A Benchmarking of IBM, Google and Wit Automatic Speech Recognition Systems

Foteini Filippidou(✉) and Lefteris Moussiades

Department of Computer Science, International Hellenic University,
Agios Loukas, 65404 Kavala, Greece
{fnfilip,lmous}@cs.ihu.gr

Abstract. As the requirements for automatic speech recognition are continually increasing, the demand for accuracy and efficiency is also of particular interest. In this paper, we present most of the well-known Automated Speech Recognition systems (ASR), and we benchmark three of them, namely the IBM Watson, Google, and Wit, using the WER, Hper, and Rper error metrics. The experimental results show that Google's automatic speech recognition performs better among the three systems. We intend to extend the benchmarking both to include most of the available Automated Speech Recognition systems and increase our test data.

Keywords: Automatic speech recognition · WER · Hper · Rper

1 Introduction

Speech is probably the primary means of man communication. Therefore, acquiring speech from computers is reasonable to contribute to more effective man-machine communication. Two primary technologies have developed concerning speech: the Automatic Speech Recognition or ASR, for short, and the Text to Speech or TTS. ASR refers to the conversion of speech into text [1], while TTS, as its name suggests, is the reverse process [2]. Although man naturally acquires speech in early growth [3], speech production and recognition by computers is a complicated process that has extensively been addressed by the research community.

As early as 1952, the first system was built that could identify digits with high precision. This early system required users to pause after each digit [4]. Raj Reddy constructed the first recognition system of continuous speech as a student at Stanford University in the late 1960s [Wikipedia - Speech Recognition].

Shortly after, T.K. Vintsyuk presented an algorithm [7] that can recognize speech, creating a sequence of words that contained in continuous and connected speech.

Later, in 1981, Logica developed a real-time speech recognition system based on the original project of the Joint Speech Research Unit in the United Kingdom. This user-friendly system is one of the first steps in improving human-machine communication [5]. Today, ASR applications are not just confined to human-machine communication for personal use but include industrial machine guidance with voice commands [6, 7], automatic

Published by Springer Nature Switzerland AG 2020
I. Maglogiannis et al. (Eds.): AIAI 2020, IFIP AICT 583, pp. 73–82, 2020.
https://doi.org/10.1007/978-3-030-49161-1_7

telephone communication [8], communication with automotive systems, military vehicles, and other equipment, communication with health care, aerospace and other systems [9]. Also, ASR systems are utilized to provide more specialized services, such as training in pronunciation and vocabulary [10].

As the number of ASR systems is continuously increasing, it is quite challenging to select the most appropriate for a particular application. Our immediate plans are to build an artificial vocabulary learning assistant. In particular, this assistant will be able to contribute to the learning of vocabulary by entering into dialogues with the trainee. In this context, it is necessary to select the appropriate ASR. Thus, this work first attempts a presentation of known ASR systems and then proceeds to benchmark three well-known systems, namely IBM Watson, Google, and Wit. Therefore, we set the foundations for a more general assessment of most ASR systems with the ultimate goal of choosing the most appropriate vocabulary learning assistant.

The rest of this review is organized as follows. In section two, we briefly present analyze the Architecture of ASR systems. In section three, some of the most well-known ASR systems are introduced. In section four, we describe criteria and metrics used for system evaluation. Next, in section five, we analyze the data and the methodology we use for the experimental procedure. Finally, section six closes the review with final remarks and conclusions.

