A new framework for Arabic recitation using speech recognition and the Jaro Winkler algorithm

Souad Larabi-Marie-Sainte*, Betool S. Alnamlah, Norah F. Alkassim, Sara Y. Alshathry

Computer Science department,
College of Computer and Information Sciences
Prince Sultan University, Saudi Arabia
*Corresponding author: *slarabi@psu.edu.sa*

Abstract

Automated recitation plays an important role in improving self-learning. It is based on Speech/Text recognition. The research in Arabic speech recognition is very limited. The few existing applications are only based on the Holy Qur'an. This article proposed a new system (Samee'a - سميع) to facilitate memorizing any kind of text such that poems, speeches and the Holy Qur'an. Samee'a system is based on Google Cloud Speech Recognition API to convert the Arabic speech to text and Jaro Winkler Distance algorithm to determine the similarity between the original and converted texts. The system has been tested using 70 collected files ranging between 12 to 400 words and some chapters from the Holy Qur'an. The average similarity achieved 83.33% for the 70 files and 69% for the selected chapters of the Holy Qur'an. These results were enhanced to 91.33 % and 95.66% after applying preprocessing operations on the text files and the Holly Qur'an respectively. To validate the obtained results, two comparison studies were performed. The Jaro Winker distance was successfully compared to the cosine and the Euclidean distance. In addition, the proposed system outperformed the related work with an improvement of the similarity reaching 5% when using section 30 of the Holy Qur'an. Finally, the user experience testing was carried out by 10 users of different ages (between 5 and 50-year-old) using small texts and some small chapters of the Holy Qur'an. The proposed system proved its efficiency.

Keywords: Arabic language; automated recitation; Natural Language Processing; speech recognition; speech to text.

1. Introduction

Over the last years, speech-processing technology has been significantly evolved due to its potential in dealing with speech recognition, speech correction, and speech synthesis (Khan Wahab *et al.*, 2016). Speech recognition can be considered a very useful tool for recognizing and capturing voices (Alkhatib *et al.*, 2017). It is currently used in building systems for learning and memorization. Usually, memorization is performed through the traditional recitation process. It is based on a face-to-face method, which requires another person that listens to the reciter to ensure the memorization and correct the mistakes. Various studies investigated the recitation based on speech recognition in English and some languages.

The Arabic language is widely spoken (Larabi-Marie-Sainte *et al.*, 2019), the research in the field of the Arabic Speech Recognition is limited in comparison with other languages (El-mashed *et al.*, 2011). For example, the authors in (Alkhatib *et al.*, 2017) and (Yousfi *et al.*, 2016) introduced only the Holy Qur'an recitation. In (Ghadage *et al.*, 2016), the authors designed a multi-language speech-to-text conversion system focusing on Marathi –Indian-English, Marathi-English to extract, characterize and recognize the information about speech. However, all the presented studies and surveys investigated either the recitation of the Holy Qur'an in the Arabic language or the recitation of other different languages. This paper proposed a new system called [Samee'a] to promote the learning/memorization of any kind of Arabic text. It supports the research of Arabic speech recognition by converting the speech signal to a sequence of words. Samee'a system employed Google Cloud Speech Recognition API. Google API has proved its efficiency in converting speech to text, and outperformed both Microsoft API and Sphinx-4 (Këpuska , 2017). The proposed system is based on three main steps. It provides the reciter the ability to upload any text file for memorization, record the recitation, and display the similarity results after comparing the uploaded text with the recorded one. The similarity result is obtained using the Jaro Winkler Distance algorithm. To enhance the results, the text files were pre-processed to remove the Arabic diacritics, punctuations, and any other noise ((Khan Wahab *et al.*, 2016), (Khan Khairullah *et al.*, 2016)). Using this self-learning tool, the reciter can easily perform the recitation skills at any time. To demonstrate the effectiveness of this system, 70 text files, with different lengths, were created and used for testing. In addition, some chapters of the Holy Qur'an were also tested to compare the obtained results with those provided in the existing studies. The contribution of this study is threefold.

1. Develop a new system for Arabic speakers to ease the self-learning and recitation in the Arabic language.

2. Provide an Arabic recitation tool tackling any kind of text, Holy Qur'an, poem, lesson, etc.

3. Support the Arabic speech recognition and the Arabic Natural Language Processing research fields.

This paper is organized as follows: Section 2 presents the related works. Section 3 includes the methodology. Section 4 introduces briefly Natural Language Processing. Section 5 discusses the experimental results. Section 6 investigates the comparison study. Section 7 displays Samee'a interface and the user experience testing. Finally, section 8 concludes the study.

2. Literature review

Nowadays, the techniques of speech recognition and speech-to-text conversion and vice versa are very common, but they are rarely in the Arabic language. It is known that how Arabic speech recognition is hard due to the pronunciation of Arabic letters. In the following, the speech-to-text /text-to-speech and speech recognition-related works are discussed.

2.1 Speech-To-Text

In (Muhammad *et al.*, 2012) the authors proposed a system called 'E-hafiz' that assists learners to recite the Holy Qur'an based on the idea of Tajweed rules. The system used MFCC feature extraction techniques to get the feature vectors of some specific verses read by some experts and stored in the system's database. For the evaluation, three groups of reciters men, women and children were chosen. The accuracy of the three categories were 92% for men, 90% for children and 86% for women.

In (Reddy *et al.*, 2013), the authors presented an android application that converts voice to text to be sent as an SMS message to the entered phone number. This helped handicapped, deaf and blind people. The speech recognition technique was based on Hidden Markov Model (HHM) and Google's servers. This idea helps with memory saving and fasts the recognition process rather than installing a complex software. The database used contains more than 230 billion words. The system compared each oral word recorded with the saved word on the server. The authors have not presented how the results are accurate or how many users are satisfied. For future work, they planned to apply it for more than one language.

In (Ahsiah *et al.*, 2014), the authors described the Mel-Frequency Cepstral Coefficient and Vector Quantization (MFCC-VQ) procedure to develop a speech recognition system for Qalqalah Tajweed Checking rule. The main objective of this research was to help students to revise and recite the holy Qur'an properly by themselves and to recognize the types of bouncing sound in both Qalqalah Sughrah and Qalqalah Kubrah on the five letters (د ب ج ط ق). The system consists of four main modules, including the input Module, training module, testing module and analysis module. The overall real-time factor outperformed the conventional MFCC algorithm by 86.928%, 94.495% and 64.683% for males, females, and children respectively. The recognition accuracy obtained was 83.9% for males, 82.1% for females, and 95.0% for children.

In (Yousfi *et al.*, 2016), the authors have discussed the progress of speech recognition with the Holy Quran; where various applications used the technology for reciting, reading and learning. They have shown an overview of speech recognition techniques that have been used. As a result, they figured out that the best technique to use for feature extraction - one step of speech recognition - is Mel-Frequency Cepstrum Coefficients (MFCC) and the best method for the classification feature is the Hidden Markov Model (HMM).

In (Ghadage *et al.*, 2016), the authors have designed a multi-language speech-to-text conversion system. It was focused on Marathi –Indian, English, Marathi-English mix speech using Mel-Frequency Cepstrum Coefficients (MFCC) technique for feature extraction. The system has been tested with 1200 samples and achieved a high accuracy between 88% to 92% for the Marathi, English and Marathi English mix.

In (Alkhatib *et al.*, 2017), the authors presented a mobile application that helps children of non native Arabic speakers to learn, reciting the Holy Qur'an by detecting incorrect pronouncing words. This was done by removing the silence of their recordings then comparing it with multiple correct recording, using a modified version of Dynamic Time Warping algorithm. As a result, 100 teachers have answered a survey to check the performance of the system, which showed that 86% of them were satisfied and approved the ability of the application to improve children's skills in studying the Holy Quran.

In (Këpuska , 2017), the authors displayed a comparison between the commercial speech recognition tools that convert speech-to-text. They have presented a comparison between Google API, Microsoft API and Sphinx-4 that focused on using audio recording then detecting word error rate (WER) to judge the tool. The results showed that Google API achieved 9% WER, where Microsoft API achieved 18% and Sphinx achieved 37%.

