Speech Control for HTML5 Hypervideo Players

Britta Meixner[1,2], **Fabian Kallmeier**[1]
[1]University of Passau, Innstrasse 43, 94032 Passau, Germany
[2]FX Palo Alto Laboratory, 3174 Porter Drive, Palo Alto, CA 94304, USA
meixner@fxpal.com, kallmeie@fim.uni-passau.de

ABSTRACT
Hypervideo usage scenarios like physiotherapy trainings or instructions for manual tasks make it hard for users to use an input device like a mouse or touch screen on a hand-held device while they are performing an exercise or use both hands to perform a manual task. In this work, we are trying to overcome this issue by providing an alternative input method for hypervideo navigation using speech commands. In a user test, we evaluated two different speech recognition libraries, annyang (in combination with the Web Speech API) and PocketSphinx.js (in combination with the Web Audio API), for their usability to control hypervideo players. Test users spoke 18 words, either in German or English, which were recorded and then processed by both libraries. We found out that annyang shows better recognition results. However, depending on other factors of influence, like the occurrence of background noise (reliability), the availability of an internet connection, or the used browser, PocketSphinx.js may be a better fit.

ACM Classification Keywords
H.5.2. Information Interfaces and Presentation (e.g. HCI): User Interfaces

Author Keywords
Hypervideo; Navigation; Language Processing; Speech Input; HTML5;

INTRODUCTION

Using speech input, users are nowadays able to control smartphones, navigation systems, and Smart-TVs without touching them. Depending on the system, either certain commands are recognized (for example in TomTom navigation systems [20]), or freely formulated questions can be asked (like Siri for iPhones [1], Utter for Android Phones [10], or the Google app [9]) which are then processed by the system trying to find an answer.

However, up to now, only few homepages and Web applications have built-in support for speech input. Especially hypervideo players could benefit from speech control. Hypervideos consist of interlinked video scenes which are enriched with additional information. Playing such videos requires special players that provide additional means of navigation in additional information, in scenes, and between scenes [14]. In usage scenarios like cooking instructions, physiotherapy and fitness trainings [15], or physical tasks that have to be done with two hands, speech controls may help the user to navigate in the hypervideo without having to interrupt the current task. The hypervideo may be paused, next scenes may be selected, or annotations may be read without having to interrupt the task/exercise using voice commands.

4th International Workshop on Interactive Content Consumption at TVX'16, June 22, 2016, Chicago, IL, USA

SPEECH RECOGNITION FRAMEWORKS

Several speech recognition APIs exist having varying features and limitations. Available APIs are, for example, Google Speech API [6] which accepts 10-15 seconds of audio, the IBM Speech to Text API [11] which uses IBM's speech recognition capabilities, wit.ai [26] which is an open and extensible natural language platform, Speechmatics [19] and the VoxSigma REST API [21] which transcribe uploaded files into text, or the open source APIs Kaldi [12] and OpenEars [18], the latter of which provides free speech recognition and speech synthesis for the iPhone. Hereafter we briefly describe the combinations of frameworks that will be tested in the remainder of this work. We chose these frameworks based on the following criteria: the framework should be able to process longer phrases (in case the speech recognition gets extended in the player). It should be possible to integrate it into a Web application and the library should not be limited to certain OSes.

Web Audio API and PocketSphinx.js

The Web Audio API is a "high-level JavaScript API for processing and synthesizing audio in web applications" [24]. It allows splitting and merging of channels in an audio stream. Audio sources from an HTML5 `<audio>` or `<video>` element can be processed. It is furthermore possible to process live audio input from a MediaStream via `getUserMedia()` [24]. The speech recognition library PocketSphinx.js is written entirely in JavaScript. It is running entirely in the web browser [7], which builds on the Web Audio API. The speech recognizer is implemented in C (PocketSphinx) [3] and converted into JavaScript using Emscripten [5]. It is possible to add words, grammar, and key phrases to extend or improve the recognition [7]. Each language needs its own language model with a vocabulary.

Web Speech API and annyang

The Web Speech API enables the incorporation of voice data into web apps [17][23]. The SpeechRecognition (Asyn-

chronous Speech Recognition) interface "provides the ability to recognize voice context from an audio input (normally via the device's default speech recognition service) and respond appropriately" [17]. The SpeechGrammar interface represents a container for a particular set of grammar (defined in the JSpeech Grammar Format (JSGF)) that an app should recognize [17]. As most modern OSes have a speech recognition system for issuing voice commands, this is used for speech recognition on the device. Speech recognition systems are, for example, Dictation on Mac OS X [2], Siri on iOS [1], Cortana on Windows 10 [16], and Android Speech [8]. The tiny standalone JavaScript SpeechRecognition library annyang lets users control a homepage with voice commands [4]. The size of the library is less than 1 kb. The back-end is supported by the (Chrome) Web Speech API [25].