2 Architecture

An ASR application accepts the speech signal as input and converts it into a series of words in text form. During the speech recognition process, a list of possible texts is created, and finally, the most relevant text to the original sound signal is selected [4, 5]. A typical ASR consists of an acoustic front-end that process the speech signal to extract useful features [11]. Then, a feature vector is generated. Several feature extraction methods are used including, Principle Component Analysis (PCA), Linear Discriminant Analysis (LDA), Independent Component Analysis (ICA), Linear Predictive Coding (LPC), Cepstral Analysis, Mel-Frequency Scale Analysis, Filter-Bank Analysis, Mel-Frequency Cepstrum Coefficients (MFCC), Kernal Based Feature Extraction, Dynamic Feature Extraction, Wavelet-based features, Spectral Subtraction and Cepstral Mean Subtraction (CMS). Generally, some spectra temporal analysis of the signal generates features that usually transform into more compact and robust vectors [11]. At the processing step, the acoustic lexicon and language model are used by a decoder (search algorithm) to produce the hypothesized word or phoneme, as shown in Fig. 1. The acoustic model contains acoustic features for each of the distinct phonetic units and typically refers to the process of establishing the statistical representations for the feature vector sequences computed from the speech waveform. The Hidden Markov Model (HMM) is one of the most commonly used to build acoustic models. Other acoustic models include segmental and super segmental models, neural networks, maximum entropy models, and conditional random fields. An acoustic model is a file that contains statistical representations of each of the distinct sounds that makes up a word [11]. The lexicon includes terms of the vocabulary of the current application [11]. The language model consists of the limitations associated with the sequence of words that are accepted in a given language.

Two popular tools for language modelling are the CMU Statistical Language Modeling (SLM) Toolkit and the Stanford Research Institute Language Modeling Toolkit [11]. The decoder aims to find the most likely sequences of words that could match the audio signal by applying appropriate models. Most decoding algorithms produce a list of possible word sequences called the n-best list [4, 5].

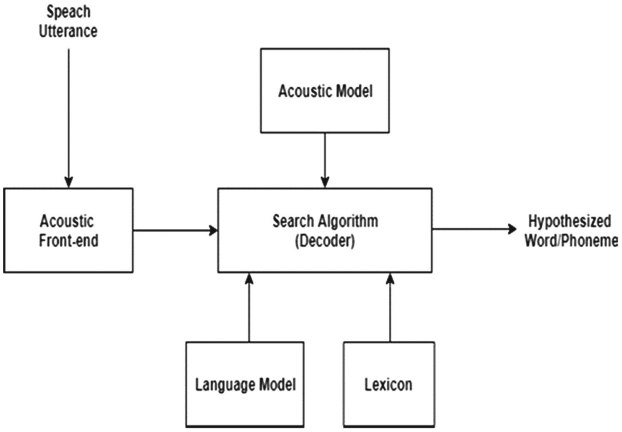

Fig. 1. Speech recognition architecture [11]

3 Well-Known ASR Systems

Many international organizations, such as Microsoft SAPI, Dragon-Natural-Speech, and Google, as well as research teams, are working in the area of ASR systems [3]. Next, we describe in short some of the most well-known ASR systems.

CMU-Sphinx was introduced in 1986 and is a set of open-source speech recognition libraries. It supports English but not Greek. Contains a series of packages for different tasks and applications:

Pocketsphinx: Lightweight library with C for written recognition [12]. It is fast, runs locally, and requires relatively modest computational resources. It provides nbest lists and lattices and supports incremental output. It provides voice activity detection functionality for continuous ASR and is also fully customizable and trainable [13].

Sphinxbase: Support for the libraries required by Pocketsphinx [12].

CMUclmtk: Language model tools [12].

Sphinxtrain: Acoustic model tool. It includes a set of programs and documentation for making and building audio models for any language [12].

Sphinx3: Voice recognition decoder is written in C [12].

Sphinx4: The latest Java-based speech recognition tool made by Carnegie Mellon University (CMU), Sun Microsystems Laboratories, and Mitsubishi

Electric Research Laboratories (MERL). It is flexible, supporting all types of acoustic models based on the Hidden Markov model (HMM), all basic types of language models, and multiple search strategies [14].

CMU also attempted to develop an Amazigh speech recognition system used in a vast geographical area of North Africa [15].

Kaldi is also an open-source speech recognition tool, written in C++. Kaldi's goal is to have a modern and flexible code that is easy to understand, modify, and expand. Kaldi supports conventional models (GMMs) and Subspace Gaussian Mixture (SGMMs) but can be easily extended to new types of models [16].

Google uses closed-source speech recognition technology based on deep learning achieving an error rate of 8% in 2015, down by more than 23% from 2013 [17].