In (Trivedi *et al.*, 2018), the authors presented different algorithms and techniques to achieve conversion from speech to text and vice versa. Speech-to-text and speech recognition followed the same steps. Various techniques such as Hidden Markov Model (HMM) with two metrics (Recognition Speed and Recognition Accuracy) were used. Then, the Artificial Neural Network was applied for the Classification with Cuckoo Search Optimization technique to remove noise and improve communication and recognition. The authors figured out that the most important step in speech recognition is pre processing to remove the unwanted waves. HMM is an excellent technique for converting speech to text because of their computational feasibility.

In (Gerhana *et al.*, 2018), the authors presented an application that helps memorize the Holy Quran by shuffling verses of the Quran using the Fisher-Yatis algorithm. The tool starts recording the recitation, after that it converts the voice to text then compares it with a version of the Holy Quran that has been added previously to check similarity. The accuracy was about 91% with an average running time of 1.9 ms.

Table 1. Existing works dealing with ths Arabic language

	Main feature	Evaluation metric	Dataset	Evaluation result
(Ahsiah *et al.*, 2014)	Reciting the Holy Qur'an and focusing on Tajweed Qalqalah rule Checking	Accuracy	Sourate Al-Ikhlas Sourate An-Nas Sourate Al-Fatihah	82.1-95% 72-93% 86.4 - 91.95%
(Alkhatib *et al.*, 2017)	Reciting the Holy Qur'an	Accuracy	Not mentioned	Men: 92% Children: 90% Women: 86%
(Elsayed *et al.*, 2019)	Holy Quran Tajweed rules according to "Hafs from Asim reading".	Recall Precision F-measure	Sourate Al-Ikhlas	92% 81% 86%
(Gerhana *et al.*, 2018)	Help memorize Al-Qur'an	Jaro Winkler distance algorithm	Sourate Al- Kautsar Sourate Al- Buruj Juz 30 of the Holy Quran	100% 100% 91%

In (Elsayed *et al.*, 2019), the authors proposed a general automatic system evaluating Quran recitation according to "Hafs reading". The aim was to solve the problem of evaluating all intonations (Tajweed) in addition to evaluate a set of Quran segments in the right arrangement of reading. The system used MFCC for feature extraction and Vector Quantization (VQ) for dimension reduction, in addition to the Quran ontology prepared for Quranic speech to support speech recitation recognition of the Quran. The recall achieved 92%, the precision was about 81%, and the F-measure was about to 86%.

2.2 Text-To-Speech

In (Hamad *et al.*, 2011), the authors discussed how rarely text-to-speech system has been implemented in Arabic. They developed guidelines for Arabic speech synthesis. They presented a new text-to-speech (TTS) system for Arabic based on the allophone concatenation method. The input was any text while the output was available in one male voice. The allophone defines the variations in phonemes, where a phoneme is the smallest unit of sound in speech. They were able to convert the entered Arabic text to speech to signals. To assess sound quality and pronunciation, they have made a survey to test a group of 20 persons with different language knowledge. The results showed that the speech was so natural and the quality was acceptable.

In (Alrouqi *et al.*, 2016), The authors presented a framework to build an Arabic Navigation System for blind people. The text-to-speech technique was used. They compared five Arabic text-to-speech (TTS) synthesizers for mobile devices and evaluated their intelligibility and naturalness using VoiceOver, Uspeech, Acapela, Adel, and SVOX synthesizers. VoiceOver got the highest score of 93.75%.

In (Oumaima *et al.*, 2018), the authors presented a web-based platform that helps kids with dictation. The text-to-speech API will convert the written text to a voice so that the student can hear what he wrote. Words with misspelling will be highlighted to detect the mistakes. The study was tested by 30 students from third, fourth and fifth grades. The results showed that the number of vocal errors and the corrected vocal errors were very similar.

2.3 Discussion

To sum up, the number of Arabic Speech recognition applications is less than the number of the existing applications in other languages. Moreover, all the presented studies in the Arabic language discussed the recitation of the Holy Qur'an as displayed in Table 1. Our contribution consists of developing a new system that targets the Arabic speakers for self-learning and memorization using the speech-to-text technique. The proposed system supports any type of text and not only the Holy Qur'an.

3. Natural language processing

Natural Language Processing (NLP) is the branch of artificial intelligence (AI) that aims at inventing theories, discovering techniques and building software that can understand, analyze and generate human languages in both written and spoken contexts. The NLP techniques are parsing language input (word, sentence, text, dialogue) according to the rules (derivational rules, inflectional rules, grammatical rules, etc.) and resources (like lexicon, corpus, dictionary) of the target language (Moath *et al.*, 2014). Arabic Natural language deals with Arabic language and involves both text (for example (Larabi-Marie-Sainte *et al.*, 2019) and (Al-Saleh *et al.*, 2021)) and speech processing (Khan Wahab *et al.*, 2016).

4. Methodology

The methodology consists of three main parts: Google Cloud Speech Recognition API, where the user enters his/her voice using a microphone then the system converts it to text. The second part is the Jaro Winkler Distance algorithm to compare the recited text with the uploaded text file, and then present the similarity percentage. The third part is Text Preprocessing to show the necessity of these operations and how they can overcome the limitations of both Google API and the Jaro Winkler Distance algorithm.

4.1 Google Cloud speech recognition API

Machine Learning is part of the Google Cloud Platform when it comes to building speech recognition software. Speech recognition (SR) is the process when a computer takes voice signal (from a microphone) and transforms it into words. SR has five stages. 1) The cleaning step, where the recorded signals are cleaned and separated from unvoiced speech and then discard unnecessary or irrelevant information (Yousfi *et al.*, 2016). 2) Feature extraction, to get utterance properties that have acoustic correlations in the speech signal (Aggarwal *et al.*, 2008). 3) The acoustic modeling, where the model links the observed speech signal with the expected phonetics of the hypothesis sentence. 4) The language model, which has the structural constraints in the language for the occurrence probabilities. It induces the probability of a word occurring after a sequence of words. 5) Features classification, this step will compare the unknown test pattern with each reference pattern in the sound class and compute a similarity measure between them (Saksamudre *et al.*, 2015). To support a global user base, Google API can understand up to 120 languages including the Arabic language. Google's API is used in this application to transform the recorded audio into text. Google cloud speech to text has three primary methods for recognizing speech. The first method is Synchronous Recognition that sends the speech signals to the speech-to-text API The API pre-processes the signals and returns the result. This method accepts data with 1 minute or less only. The second method is Asynchronous Recognition, which is the same as Synchronous Recognition but with a Long Running Operation (longer than 1 minute). Unlike synchronous method, this method can be processed while other operations are running. Last method is Streaming Recognition, which is designed for the real-time identifying purposes. Unlike the synchronous and asynchronous calls where both configuration and audio can be sent together in one request, this method requires sending multiple requests each time, In other words, the live audio taken from a microphone is immediately translated while the user is still speaking. In this study, the Asynchronous Recognition is applied since the recitation may take more than 1 minute, and there is no need for real time identification.

4.2 Jaro Winkler Distance algorithm

It consists of calculating the distance between two strings sequence to check the similarity of two words. The measurement scale is 0 to 1, 1.0 is a positive match and 0.0 is the least likely. It is done in three steps. Firstly, calculating the string length. Then, count the character in both words. Finally, check the number of character transpositions. Jaro Winkler uses the formula in Equation 1 to calculate distance.

$$dj = \frac{1}{3} * (\frac{m}{|S^1|} + \frac{m}{|S^2|} + \frac{m-t}{|m|}) \tag{1}$$

$|S^1| and |S^2|$: are the length of the first and second strings. m: is the total of matches characters, even the unordered one. t: is the half number of character transpositions, matched characters that are not in the same order. Note that matched characters in two strings cannot be further away in position than $(\frac{max|S^1|,|S^2|}{2}) - 1$ to be considered for matching. The algorithm uses a prefix scale p that provides accurate judgments, defined in Equation 2

Table 2. Empirical Example to carry out the Jaro Winker Distance Algorithm

	String 1 | S1|	String 2 |S2|	m	t	Result dj	l	Result dw
1	٦- اوراقة	٦- اوراقة	6	1	0.9444	3	0.96
2	٤- شجرة	٥- جراحي	2	0	0.6333	0	0.633

	dj formula	dw formula
1	$\frac{1}{3} * (\frac{6}{6} + \frac{6}{6} + \frac{6-1}{6})$	0.944+(3*0.1*(1-0.944))
2	$\frac{1}{3} * (\frac{2}{4} + \frac{2}{6} + \frac{2-0}{2})$	0.633+(0*0.1*(1-0.633))

$$dw = dj + (lp(1 - dj)) \tag{2}$$

Where: dw: is the Jaro Winker formula. dj: is the result of the similarity between two strings after comparison. p: is a constant defining how much the score is adjusted upwards to have common prefixes, the standard value, according to Jaro, is p=0.1. l: is the length of the similar prefix, checked from the start of a string up to the 4^{th} character maximum (Gerhana *et al.*, 2018). To understand the Jaro Winker Distance Algorithm, an example is given in Table 2.