IMPLEMENTATION

In order to test the annyang and the PocketSphinx.js projects, we installed a reference platform. It consists of a Web server (Apache 2.4.10) and a database (MySQL 5.5.38). We used Perl (Version 5.14.2) for the implementation of the dynamic test Web page, which shows contents in the selected language (German or English) and provides a log-in system to avoid abuse and falsification of the test results. We only used Google Chrome for our tests, because annyang is built on the Web Speech API which was only available for Google Chrome at the time of the tests.

annyang

For an implementation of speech detection and recognition with annyang, it was only necessary to include the JavaScript library into the Web application. In order to allow the usage of the microphone it was mandatory to install an SSL-certificate. We furthermore rewrote the `onResult` function of the annyang project to make the implementation conform with the PocketSphinx.js implementation described hereafter. To test German words, the language only had to be set to German using the `setLanguage` function. Further modifications and additions were not necessary.

PocketSphinx.js

The implementation of speech detection and recognition with PocketSphinx.js required more effort compared to the implementation with annyang, because the source code of PocketSphinx.js only comes with an English acoustical model. To avoid having to generate our own acoustical model for German words, we used the one provided by VoxForge [22]. We furthermore used the possibility to add words and grammars during run-time to avoid too large files which could lead to crashes of the browser. For that reason we compiled the acoustical models outside the main file which led to smaller files and a better performance.

Test System

The web page used for the tests consisted of a `index.pl` file in Perl (which generated the HTML code), some JavaScript files, and a MySQL database. The database stored user names and passwords, test words and their pronunciation, test results, as well as data that might be displayed on the dynamic web page. JavaScript was used to start and stop voice recording and recognition in the different technologies. In our tests, we used a fixed set of words which was shown in our test application and had to be spoken out loud by the participants. We furthermore used a timer which limited the recognition time per word and showed each word for 10 seconds. If a word was recognized correctly during the first attempt, the next word was loaded. If it was not recognized, a second attempt with a new timer was started. The user then had to repeat the word. The Web page informed the user whether the word was recognized correctly. The system used versions of annyang and Pocketsphinx.js available in November 2014.

STUDY/METHOD

To find out if annyang or PocketSphinx.js performs better, we conducted a study with 58 participants.

Procedure/Data Collection

We used the 18 words shown in Table 1 which represent the key functions of our HTML5 hypervideo player. The words were presented to the participants in random order to avoid exercise effects towards higher word IDs. Each recorded word was tested with both technologies, annyang and PocketSphinx.js. Before starting the tests, the users had to select which language they wanted to do the test in. As a result, 33 participants used the German version of the test and 25 users participated in the English version.

Table 1. Words tested for recognition.

ID	German word	English word
1	abspielen	play
2	wiederholen	repeat
3	öffnen	open
4	schließen	close
5	lauter	volume up
6	leiser	volume down
7	einblenden	fade in
8	ausblenden	fade out
9	vorwärts	previous
10	zurück	next
11	Inhaltsverzeichnis	content
12	Suche	search
13	Tagebuch	journal
14	Vollbild	full screen
15	Fensteransicht	windows view
16	Bild	picture
17	Bildergalerie	picture gallery
18	Hauptvideo	main video

Participants

The participants in our study were mainly between 18 and 60 years old. 34 of the participants were male, 24 were female. The test was mainly distributed in Germany, so most of the participants were native German speakers. The tests were conducted on desktop computers or laptops, whereby 33 participants used internal and 25 participants used external microphones. See Table 2 for more precise demographic data.

Table 2. Test participant demographics.

		part.
Age	below 18	1
	18-29	32
	30-45	11
	46-60	14
	above 60	0
Gender	male	34
	female	24
First language	German	55
	English	0
	other	3
Microphone	internal	33
	external	25
Experience with speech input	none	20
	some	15
	medium	20
	often	3
	daily	0

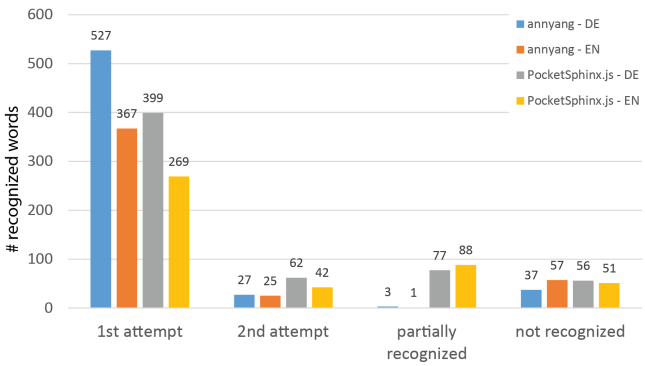

Figure 1. Recognition grouped by attempts.