Dragon Naturally Speaking launched in 1975 by Dr James Baker supports ASR and TTS [18].

Microsoft Speech API (SAPI) is a closed-source library developed that supports ASR and TTS within Windows applications. SAPI 5 is the latest major version released in 2000 [2].

DragonDictate (TM)-30 K an ASR containing an extensive vocabulary that allows interactive users to create printed text faster through speech than they would have done by hand [19].

Amazon Transcribe is an ASR service used in many typical applications, including phone customer service systems, and automatic subtitling for audio and video. Amazon Transcribe is continuously learning and improving to keep up with the development of language [20].

Microsoft Azure provided as a service, converts speech to text in real-time. It includes an API Speech SDK and a REST API. The latter can be used only for queries containing up to 10 s recorded audio [21].

Wit is free even for commercial use. It supports over 130 languages, and it applies the applicable EU data protection laws, including GDPR. Wit supports Node, Python, and Ruby programming languages [22].

Twilio includes vocabulary to support the industry. It recognizes 119 languages and dialects. It is paid but also has a free trial version [23].

Houndify is a platform that allows anyone to create intelligent communication interfaces with voice capability. It supports both ASR and TTS on Android, iOS, C++, Web, Python, Java, and C Sharp. It is available for a fee but also has a free version [24].

IBM® (Speech to Text) allows users to add up to 90 thousand out-of-vocabulary (OOV) words to a custom language model. The service utilizes machine learning to combine the knowledge of grammar and language structure as well as the synthesis of sound and voice signals to transform the human voice accurately. It continually renews and improves its function as it receives speech. It can support continuous speech. Programming languages that supports are Node, Java ™, Python, Ruby, Swift, and Go SDKs [25]. The service offers three interfaces: A WebSocket and synchronous HTTP interface, each being able to transfer up to 100 MB of audio data with one request. It also offers an asynchronous HTTP interface, which with one request can pass up to 1 GB of audio data [25].

AT&T Watson, created in 1995 by AT & T Research [26]. It allows users to create real-time multilingual voice and speech recognition applications [27]. It is a cloud-based service that can be accessed through HTML POST requests.

Web Speech API, developed by the W3C Speech API team, it is open but seems to be driven by two companies: Google and Openstream. To upload the audio data, the user needs to open an HTTPS connection with the network service through a POST request [28].

NaturalReader. It can read any text, such as Microsoft Word files, Web pages, PDFs, and emails, and convert them into audio files. It is available in free and paid versions [29].

iSpeech is an online free solution available for mobile applications. It has a unique multithreaded and multicore method for text-to-speech conversion, which allows for a simple text-to-speech approach with an unlimited number of processors simultaneously. Thus, it is speedy. It also provides an SDK for java applications [30].

WaveSurfer is an open-source model made in KTH's Center for Speech Technology. An essential feature of WaveSurfer is its versatile configuration [31].

Julius is open-source written in C with high performance. It incorporates significant cutting-edge speech recognition techniques, achieving LVCSR, real-time vocabulary, and real-time speech recognition. It works on Linux, Windows, Mac OS X, Solaris, and other Unix variants and also runs on the Apple iPhone. It supports several languages [32].

4 Evaluation Metrics

There are three types of errors that occur in speech recognition [33]:

If a word in the reference sequence is transcribed as a different word, it is called substitution. When a word is completely missing in the automatic transcription, it is characterized as deletion. The appearance of a word in the transcription that has no correspondent in the reference word sequence is called insertion.

The performance accuracy of a system is usually rated by the word error rate (WER) (1), a popular ASR comparison index [34]. It expresses the distance between the word sequence that produces an ASR and the reference series.

Defined as:

$$\mathrm{WER} = (\mathrm{S} + \mathrm{D} + \mathrm{I})/\mathrm{N}_1 = (\mathrm{S} + \mathrm{D} + \mathrm{I})/(\mathrm{H} + \mathrm{S} + \mathrm{D}) \quad (1)$$

where I = the total number of entries, D = total number of deletions, S = total number of replacements, H = total number of successes, and N_1 = total number of reference words [33].