4.3 Text Preprocessing

This stage involves a set of operations that are performed on the original text because Google API converts the voice to a preprocessed text. It consists of the following:

1. Remove the diacritics: because the Jaro Winkler Distance does not ignore diacritics while comparing two texts. So, they are removed to increase the comparison performance.
2. Remove the punctuation: because they will not appear in the converted text as they are not pronounced by the reciter.
3. Remove the white space: as it affects the comparison results.
4. Remove the noise: such as / العربيـــة / will be converted to / العربية /
5. Remove the prefixes /Suffixes: the Google API does not detect some confused Arabic letters such as letter "ta" / ـة / and the letter "ha"/ ـه /. The first letter can be pronounced either "ta" or "ha". The Google API always detects it as "ha" even if it was "ta". Also, Arabic language has two types of "a", [alif al-qaT'] which is with 'Hamza' /أ إ/ and [alif al-wasl] which is without 'Hamza'/ ا /. It was observed that only 'alif' as [alif al-wasl] was detected.

Figure 1 displays these operations. The framework process is presented in Figure 2. The user recites the text using a microphone, then the system saves the voice in a file and various sound processing are carried out. After that, the sound file is sent to Google API to be translated into text. The text is returned to the application in JSON format. Then, the Preprocessing operations are employed in the uploaded text file. The last step is the comparison between the preprocessed text and recited text file using the Jaro Winkler Distance algorithm. Finally, the similarity result is calculated and shown to the user.

5. Experimental results

To ensure that the proposed methodology achieves its intended goal, different experiments were conducted. This section starts with the data collection, then discusses three experiments. The first experiment demonstrates the efficiency of the Google API in translating the Arabic recitation into Arabic text. The second experiment shows the similarity results provided by the Jarro-Winker algorithm before and after applying the preprocessing operations. The third experiment involves testing the Holy Qur'an with

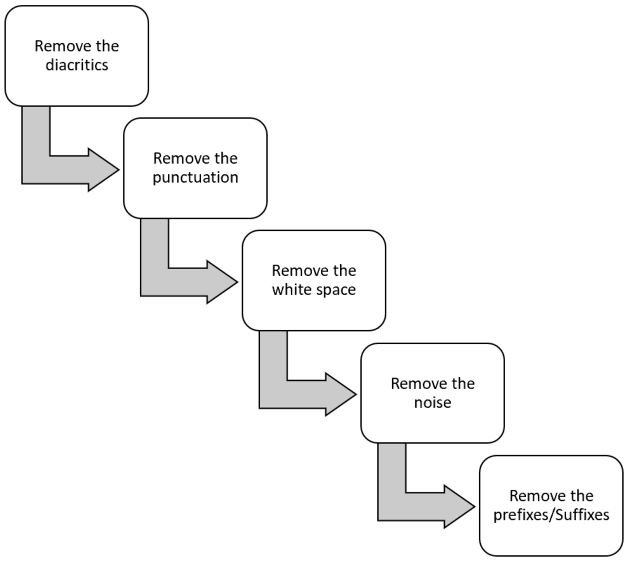


Fig. 1. Text Preprocessing operations

Table 3. File Level's classification

ID	File Level	Number of words	Number of files
1	Small	Less than 50	40
2	Medium	50 - 140	20
3	Large	141 and above	10

and without performing the preprocessing operations. For the implementation phase, the main Python library, PyArabic, was used for Arabic processing. It provides the basic functions to manipulate Arabic letters and text, this library has been used for all the preprocessing operations used in this study.

5.1 Data Collection

The dataset used in this study was collected and available in (`https://drive.google.com/file/d/1sTVZRkWud0rVfu-XhgPbZCtobu9i6BBz/view?usp=sharing`). Seventy files have been collected from different open sources. The files are from different categories like kid's stories, poems, articles and the Holy Qur'an. They have been classified into three different levels, including small, medium, and large files. Figure 3 shows a sample text divided into three levels having a different number of words. The different levels are explained in Table 3. These levels have been chosen to test the extent of the Samee'a system on different types of texts and with different text's complexity stages. The experiments were performed by different ages of the reciters.

5.2 Experiment 1

The aim is to show that the Google Cloud Speech Recognition API well recognizes the Arabic speech. To this purpose, three objectives were investigated:

- Objective1: ensure that all the well pronounced letters are well recognized.
- Objective2: ensure that the short and long vowels are well recognized.
- Objective3: ensure that the Arabic letters outgoing from the same exit are well recognized when they are well pronounced.

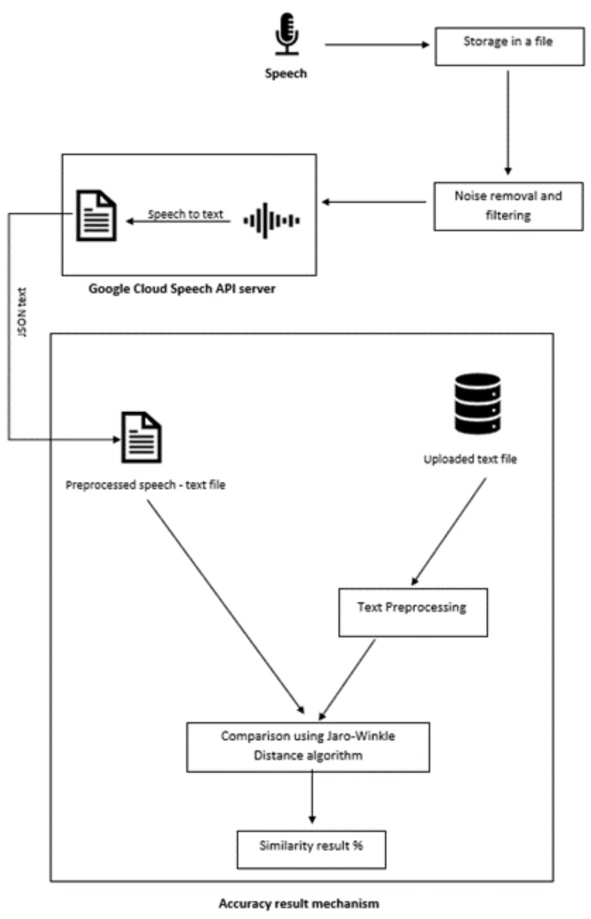

Speech
Storage in a file
Noise removal and filtering
Speech to text
Google Cloud Speech API server
JSON text
Preprocessed speech - text file
Uploaded text file
Text Preprocessing
Comparison using Jaro-Winkle Distance algorithm
Similarity result %
Accuracy result mechanism

Fig. 2. The mechanism of the Samee'a system

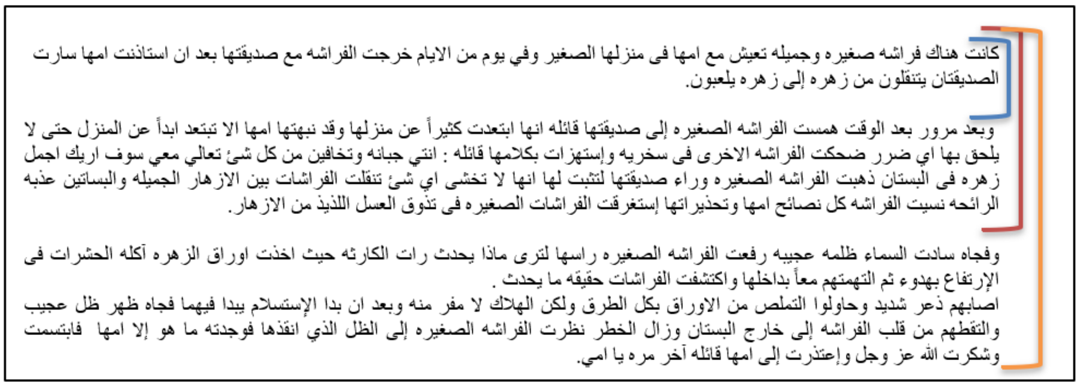
The experiment involved reading (instead of reciting) 35 texts from the collected text files (half of the provided text files, See Table 3) by Arabic native speakers. Then, the resulting file (converted from audio to text) was observed and checked to validate the aforementioned objectives. Reading the text files is performed correctly but sometimes with intentional errors.

