., Heinrich, C., Barfüsser, S., Gilg, T.: ALC—Alcohol Language Corpus. In: Proc. of the LREC. Marrakesh, Morocco (2008)
20. Schmitt, A., Scholz, M., Minker, W., Liscombe, J., Suendermann, D.: Is It Possible to Predict Task Completion in Automated Troubleshooters? In: Proc. of the Interspeech. Makuhari, Japan (2010)
21. Schnelle-Walka, D., Radomski, S., Mühlhäuser, M.: JVoiceXML as a Modality Component in the W3C Multimodal Architecture. Journal on Multimodal User Interfaces (2013)

22. Schröder, M., Trouvain, J.: The german text-to-speech synthesis system mary: A tool for research, development and teaching. International Journal of Speech Technology **6**(4), 365–377 (2003)
23. Schuller, B., Steidl, S., Batliner, A., Schiel, F., Krajewski, J.: The interspeech 2011 speaker state challenge. In: INTERSPEECH, pp. 3201–3204 (2011)
24. Seneff, S., Wang, C., Zhang, J.: Spoken conversational interaction for language learning. In: InSTIL/ICALL Symposium 2004 (2004)
25. Suendermann, D.: Advances in Commercial Deployment of Spoken Dialog Systems. Springer, New York, USA (2011)
26. Suendermann-Oeft, D.: Modern conversational agents. In: Technologien für digitale Innovationen, pp. 63–84. Springer (2014)
27. Taylor, P., Black, A., Caley, R.: The Architecture of the Festival Speech Synthesis System. In: Proc. of the ESCA Workshop on Speech Synthesis. Jenolan Caves, Australia (1998)
28. Williams, J.D., Young, S.: Partially observable markov decision processes for spoken dialog systems. Computer Speech & Language **21**(2), 393–422 (2007)
29. Xu, Y., Seneff, S.: A generic framework for building dialogue games for language learning: application in the flight domain. In: SLaTE, pp. 73–76. Citeseer (2011)
30. Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., Yu, K.: The hidden information state model: A practical framework for pomdp-based spoken dialogue management. Computer Speech & Language **24**(2), 150–174 (2010)
31. van Zaanen, M.: Multi-lingual question answering using OpenEphyra. In: Working Notes for the Cross Language Evaluation Forum (CLEF), pp. 1–6 (2008)