Despite being the most commonly used, WER has some cons. It is not an actual percentage because it has no upper bound. When S = D = 0 and we have two insertions for each input word, then I = N1 (namely when the length of the results is higher than the number of words in the prompt), which means WER = 200%. Therefore it does not tell how good a system is, but only that one is better than another. Moreover, in noisy conditions, WER could exceed 100%, as it gives far more weight to insertions than to deletions [33].

Match Error Rate (MER) is the proportion of I/O word matches, which are errors, which means that is the probability of a given match being incorrect. The ranking behaviour of MER (2) is between that of WER and WIL [35].

$$\text{MER} = (\text{S} + \text{D} + \text{I})/(\text{N} = \text{H} + \text{S} + \text{D} + \text{I}) = 1 - \text{H}/\text{N} \tag{2}$$

WIL is a simple approximation to the proportion of word information lost, which overcomes the problems associated with the RIL (relevant information lost) measure that was proposed years ago [35].

The Relative Information Lost (RIL) measure was not widely taken up because it is not as simple as WER, and it measures zero error for any one-one mapping between input and output words, which is unacceptable [35].

The evaluation of an ASR system also correlates with PER (Position-Independent word Error Rate) automatic metric. PER compares the words in the two sentences (hypothesis and reference) and is always lower than or equal to WER. Hypothesis per (Hper) (similar to precision) (3) refers to the set of words in a hypothesis sentence which do not appear in the reference sentence. Reference PER (Rper) (similar to recall) (4), denotes the set of words in a reference sentence which do not appear in the hypothesis sentence. In other words, the main goal of Hper and Rper is to identify all words in the hypothesis which do not have a counterpart in the reference and vice versa [36].

$$\text{WER} = \frac{\text{S} + \text{D} + \text{I}}{\text{N}_1} = \frac{\text{S} + \text{D} + \text{I}}{\text{H} + \text{S} + \text{D}} \tag{3}$$

$$\text{MER} = \frac{\text{S} + \text{D} + \text{I}}{\text{N} = \text{H} + \text{S} + \text{D} + \text{I}} = 1 - \frac{\text{H}}{\text{N}} \tag{4}$$

5 Experimentation

5.1 Data and Methodology

From the tools described above, IBM Watson, Wit, and Google were selected to be tested and compared to the accuracy of the results they produce.

Three speakers recorded specific sentences in English. The recorded sentences are typical in dialogues between a teacher and a student while learning English as a second language. The used sentences contain words, numbers, and names.

Speakers are not native English speakers, and their native language is Greek. Speakers A and B are women, and speaker C is a man. We intend to evaluate ASR systems for an application that will train the student in vocabulary learning. Recording took place in an ideal environment with no noise except for speaker C, whose recording has a low level of noise. Each of the three speakers recorded twenty sentences.

Experimental measurements were made using audio files, recorded with Audacity 2.3.1. The WAV files were stereo with two recording channels, 44100 Hz, 32 bit, and they range in size from about 350 KB to 0.99 MB.

Generated texts were compared to the corresponding correct texts that had been given to the speakers. The Word error rate (WER), Hper, and Rper were measured using an Open Source Tool for Automatic Error Classification of Machine Translation Output available on GitHub [37].

6 Results

The text files were created with Uberi, available at https://github.com/Uberi/speech_recognition. Uberi uses the Python Speech Recognition library and supports several ASR engines.

Comparing the resulting texts with the reference texts, we calculated the WER error measurements as a percentage with an accuracy of two decimal places. WER measurements were made using the tools that are available on pages https://github.com/belambert/asr-evaluation and https://github.com/cidermole/hjerson. In the end, the average error of each tool per speaker was calculated and is presented in Table 1.

Table 1. Average WER (%) measurements

Speaker A			Speaker B			Speaker C		
IBM	Google	Wit	IBM	Google	Wit	IBM	Google	Wit
30.10	16.60	25.87	47.73	20.45	23.28	36.51	24.85	58.87

According to Table 1, we note that Google has the smallest average error in any case. Specifically, Google system presents WER error 16.60%, 20.45%, and 24.85% for Speaker A, B, and C respectively.