After reading (with a correct way) the selected text files, it was noticed that Google API efficiently detected all the letters/words. Figure 4 shows an example of 100% detection. Thus, the first objective was fulfilled.

Moreover, it was noticed that the converted texts are written without the short vowels (diacritics), which means that the Google API does not support the diacritics. However, the long vowels (/ لد ا/) are detected if they are well pronounced. Figure 5 displays one example of this experience. Hence, objective 2 was investigated.

To check objective 3, two scenarios were performed. The first one was to correctly read complex texts with many confused letters (outgoing from the same or approaching exits) such that / ك ،ق/ , / ح ،ه/ , / ع ،غ/, .etc. It was noted that very few letters are not detected (around 3 letters over 20). The second scenario intended to read some texts with bad pronunciation (intentional errors). Figure 6 displays an example of this experience. The resulted file contained errors as expected. Consequently, confused letters are detected only if they are well pronunciation.

To conclude, this experiment demonstrated that the efficiency of Google API depends on good pronunciation and reading.

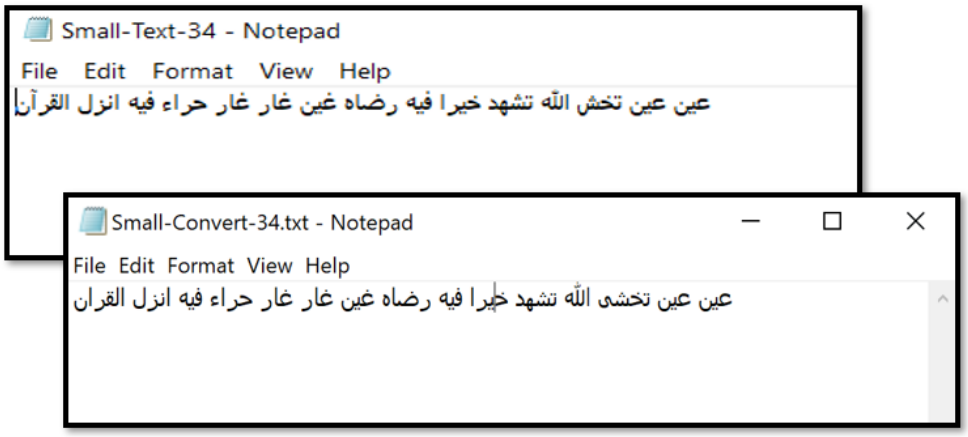


كانت هناك فراشه صغيره وجميله تعيش مع امها فى منزلها الصغير وفي يوم من الايام خرجت الفراشه مع صديقتها بعد ان استاذنت امها سارت الصديقتان يتنقلون من زهره إلى زهره يلعبون.

وبعد مرور بعد الوقت همست الفراشه الصغيره إلى صديقتها قائله انها ابتعدت كثيراً عن منزلها وقد نبهتها امها الا تبتعد ابداً عن المنزل حتى لا يلحق بها اي ضرر ضحكت الفراشه الاخرى فى سخريه وإستهزات بكلامها قائله : انتي جبانه وتخافين من كل شئ تعالي معي سوف اريك اجمل زهره فى البستان ذهبت الفراشه الصغيره وراء صديقتها لتثبت لها انها لا تخشى اي شئ تنقلت الفراشات بين الازهار الجميله والبساتين عذبه الرائحه نسيت الفراشه كل نصائح امها وتحذيراتها إستغرقت الفراشات الصغيره فى تذوق العسل اللذيذ من الازهار.

وفجاه سادت السماء ظلمه عجيبه رفعت الفراشه الصغيره راسها لترى ماذا يحدث رات الكارثه حيث اخذت اوراق الزهره آكله الحشرات فى الإرتفاع بهدوء ثم التهمتهم معاً بداخلها واكتشفت الفراشات حقيقه ما يحدث .
اصابهم ذعر شديد وحاولوا التملص من الاوراق بكل الطرق ولكن الهلاك لا مفر منه وبعد ان بدا الإستسلام يبدا فيهما فجاه ظهر ظل عجيب والتقطهم من قلب الفراشه إلى خارج البستان وزال الخطر نظرت الفراشه الصغيره إلى الظل الذي انقذها فوجدته ما هو إلا امها فابتسمت وشكرت الله عز وجل وإعتذرت إلى امها قائله آخر مره يا امي.

Fig. 3. Sample text file



Small-Text-34 - Notepad

File Edit Format View Help

عين عين تخش الله تشهد خيرا فيه رضاه غين غار غار حراء فيه انزل القرآن

Small-Convert-34.txt - Notepad

File Edit Format View Help

عين عين تخشى الله تشهد خيرا فيه رضاه غين غار غار حراء فيه انزل القران

Fig. 4. An example of Google API conversion: Perfect Arabic words recognition

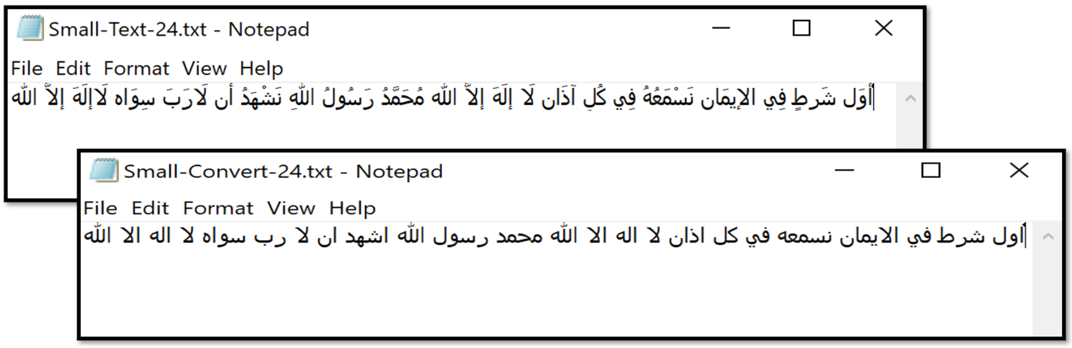
5.3 Experiment 2

As mentioned above, 70 files were used to test the proposed system. Table 4 shows the similarity results before and after applying the preprocessing. The table also includes the running time and some information about the used text file such as the level, the number of words, and the number of the test file to allow any reader to check or collect it from the dataset.

Table 4. Experiment 1-Similarity and run time results before and after applying the preprocessing

			Before Preprocessing		After Preprocessing	
Level	Test File #	Number of words	Similarity Result	Run Time	Similarity Result	Run Time
Small	1	23	0.87688	3.61732	1.0	4.22811
Small	2	16	0.84549	3.52537	1.0	3.22901
Small	3	22	0.90931	3.32654	0.92833	3.32647
Small	4	17	0.83106	3.39179	1.0	5.92216
Small	5	27	0.85387	5.07407	0.83450	9.69097
Small	6	22	0.90039	3.14730	0.94069	4.60508
Small	7	20	0.89899	3.09534	1.0	4.09458
Small	8	15	0.82521	2.92113	0.92023	6.09962
Small	9	20	0.82863	3.52796	0.97910	10.47554
Small	10	17	0.81975	2.71626	0.95241	4.08985
Small	11	33	0.89509	4.61103	0.90537	5.31696
Small	12	30	0.80312	4.62008	0.896203	7.01379
Small	13	22	0.86612	3.36361	1.0	3.97660