ANALYSIS AND RESULTS

We analyzed the frameworks in two different ways. On the one hand, we analyzed the number of recognized words per language and per framework. On the other hand, we compared the two frameworks in different categories relevant for practical usage in our hypervideo player.

Recognition of Words

We analyze the recognition of the words for the two languages first separately and then together. Taking a look at the recognition of the German words, it can be said that annyang has a better recognition rate than PocketSphinx.js (see Table 3 and Figure 1, blue and gray bars). Out of 594 words (18 words spoken by 33 test users), annyang recognized 527 words in the first and 27 in the second attempt which results in 554 recognized words. PocketSphinx.js in contrast recognized 399 words in the first and 62 in the second attempt which results in 461 recognized words. The annyang library failed to recognize 37 words, while the number of not recognized words for PocketSphinx.js was 56. The biggest difference was in the number of partially recognized words[1], where the number for annyang was quite low, but PocketSphinx.js recognized 77 words partially.

Table 3. Recognition of German words.

	annyang	**PocketSphinx.js**
1st attempt	527	399
2nd attempt	27	62
Partially recognized	3	77
Not recognized	37	56

Taking a look at the results for the English words (see Table 4 and Figure 1, orange and yellow bars), the results are similar to those of the German words. Out of 450 words (18 words spoken by 25 test users), annyang recognized 367 in the first and 25 in the second attempt resulting in 392 correctly recognized words. In contrast, PocketSphinx.js recognized 269 words in the first and 42 words in the second attempt resulting in 311 correctly recognized words. Only 1 word was recognized partially using annyang, whereas PocketSphinx.js recognized 88 words partially. For the English words, annyang showed slightly worse results (57 not recognized words) than PocketSphinx.js (51 not recognized words). One reason for the higher number of not recognized words might be the fact that the words were not spoken by native speakers. The level of correct pronunciation is unfortunately not known in this case.

Table 4. Recognition of English words.

	annyang	**PocketSphinx.js**
1st attempt	367	269
2nd attempt	25	42
Partially recognized	1	88
Not recognized	57	51

Summarizing the results over all languages, it can be noted that annyang showed better overall results than PocketSphinx.js (see Table 5). Annyang had a recognition rate of 90.61 % while PocketSphinx.js recognized only about three quarters (73.94 %) of the words. The rate of not recognized words was around 10 % for both libraries. One reason for the worse results for PocketSphinx.js may be background noise which has a greater influence on PocketSphinx.js than on annyang.

Table 5. Recognition rate of all words in percent.

	annyang	**PocketSphinx.js**
1st attempt	85.63	63.98
2nd attempt	4.98	9.96
Partially recognized	0.38	15.80
Not recognized	9.00	10.25

Taking a look at the recognition performance of individual words (see Figure 2), it can be stated that the recognition for

[1]Partially recognized words are words that either are only a part of the given word or contain the given word (but also other letters), meaning the recognized word contains more or less letters than the given word.