Taking into account the three error values, we calculate the average error for each system. The overall mean error was 38.1% for IBM, 20.63% for Google, and 36% for Wit. We observe that among the three, the smallest average error is reported by the Google tool (20.63%). Based on these measurements, it seems that Google's system responds better regardless of the speaker's type.

Then we recorded how many sentences from the ones processed by each system were fully recognized, without any errors.

Table 2. Percent (%) accurate prediction of all 60 sentence

IBM	Google	Wit
26.67	43.33	36.67

As we see in Table 2, Google presents a higher percentage of accurate forecasts (43.33%) than the other systems. Wit tool is the second in the rankings and the IBM the last. These measurements reveal the ability of ASR systems to recognize a sentence without any errors. This ability is essential if the ASR system is to be used in teaching English as a foreign language, as the application we intend to build. Our app must be able to fully recognize the pupil's speech to teach him the correct pronunciation, grammar, and pension, so we do not want to have an approximation but an accurate recognition.

To get a better and more accurate estimate of the three systems, we measured the Hper και Rper errors using the available tool on https://github.com/cidermole/hjerson. In Table 3, we present the results of these measurements.

Table 3. Rper και Hper error percentage (%)

	Speaker A			Speaker B			Speaker C		
	IBM	Google	Wit	IBM	Google	Wit	IBM	Google	Wit
Rper	28.45	16.81	24.21	42.24	20.44	26,64	30.66	26.94	61.78
Hper	28.60	13.35	19.79	42.09	19.53	24,17	32.97	22.29	52.99

According to Table 2, the Google ASR system presents the lower Rper and Hper errors for each speaker. Knowing that Rper and Hper errors are associated with adding or removing words in recognized speech, the measurements confirm that the Google system presents the fewest distortions.

Finally, we calculated the total mean errors per tool, which are shown in Table 4.

Table 4. Total mean errors (%)

	IBM	Google	Wit
Rper	23,90	21,40	37,54
Hper	34,55	18,39	32,32

Observing the values in Table 4, we find out that Google's ASR shows the lower average error for both metrics (Rper and Hper).

7 Conclusions

This paper presents some of the most widely known speech recognition tools currently in use. Three of them were then selected to study and compare their performance. The tools selected are IBM Watson, Google API, and Wit. All three tools were tested on the same audio data files and not in a continuous speech stream. The audio files involved texts that occur very often when teaching English as a foreign language. The WER, Hper, and Rper error metrics were measured to evaluate the tools. The results showed that the Google tool is superior to the other two. The Google tool was able to predict most of the sentences accurately, and it presented the smallest error percent. In the future, the size of the data set could be increased by adding more sentences to increase the reliability of the measurements and conclusions. Other tools could also be tested to allow us to decide which one is most suitable for our application.

Acknowledgements. This work is partially supported by MPhil program "Advanced Technologies in Informatics and Computers", hosted by Department of Computer Science, International Hellenic University.