Small	14	24	0.86396	2.91833	0.913019	5.66293
Small	15	24	0.96263	3.56152	0.970358	4.32730
Small	16	19	0.80710	3.59690	1.0	3.27493
Small	17	20	0.80200	3.21670	0.97623	2.83460
Small	18	18	0.86326	3.09410	1.0	3.95990
Small	19	12	0.99427	1.68401	1.0	1.90216
Small	20	17	0.96265	2.74962	1.0	3.61752
Small	21	35	0.85103	3.62400	0.87482	3.03745
Small	22	19	0.88942	2.45489	0.91321	2.51097
Small	23	16	0.85746	2.58014	0.88977	2.70789
Small	24	15	0.92825	3.98225	0.96700	2.90023
Small	25	17	0.90813	2.66837	0.98327	2.66220
Small	26	17	0.93199	2.77364	0.95909	2.23938
Small	27	16	0.88893	2.88976	0.90926	2.59419
Small	28	17	0.89899	2.56377	1.0	3.57457
Small	29	46	0.84435	3.76199	0.854188	5.19949
Small	30	24	0.88404	2.89221	0.88968	2.77268
Small	31	24	1.0	3.11782	1.0	3.02829
Small	32	16	0.818917	2.45622	0.890864	2.48279
Small	33	14	0.90703	2.44323	0.908298	2.20252
Small	34	15	0.98543	2.76236	0.995720	2.31327
Small	35	15	0.99061	2.98273	1.0	2.51088
Small	36	24	0.87374	3.78221	0.90099	2.65363
Small	37	21	0.86855	3.23568	0.92431	2.70739
Small	38	13	0.971332	2.19332	1.0	2.65025
Small	39	14	0.980238	2.33628	1.0	2.37025
Small	40	13	0.832128	1.77261	0.96422	2.28943
Average			**88%**	**3.12581sec**	**95%**	**3.8264 sec**
Medium	41	100	0.78618	11.89413	0.90963	16.76075
Medium	42	116	0.86190	19.11728	0.89931	26.48759
Medium	43	112	0.88770	9.151897	0.95247	13.13580
Medium	44	139	0.61680	26.32314	0.83190	31.45933
Medium	45	97	0.71968	11.33177	0.87517	17.44564
Medium	46	122	0.76770	12.83897	0.85338	16.11669
Medium	47	115	0.79531	15.86353	0.99531	13.12489
Medium	48	122	0.81531	22.91785	0.84215	25.16138
Medium	49	140	0.79860	14.15330	0.93860	25.72329
Medium	50	137	0.79518	21.42546	0.88098	31.57457
Medium	51	77	0.88368	11.57489	0.93727	7.80486
Medium	52	111	0.84961	10.37192	0.884576	8.29505
Medium	53	68	0.849355	12.03203	0.89465	5.874893
Medium	54	97	0.837759	10.62012	0.86166	8.60130
Medium	55	120	0.888841	10.44083	0.90264	9.62636
Medium	56	71	0.873462	8.74923	0.956882	5.81076
Medium	57	74	0.843235	9.52837	0.850807	5.97167
Medium	58	80	0.832297	10.92273	0.903698	7.23172
Medium	59	70	0.871828	9.22783	0.933640	5.95164
Medium	60	68	0.904724	8.44623	0.957127	7.41575
Average			**82%**	**13.34657 sec**	**93%**	**3.8264 sec**
Large	61	148	0.71832	28.86499	0.81832	38.87629
Large	62	166	0.73923	27.40820	0.85446	27.96598
Large	63	244	0.80648	39.46840	0.88356	41.95432
Large	64	240	0.80193	54.10544	0.89543	62.94765
Large	65	209	0.81762	31.09710	0.88341	43.47032
Large	66	182	0.847134	17.31701	0.86010	32.31429
Large	67	245	0.839067	19.19812	0.87462	28.57876
Large	68	220	0.838290	17.97989	0.85035	24.10581
Large	69	401	0.833130	43.32693	0.86029	40.16558
Large	70	155	0.834759	17.77302	0.89333	17.3188
Average			**80%**	**29.65391 sec**	**86%**	**35.76978 sec**


Small-Text-24.txt - Notepad

File Edit Format View Help

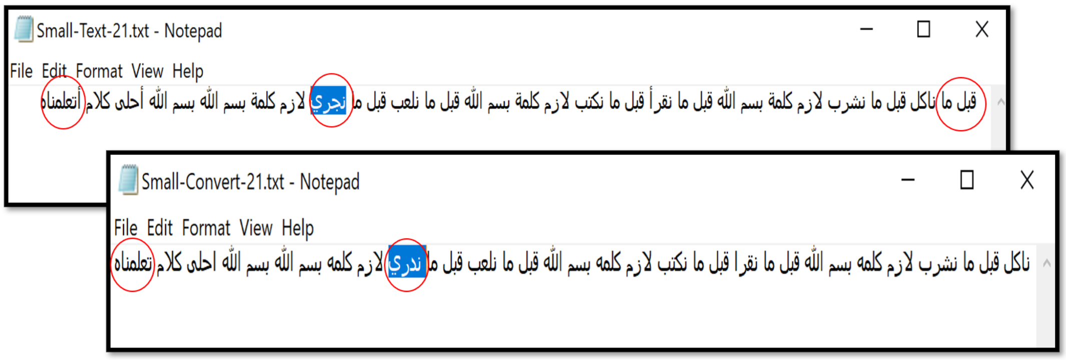
أَول شَرطٍ فِي الإيمَان نَسْمَعُهُ فِي كُلِ آذَان لَا إِلَهَ إِلاّ اللّه مُحَمَّدُ رَسُولُ اللّهِ نَشْهَدُ أن لَارَبَ سِوَاه لَاإِلَهَ إِلاّ اللّه

Small-Convert-24.txt - Notepad

File Edit Format View Help

اول شرط في الايمان نسمعه في كل اذان لا اله الا الله محمد رسول الله اشهد ان لا رب سواه لا اله الا الله

Fig. 5. An example of Google API conversion: Absence of diacritics in the converted text


Small-Text-21.txt - Notepad

File Edit Format View Help

قبل ما ناكل قبل ما نشرب لازم كلمة بسم الله قبل ما نقرأ قبل ما نكتب لازم كلمة بسم الله قبل ما نلعب قبل ما نجري لازم كلمة بسم الله بسم الله أحلى كلام أتعلمناه

Small-Convert-21.txt - Notepad

File Edit Format View Help

ناكل قبل ما نشرب لازم كلمه بسم الله قبل ما نقرا قبل ما نكتب لازم كلمه بسم الله قبل ما نلعب قبل ما ندري لازم كلمه بسم الله بسم الله احلى كلام تعلمناه

Fig. 6. An example of Google API conversion: Bad utterance of some words

As indicated in this table, the average similarity of the small, medium and large files are 88%, 82%, and 80% respectively. As for the running time, the small level took 3 seconds, and the medium level took 13 seconds, while it is 29 seconds for the large level. On the other hand, the average similarities of the 3 type files after the pre-processing stage are 95%, 90% and 87% respectively. The average running time for the small level is 4 seconds, 14 seconds for the medium level, and 34 seconds for the large level. This result clearly shows the proportional relationship between:

1. The number of words in each file and the similarity percentage. The similarity slightly decreases while the file size increases. This is because the proposed medium and large text files contain a certain level of complexity in terms of intricate words (with confused letters). Besides, experiment 1 showed that Google API produced some errors when dealing with Medium and Large text files. Figure 7 displays this relationship after and before applying the preprocessing.

2. The number of words in each file and the running time. More the file is large more time is spent to find the result.

Moreover, this experimental result shows that the preprocessing stage, enhances the similarity percentage by 7% as shown in Figure 5, but raises the average running time by 2.5 seconds After performing many test cases, we found that Jaro Winkler Distance does not take more than average 0.02 seconds to find the similarity, and in case of the similarity equals to 1, Jaro Winkler Distance takes 0 seconds.