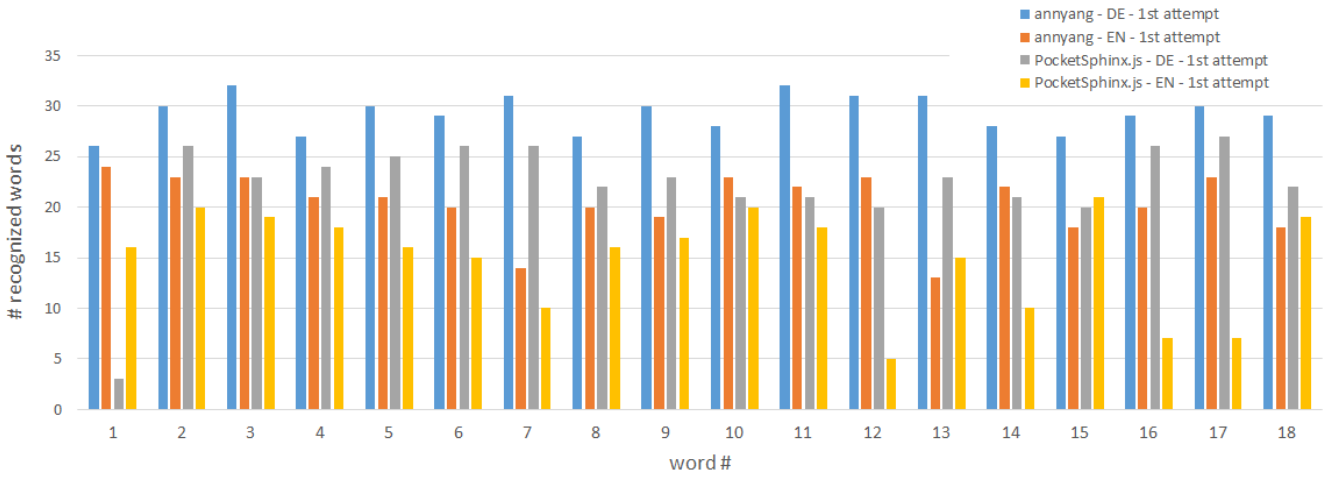

Figure 2. Recognition grouped by words.

the German words for annyang did not show huge differences between the words. The English words in contrast showed larger differences. The words "fade in" and "journal" were recognized correctly less than 15 out of 25 times compared to annyang. The results for German words with PocketSphinx.js are worse for all words, especially for the word "abspielen". Taking a look at the results for the English words, it can be said that especially the words "fade in", "search", "full screen", "picture", and "picture gallery" showed worse results with 10 or less out of 25 recognized words.

Practical Comparison

While the results in the user tests regarding word recognition performance clearly were in favor of annyang, the decision for using one of the libraries in real world HTML5 hypervideo players requires further thoughts. We examined 5 factors further, namely: dependencies and integration, reliability, availability, browser support, and supported languages. Dependencies and integration as well as supported languages may be of less interest. Assuming that no large changes are made in the Web application that uses the speech recognition, the integration only has to be implemented once. Regarding language support, both libraries show a large number of supported languages or provide the possibility to extend or create language models in case they do not exist aready.

Reliability, availability, and browser support play a more important role. Depending on the hypervideo application area, background noise may occur, an internet connection may not be available at all times, or company restrictions may not allow to use certain browsers. Please refer to Table 6 for a comparison what to use best in a given scenario.

Table 6. Practical comparison of annyang and PocketSphinx.js

	annyang	**PocketSphinx.js**
Reliability	Good: Background noise is reliably distinguished from language; recognition of spoken words is reliable in most cases	Satisfactory: recognition is reliable as long as the surroundings have no background noises
Availability	Internet connection is necessary	Application on client side, no Internet connection necessary
Browser support	Limited to Chrome	All current browsers except Internet Explorer

CONCLUSION

In this work, we describe the implementation of a test framework for the speech recognition libraries annyang and PocketSphinx.js. We wanted to test the quality of the recognition of certain words that could be used to verbally control hypervideo players. As a result, it can be noted that annyang provides better recognition results both for English and German words.

However, recognition may not be the only factor to consider when integrating one of the libraries into a hypervideo player. Depending on the application area, the occurrence of background noise (reliability), the availability of an internet connection, and the used browsers may influence the selection of the library.

In the tests described in this work, we only used Google Chrome, due to a missing support of the libraries in other browsers. In future work a test with other browsers and the test of other libraries may bring further results that may influence the selection of one of the libraries.

The voice control should be integrated into the hypervideo player and tested in a real world scenario measuring user frustration due to speech recognition performance in a real world setting. Depending on the scenario the hypervideo player is used in, another hypervideo control approach may also be helpful. In case of a physiotherapy or fitness training, for example, it is helpful to show main video contents on a bigger screen. A solution to enable an easier control of the hypervideo in this specific case may be a second screen

application that splits contents from control elements [13]. Both approaches should be compared for their suitability in these scenarios.