References

1. Anusuya, M.A., Katti, S.K.: Speech Recognition by Machine, A Review. ArXiv10012267 Cs, January 2010
2. Sharma, F., Wasson, S.G.: A Speech Recognition and Synthesis Tool : Assistive Technology for Physically Disabled Persons (2012)
3. Using Statistical Methods in a Speech Recognition System for Romanian Language - ScienceDirect. https://www.sciencedirect.com/science/article/pii/S1474667015373067. Accessed 05 May 2019
4. Automatic Recognition of Spoken Digits. J. Acoust. Soc. Am. **24**(6). https://asa.scitation.org/doi/abs/10.1121/1.1906946. Accessed 05 May 2019
5. Britain scores in easier man-machine communication. Sens. Rev. **1**(4), 172–173 (1981)
6. Upton, J.: Speech recognition for computers in industry. Sens. Rev. **4**(4), 177–178 (1984)
7. Applications and potential of speech recognition. Sens. Rev. 4 (1983)
8. US7363228B2 - Speech recognition system and method - Google Patents. https://patents.google.com/patent/US7363228B2/en. Accessed 11 May 2019
9. Alyousefi, S.: Digital Automatic Speech Recognition using Kaldi, Thesis (2018)
10. Srikanth, R., Salsman, J.: Automatic pronunciation scoring and mispronunciation detection using CMUSphinx. In: Proceedings of the Workshop on Speech and Language Processing Tools in Education, Mumbai, India, pp. 61–68 (2012)
11. Karpagavalli, S., Chandra, E.: A Review on Automatic Speech Recognition Architecture and Approaches (2016)
12. Shmyrev, N.: CMUSphinx Open Source Speech Recognition, CMUSphinx Open Source Speech Recognition. http://cmusphinx.github.io/. Accessed 05 May 2019
13. Morbini, F. et al.: Which ASR should I choose for my dialogue system? In: Proceedings of the SIGDIAL 2013 Conference, pp. 394–403 (2013)
14. Lamere, P., et al.: The CMU Sphinx-4 Speech Recognition System, p. 4 (2003)
15. Satori, H., ElHaoussi, F.: Investigation Amazigh speech recognition using CMU tools. Int. J. Speech Technol. **17**(3), 235–243 (2014)
16. Povey, D., et al.: The Kaldi Speech Recognition Toolkit (2011)
17. Këpuska, V.: Comparing speech recognition systems (Microsoft API, Google API And CMU Sphinx). Int. J. Eng. Res. Appl. **07**(03), 20–24 (2017). https://doi.org/10.1007/s10772-014-9223-y
18. Dragon NaturallySpeaking, Wikipedia, 17 April 2019
19. Baker, J.M.: DragonDictatetm-30K: natural language speech recognition with 30,000 words. In: Presented at the First European Conference on Speech Communication and Technology, Paris, France, September 1989. Accessed 26 Apr 2019
20. Amazon Transcribe – Automatic Speech Recognition - AWS, Amazon Web Services, Inc. https://aws.amazon.com/transcribe/. Accessed 26 Apr 2019
21. erhopf: Speech-to-text with Azure Speech Services - Azure Cognitive Services. https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/speech-to-text. Accessed 05 May 2019
22. Wit.ai. https://wit.ai/. Accessed 05 May 2019
23. Speech Recognition API - Twilio. https://www.twilio.com/speech-recognition. Accessed 05 May 2019
24. Houndify | Add voice enabled, conversational interface to anything. https://www.houndify.com/. Accessed 05 May 2019
25. IBM Watson | IBM. https://www.ibm.com/watson. Accessed 05 May2019
26. Sharp, R.D., et al.: The Watson speech recognition engine. In: 1997 IEEE International Conference Acoustics Speech Signal Process

27. Goffin, V., et al.: The AT&T WATSON Speech Recognizer. In: Proceedings of ICASSP 3905 IEEE International Conference Acoustics Speech Signal Process (2005)
28. Adorf, J.: Web Speech API (2013)
29. Text to speech online. https://www.naturalreaders.com/. Accessed 05 May 2019
30. Text to Speech | TTS SDK | Speech Recognition (ASR). https://www.ispeech.org/. Accessed 05 May 2019
31. Sjölander, K., Beskow, J.: WAVESURFER - AN OPEN SOURCE SPEECH TOOL, p. 4 (2001)
32. Lee, A., Kawahara, T.: Recent development of open-source speech recognition engine julius. In: Proceedings: APSIPA ASC 2009: Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference, pp. 131–137, October 2009
33. Errattahi, R., El Hannani, A., Ouahmane, H.: Automatic speech recognition errors detection and correction: a review. Proc. Comput. Sci. **128**, 32–37 (2018)
34. Gaikwad, S.K., Gawali, B.W., Yannawar, P.: A review on speech recognition technique. Int. J. Comput. Appl. **10**(3), 16–24 (2010)
35. Morris, C., Maier, V., Green, P.: From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition, p. 4 (2004)
36. Seljan, S., Dunđer, I.: Combined Automatic Speech Recognition and Machine Translation in Business Correspondence Domain for English-Croatian (2014)
37. Madl, D.: Hjerson: An Open Source Tool for Automatic Error Classification of Machine Translation Output: cidermole/hjerson (2018)