A deep investigation of the importance of the preprocessing was done to determine and overcome the limitation of both the Google Cloud Speech Recognition API and the Jaro Winkler Distance algorithm. As displayed in table 4, files number 31 and 45 have the highest and lowest similarity percentages respectively, without performing the preprocessing. File 31 is small with 25 simple and clear words. The Google API recognizes all the words correctly due to their simplicity. In addition, the original text file didn't contain Arabic diacritics. While file number 45 is medium with 139 words containing an Arabic

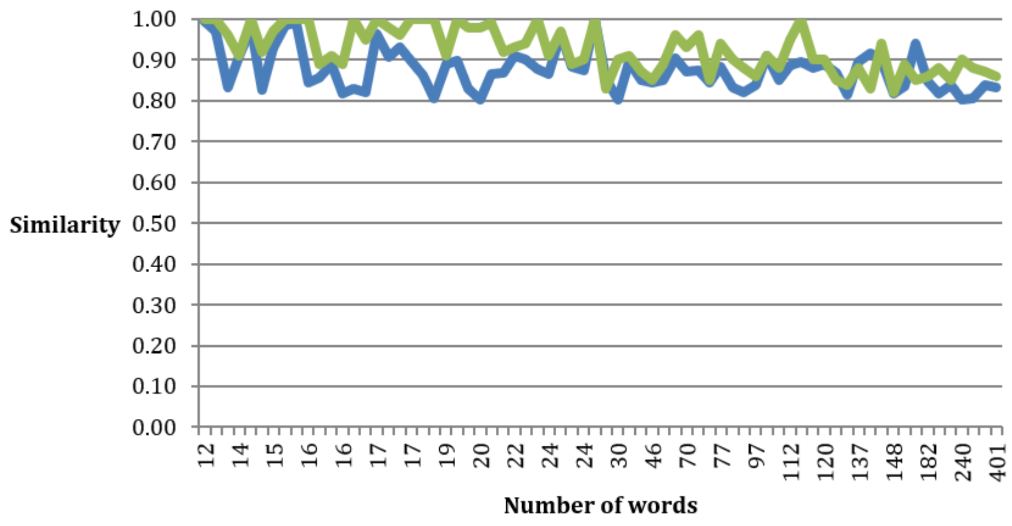


Fig. 7. The variation of the similarity percentages based on the number of words

complex poem. This file contains a lot of confusing words and the original text file has the Arabic diacritics. It is worth to noticing that Arabic diacritics are used to indicate recognizably the presence or absence of short vowels, distinguish long vowels from glides or diphthongs, and indicate geminate consonants [1]. Unfortunately, the Jaro Winkler Distance algorithm can't ignore these diacritics. For example, comparing two similar words: one with diacritics and the other without such as: / قلم/ and / قَلم/ will give a Similarity = 0.9249999999999999. And as expected, removing the diacritic improves similarity to 1. However, it is interesting to see that the similarity is not very low when having two words with a different letter such as / قلم/ and / قلب/ will give a Similarity = 0.8222222222222222.

5.4 Experiment 3

This experiment involves testing some chapters from the Holy Qur'an. A like the first experiment, the same file levels (small, medium, and large) were used as indicated in Table 3. However, the number of files used for each level type is 11, 8, 2 respectively. So, the experiment was conducted using a total of 32 chapters from the Holy Qur'an.

Table 5. Experiment 2-Similarity and run time results using the Holy Qur'an before and after applying the preprocessing

				Before Preprocessing		After Preprocessing	
Level	Test File#	Name and Verses of Al-Quran	# of words	Similarity Result	Run Time	Similarity Result	Run Time
Small	1	Al-Fatiha: 1-7	29	0.767647	2.21977	1.0	5.97192
Small	2	Quraish: 1-3	39	0.37963	3.11728	0.99537	3.53565
Small	3	Al-Humazah: 1-3	49	0.45771	3.83721	0.93632	4.29911
Small	4	Al-Takathur: 1-4	45	0.60118	2.18253	0.893376	7.66418
Small	5	Al-Qariaah:1-3	43	0.71322	1.29918	0.99217	3.34298
Small	6	Al-Qader: 1-3	47	0.88129	3.88232	0.997549	6.93542
Small	7	At-Teen: 1-4	40	0.77100	3.18236	1.0	3.72891
Small	8	Ash-Sharh: 1-4	36	0.87987	5.62983	1.0	7.58649
Small	9	Al-Alaq: 1-4	47	0.639750	5.01125	0.969750	5.89196
Small	10	Al-Balad: 1-4	41	0.882271	6.81721	0.982271	8.14294
Small	11	Az-Zalzalah: 1-3	32	0.89585	5.91283	0.96585	9.57014
Average				**72%**	**3.91743sec**	**98%**	**6.06088 sec**
Medium	12	Al-Kahf: 1-9	89	0.69219	8.47342	0.94530	23.2740
Medium	13	Al-Kahf: 10-16	99		0.68941	0.84536	23.329792

Medium	14	Al-Kahf: 17-21	139	0.68947	9.868973	0.86980	12.325084
Medium	15	Abasa: 1-4	70	0.87258	4.88273	1.0	4.90939
Medium	16	Ash-Shams: 1-4	71	0.87974	4.36555	1.0	4.80218
Medium	17	Al-Enfitar: 1-4	75	0.88876	3.79123	1.0	5.40777
Medium	18	Al-Haqqah:1-5	93	0.85373	6.98952	1.0	9.34521
Medium	19	Al-Qalam: 1-4	88	0.72971	6.19427	1.0	6.90862
Average				**80%**	**7.03009 sec**	**96%**	**12.32414 sec**
Large	20	As-Saf: 1-14	230	0.701531	19.61045	0.86638	18.41815
Large	21	Al-Mulk: 1-4	241	0.408293	17.46755	0.99923	16.44456
Average				**55%**	**18.53900 sec**	**93%**	**17.43135 sec**

As seen in Table 5, the average similarity percentages of the small, medium and large files before the pre-processing stage are 72%, 80%, 55% respectively. While the running time is 4 seconds for the small level, 7 seconds for the medium level and 18 seconds for the large level. On the other hand, the similarities of the 3 categories after the pre-processing stage are 98%, 96%, and 93%. The running time for the same categories are 6 seconds, 12 seconds and 17 seconds. We can see here the same relationship as the previous experiment between the number of words in each file and the similarity percentage, and also between the run time and length of the audio. Furthermore, the result was outstandingly enhanced after applying the pre-processing stage. The similarity is getting more accurate achieving an improvement of 26% for the small level, 16% for the medium, and 38% for the large level. The need for the preprocessing operations is remarkably observed in this experiment. This is because the chapters of the Quran contain a lot of diacritics, which affect the similarity.

6. Comparison study

This section discusses two comparison studies using two well-known similarity distances and the state-of-the-art-works.

6.1 Comparison with the Similarity distances

In this experiment, the Jaro Winker distance was compared to Cosine and Euclidean distances. It is worth to noticing that both metrics require numeric vectors for comparison purposes. Thus, some operations were performed on the original and converted texts as follows. Firstly, the aforementioned preprocessing operations were employed to both text files. Next, the stop-words were removed. Later, the stemming was applied to reduce the derived words to their root forms. Finally, the term frequency inverse document frequency was calculated to determine the frequency and the existence of each word in both text files. In fact, all the words detected in both documents are considered. Each word was represented by its frequency (the number of times it appeared in a document) or 0 when it does not exist in such a document. For example, if one word appears one time in the original text but was not mentioned in the converted text, then this word will be represented by 1 in the original file and 0 in the converted file. To perform this experiment, some files were taken from the collected data (20 small files, 10 medium, and 3 large files). Table 6 figures out the results of the evaluation measures and the time required to calculate the metrics. As displayed, the results of the Cosine metric are very close to those of the Jaro Winker distance. Whereas, the Euclidean distance results are very small. The Jaro Winker distance yielded better average of the similarity results (94% for Small texts and 87% for the large texts) than the Cosine measure (92% for small and 84% for the large texts) for small and large text files. However, the Cosine provided the highest average of the similarity (93%) when dealing with medium text files. For the running time, the Cosine and the Euclidean distance provide the similarity results in an insignificant time (an average of 0.005 sec). This is because the time displayed in this table does not include the preprocessing and text representation operations nor the time required by Google API to convert the file. It just comprises the comparison between two numerical vectors. However, the running time provided for the Jaro Winker distance (an average of 32 sec) includes the whole process (API conversion, the preprocessing, and the similarity calculation). Not to mention that the Jaro Winker does a comparison in terms of words and letters which requires time. Consequently, Jaro Winker is a competitive similarity metric that achieved excellent results in comparing two Arabic text files without performing text representation.