REFERENCES

1. Apple Inc. 2016a. Use Siri on your iPhone, iPad, or iPod touch. (2016). Website https://support.apple.com/en-us/HT204389 (accessed April 20, 2016).
2. Apple Inc. 2016b. Use your voice to enter text on your Mac. (2016). Website https://support.apple.com/en-us/HT202584 (accessed May 27, 2016).
3. Carnegie Mellon University. 2016. CMU Sphinx - OPEN SOURCE SPEECH RECOGNITION TOOLKIT. (2016). Website http://cmusphinx.sourceforge.net/ (accessed April 20, 2016).
4. GitHub, Inc. 2016a. annyang - Speech recognition for your site. (2016). Website https://github.com/TalAter/annyang (accessed April 20, 2016).
5. GitHub, Inc. 2016b. Emscripten: An LLVM-to-JavaScript Compiler. (2016). Website https://github.com/kripken/emscripten (accessed April 20, 2016).
6. GitHub, Inc. 2016c. Google Speech API v2. (2016). Website https://github.com/gillesdemey/google-speech-v2 (accessed May 27, 2016).
7. GitHub, Inc. 2016d. Pocketsphinx.js - Speech Recognition in JavaScript. (2016). Website https://github.com/syl22-00/pocketsphinx.js/blob/master/README.md (accessed April 20, 2016).
8. Google. 2016a. android.speech. (2016). Website https://developer.android.com/reference/android/speech/package-summary.html (accessed April 20, 2016).
9. Google. 2016b. Meet the Google app. (2016). Website http://www.google.com/search/about/ (accessed April 20, 2016).
10. Google. 2016c. utter! Voice Commands BETA! (2016). Website https://play.google.com/store/apps/details?id=com.brandall.nutter (accessed April 20, 2016).
11. IBM. 2016. Speech to Text. (2016). Website http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/speech-to-text.html$#$how-it-is-used-block (accessed May 27, 2016).
12. Kaldi. 2016. Kaldi. (2016). Website http://kaldi-asr.org/ (accessed May 27, 2016).
13. Britta Meixner, Christian Handschigl, Stefan John, and Michael Granitzer. 2016. From Single Screen to Dual Screen - a Design Study for a User-Controlled Hypervideo-Based Physiotherapy Training. In *Proceedings of WSICC 2016*.
14. Britta Meixner, Stefan John, and Christian Handschigl. 2015. SIVA Suite: Framework for Hypervideo Creation, Playback and Management. In *Proceedings of the 23rd ACM International Conference on Multimedia (MM '15)*. ACM, New York, NY, USA, 713–716. DOI: http://dx.doi.org/10.1145/2733373.2807413
15. Britta Meixner, Katrin Tonndorf, Stefan John, Christian Handschigl, Kai Hofmann, and Michael Granitzer. 2014. A Multimedia Help System for a Medical Scenario in a Rehabilitation Clinic. In *Proceedings of the 14th International Conference on Knowledge Technologies and Data-driven Business (i-KNOW '14)*. ACM, New York, NY, USA, Article 25, 8 pages. DOI: http://dx.doi.org/10.1145/2637748.2638429
16. Microsoft. 2016. Get Started with Windows 10 - What is Cortana? (2016). Website http://windows.microsoft.com/en-us/windows-10/getstarted-what-is-cortana (accessed April 20, 2016).
17. Mozilla Developer Network. 2016. Web Speech API. (2016). Website https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API (accessed April 20, 2016).
18. Politepix. 2016. OpenEars - iPhone Voice Recognition and Text-To-Speech. (2016). Website http://www.politepix.com/openears/ (accessed May 27, 2016).
19. Speechmatics. 2016. speech made simple. (2016). Website https://speechmatics.com/ (accessed May 27, 2016).
20. TomTom International BV. 2016. Why TomTom devices are the easiest. (2016). Website http://www.tomtom.com/whytomtom/subject.php?subject=4 (accessed April 20, 2016).
21. Vocapia Research SAS. 2016. Speech to Text API. (2016). Website http://www.vocapia.com/speech-to-text-api.html (accessed May 27, 2016).
22. VoxForge. 2016. VoxForge - Downloads - German. (2016). Website http://www.voxforge.org/de/Downloads (accessed April 20, 2016).
23. W3C. 2012. Web Speech API Specification (19 October 2012). (2012). Website https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html (accessed June 09, 2016).
24. W3C. 2016. Web Audio API - W3C Editor's Draft 15 April 2016. (2016). Website https://webaudio.github.io/web-audio-api/ (accessed April 19, 2016).
25. WEBRESOURCESDEPOT. 2016. Speech Recognition With JavaScript - annyang. (2016). Website http://webresourcesdepot.com/speech-recognition-with-javascript-annyang/ (accessed April 20, 2016).
26. Wit.ai, Inc. 2016. wit.ai - Natural Language for Developers. (2016). Website https://wit.ai/ (accessed May 27, 2016).