Table 6. Comparison study using Cosine and Euclidean distance metrics

			Jarro-Winker		Euclidean distance		Cosine	
Level	Test File #	Number of words	Similarity Result	Run Time	Similarity Result	Run Time	Similarity Result	Run Time
Small	21	35	0.87482	3.03745	2	0.0115	0.9661	0.0080
Small	22	19	0.91321	2.51097	2.4495	0.0102	0.7273	0.0111
Small	23	16	0.88977	2.70789	0	0.0070	1	0.0050
Small	24	15	0.96700	2.90023	2	0.0116	0.8903	0.0040
Small	25	17	0.98327	2.66220	2.45	0.0100	0.7526	0.0060
Small	26	17	0.95909	2.23938	1	0.0030	0.9487	0.0020
Small	27	16	0.90926	2.59419	0	0.0039	1	0.0030
Small	28	17	1.0	3.57457	3.61	0.0030	0.8941	0.0020
Small	29	46	0.854188	5.19949	0	0.0040	1	0.0030
Small	30	24	0.88968	2.77268	0	0.0040	1	0.0030
Small	31	24	1.0	3.02829	0	0.004	1	0.003
Small	32	16	0.890864	2.48279	2.65	0.005	0.7206	0.0020
Small	33	14	0.908298	2.20252	2	0.0040	0.8462	0.0030
Small	34	15	0.995720	2.31327	1.41	0.0040	0.9375	0.0040
Small	35	15	1.0	2.51088	1.41	0.0040	0.9375	0.0030
Small	36	24	0.90099	2.65363	2	0.0040	0.9091	0.0020
Small	37	21	0.92431	2.70739	2	0.0040	0.9091	0.0020
Small	38	13	1.0	2.65025	0	0.0040	1	0.0030
Small	39	14	1.0	2.37025	1.7320	0.0070	0.8771	0.0030
Small	40	13	0.96422	2.28943	0	0.0040	1	0.0020
Average			**94%**	**2.592 sec**	**1.34**	**0.006 sec**	**92%**	**0.004 sec**
Medium	51	77	0.93727	7.80486	0	0.0050	1	0.0030
Medium	52	111	0.884576	8.29505	3.7417	0.0163	0.9314	0.0153
Medium	53	68	0.89465	5.874893	2.66	0.0162	0.9321	0.0052
Medium	54	97	0.86166	8.60130	4	0.0070	0.9177	0.0060
Medium	55	120	0.90264	9.62636	4.90	0.0070	0.9250	0.0050
Medium	56	71	0.956882	5.81076	2	0.0060	0.9565	0.0040
Medium	57	74	0.850807	5.97167	4.47	0.0084	0.8306	0.0070
Medium	58	80	0.903698	7.23172	3.46	0.0040	0.9178	0.0050
Medium	59	70	0.933640	5.95164	2.45	0.0060	0.9651	0.0045
Medium	60	68	0.957127	7.41575	3	0.0050	0.9091	0.0050
Average			**91%**	**7.258 sec**	**3**	**0.007 sec**	**93%**	**0.005 sec**
Large	66	182	0.86010	32.31429	6.63	0.0064	0.8698	0.0050
Large	68	220	0.85035	24.10581	6.40	0.005	0.9191	0.0050
Large	69	401	0.86029	40.16558	10.91	0.0040	0.7301	0.005
Average			**87%**	**32.20 sec**	**7.98**	**0.005 sec**	**84%**	**0.005 sec**

6.2 Comparison with the state of the art studies

In the following, the most similar study presented in (Gerhana *et al.*, 2018) was chosen for the comparison study. This work was selected because it used the same similarity metric and provided the names of chapters tested including. Al- Kautsar Al- Buruj, and all the chapters of section 30 (Juz 30). Accordingly, we have tested the same chapters using the proposed system. Table 7 shows the similarity results of all the chapters included in section 30 (Juz 30) using the proposed system.

Table 8 figures out the comparison between the obtained results and those presented in (Gerhana *et al.*, 2018).

As it can be seen in Table 8, both studies achieved a similarity of 100% for chapters Al- Buruj and Al- Kautsar. However, (Gerhana *et al.*, 2018) reached a similarity of 91% whereas the proposed system yielded a similarity of 96% when reciting Juz 30. Thus, the proposed system has a better result.

7. The Graphical user interface of the proposed system

The main contribution of this article is the implementation of a new tool for Arabic recitation. This tool is available and can be downloaded on (`https://drive.google.com/file/d/`

Table 7. Similarity result of Juz 30 of the Holy Qur'an using the proposed system

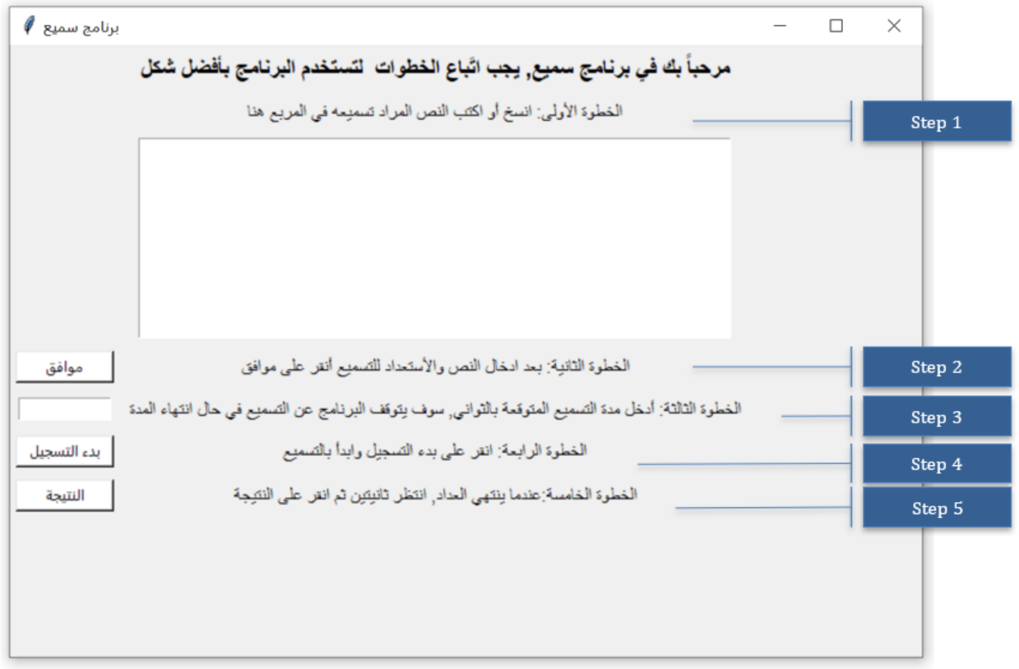
Test file #	Sourat	Result
1	An- Naba	0.87547
2	An-Nazi'aat	0.86031
3	Abasa	0.88415
4	At-Takwir	0.88795
5	Al-Infithar	0.93963
6	Al- Muthaffifin	0.90740
7	Al-Insyiqaaq	0.90196
8	Al-Buruuj	1.0
9	Ath-Thaariq	1.0
10	Al-'Ala	0.88352
11	Al-Ghasyiyah	0.88178
12	Al-Fajr	0.87892
13	Al-Balad	0.90242
14	Asy-Syams	0.90796
15	Al- Lail	0.99918
16	Add-Dhuha	0.99853
17	Al-Inshirah	0.99780
18	At-Tiin	0.99847
19	Al-'Alaq	0.97338
20	Al- Qadr	0.92872
21	Al-Bayyinah	0.945957
22	Az-Zalzalah	0.99231
23	Al-'Aadiyaat	1.0
24	Al- Qaari'ah	0.97401
25	At-Takaatsur	1.0
26	Al- 'Ashr	1.0
27	Al-Humazah	1.0
28	Al-Fiil	0.91598
29	Quraisy	1.0
30	Al-Maa'uun	1.0
31	Al-Kautsar	1.0
32	Al-Kaafiruun	1.0
33	An- Nasr	1.0
34	Al-Lahab	0.99737
35	AlIkhlas	1.0
36	Al-Falaq	1.0
37	An-Naas	1.0

Table 8. Comparison study using some chapters of the Holy Qur'an

Sourat	Related work (Gerhana *et al.*, 2018)	Samee'a System
Al- Buruj	100%	100%
Al- Kautsar	100%	100%
Section (Juz) 30 of the Holy Quran	91%	96%

`1seBRBJC10QoPbc7dkDtHabbr8JIpncWz/view?usp=sharing)` following the path: Sameea\Recite\ArabicApplication\Main\dist and run the exe file.

This section introduces the interface of the proposed tool and shows how it works. In addition, the user testing was performed to show the efficiency of this tool.



برنامج سميع

مرحباً بك في برنامج سميع, يجب اتّباع الخطوات لتستخدم البرنامج بأفضل شكل

الخطوة الأولى: انسخ أو اكتب النص المراد تسميعه في المربع هنا

الخطوة الثانية: بعد ادخال النص والأستعداد للتسميع أنقر على موافق

الخطوة الثالثة: أدخل مدة التسميع المتوقعة بالثواني, سوف يتوقف البرنامج عن التسميع في حال انتهاء المدة

الخطوة الرابعة: انقر على بدء التسجيل وابدأ بالتسميع

الخطوة الخامسة:عندما ينتهي العداد, انتظر ثانيتين ثم انقر على النتيجة

Fig. 8. The Graphical user interface of the proposed system

Figure 8 shows the proposed system. The GUI of the application consists of 5 steps. 1. The text entry is where the user needs to type or paste the text into it. 2. The "Ok" button allows system to save the text and preprocess it. 3. The counter allows the user to set the time (in seconds) needed for the recitation. The counter will be countdown to let the user know when to stop reciting. Once the counter is null, the system stops saving and proceeds to the recognition phase. 4. The "Start recitation" button allows the system to start the speech recognition process. When this button is clicked, the system will hide the text entered from the user. 5. The "Result" button allows the system to display the result of the recitation.

7.1 User experience testing

A user experience testing has been conducted for 2 different texts as displayed in Table 9

As seen in this table, the average result is 98% . We can observe that the utterance of the user affects the result. Younger users may have non clear letter exits, which affects the Google Speech recognition for finding the correct matched words. Yet, the obtained result is satisfactory.

Table 9. User Experience testing performed using the proposed system

Text type	Text File number	User age	Result
Small Text	7	7 years	93.2452%
		11 years	95.8876%
		16 years	100%
		26 years	100%
		50 years	98.5432%
Small Chapter from the Holy Qur'an	33	7 years	94.5427%
		11 years	99.5432%
		16 years	100%
		26 years	100%
		50 years	100%
Average		**98%**	

8. Conclusion

This paper presents a new application for Arabic recitation. The proposed system comprises two parts, Arabic speech recognition using Google API and finding the similarity between the recognized speech and the text file using the Jaro Winker algorithm. A GUI was developed using Python development language. The proposed system was tested using various types of texts from simple and short to long and complex. Three experiments were done. The first one involved demonstrating the effectiveness of Google API in converting Arabic speech into text. The results showed that the conversion was successfully performed but depends on the pronunciation. The two other experiments consisted of two parts, with and without text preprocessing using the collected datasets and the Holly Qur'an respectively. It was shown that the preprocessing operations mainly increased the similarity results. Moreover, it was proved that the Jaro Winker distance is a competitive metric compared to the Cosine and Euclidean distance. Beside, the proposed study outperformed the existing study using the Holly Qur'an. The last experiment consisted of user acceptance testing, the obtained results were prominent. It was noted that the user's utterance affects the recognition and similarity results. Finally, this work could be extended to enhance the proposed system. For example, displaying to the user the wrong uttered words could enhance both the user's recitation/reading skills and the similarity results. Moreover, the user's utterance could be more investigated to enhance Arabic speech recognition using Google API.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the support of prince sultan university. This work was also supported by the artificial intelligence and data analytics lab (AIDA), prince sultan university, Riyadh, Saudi Arabia.

References

Ahsiah, I., Mohd, Y.I.I., Noorzaily, M.N., Zaidi, R., & Zulkifli, M.Y. (2014). MFCC-VQ Approach For QalqalahTajweed Rule Checking. *Malaysian Journal of Computer Science*, 27(4), pp. 275–293. https://ejournal.um.edu.my/index.php/MJCS/article/view/6829.

Aggarwal, R. K. & Dave, M. (2008). Implementing a Speech Recognition System Interface for Indian Languages. *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages/* https: //aclanthology.org/I08-3017.

Alkhatib, B., Kawas, M., Alnahhas, A., Bondok, R. & Kannous, R. (2017). Building an assistant mo-bile application for teaching arabic pronunciation using a new approach for Arabic speech recognition. Journal of Theoretical and Applied Information Technology, 95(3).

Alrouqi, H., Alarifi, A., Alnafessah, A., Alhadhrami. S., Al-Khalifa, S.H, Al-Salman, A.S. & Al-Ammar, M.A. (2016). Evaluating Arabic Text-to-Speech synthesizers for mobile phones. 10th International Conference on Digital Information Management, ICDIM2015, pp. 89-94. doi: 10.1109/ICDIM.2015.7381856.

Al-Saleh, D. & Larabi-Marie-Sainte, S. (2021). Arabic Text Classification Using Convolutional Neural Network and Genetic Algorithms. *IEEE ACESS Journal*, 9, pp. 91670-91685. DOI. 10.1109/ACCESS.2021.3091376.

El-mashed, S. Y., Sharway, M. I. & Zayed, H. H. (2011) Speaker Independent Arabic Speech Recognition using Support Vector Machine. *ICI 11, Conference and Exhibition on Information and Communication Technology*, 11, pp. 401–416.

Elsayed, E. and Fathy, D. (2019). Evaluation of Quran recitation via OWL ontology based system. *International Arab Journal of Information Technology*, 16(6), pp. 970–977.

Gerhana, Y. A., Atmadja, A.R., Maylawati, D.S., Rahman, A., Nufus, K., Qodim, H., Busr, H. & Ramdhani, M.A. (2018). Computer speech recognition to text for reciting Holy Quran. *in IOP Conference Series: Materials Science and Engineering*. 434. doi: 10.1088/1757-899X/434/1/012044.

Ghadage, Y. H. and Shelke, S. D. (2016). Speech to text conversion for multilingual languages. *International Conference on Communication and Signal Processing, ICCSP 2016*, pp. 0236-0240. doi: 10.1109/ICCSP.2016.7754130.

Hamad, M. and Hussain, M. (2011). Arabic text-to-speech synthesizer. *Proceedings - 2011 IEEE Student Conference on Research and Development, SCOReD 2011*, pp. 409-414. DOI: 10.1109/SCOReD.2011.6148774.

Këpuska, V. (2017). Comparing Speech Recognition Systems (Microsoft API, Google API And CMU Sphinx). *International Journal of Engineering Research and Applications*, 2248-9622(3), pp. 20-24. DOI: 10.9790/9622-0703022024.

Khan, W. and Daud, A. and Nasir, A.J. and Amjad, T. (2016). A survey on the state-of-the-art machine learning models in the context of NLP. *Kuwait Journal of Science*. 43 (4), pp. 95-113.

Khan, K. and Ullah, A. and Baharudin, B. (2016). Pattern and semantic analysis to improve unsupervised techniques for opinion target identification. *Kuwait Journal of Science*. 43 (1), pp. 129-149.

Larabi-Marie-Sainte, S., Alalyani, N., Alotaibi, S., Ghouzali, S. and Abunadi, I. (2019). Arabic Natural Language Processing and Machine Learning-Based Systems. *IEEE Access Journal*, Vol 7, pp. 7011-7020. doi:10.1109/ACCESS.2018.2890076.

Moath M. Najeeb, Abdelkarim A. Abdelkader and Musab B. Al-Zghoul (2014). Arabic Natural Language Processing Laboratory serving Islamic Sciences. *International Journal of Advanced Computer Science and Applications*, 5(3), pp. 114–117. DOI: 10.14569/IJACSA.2014.050316.

Muhammad, A., Ul Qayyum, Z., Mirza, W.M., Tanveer, S., Martinez-Enriquez A.M., Syed, A.Z. (2012). El-hafiz: Intelligent system to help Muslims in recitation and memorization of Quran. *Life Science Journal*, 9(1), pp. 534–541.

Oumaima, Z., Abdelouafi, M. and Meryem, E. H. (2018). Text-to-Speech Technology for Arabic Language Learners. *in Colloquium in Information Science and Technology, CIST*, pp. 432-436. DOI: 10.1109/CIST.2018.8596372.

Reddy, B. R. and Mahender, E. (2013). Speech to Text Conversion using Android Platform. Interna-tional Journal of Engineering Researc and Application, 3(1), pp.253-258.

Saksamudre, K.S., Shrishrimal, P. P., and Deshmukh, R. R. (2015). A Review on Different Approaches for Speech Recognition System. International Journal of Computer Applications, 115(22), pp. 23-28. doi: 10.5120/20284-2839.

Trivedi, A., Pant, N., Shah, P., Sonik, S. and Agrawal, S. (2018). Speech to text and text to speech recognition systems - A review. *IOSR Journal of Computer Engineering (IOSR-JCE)*, 20(2), pp. 36-43. DOI: 10.9790/0661-2002013643.

Yousfi, B. and Zeki, A. M. (2016). Automatic Speech Recognition for the Holy Qur'an, A Review. *The International Conference on Data Mining, Multimedia, Image Processing and their Applications (ICDMMIPA2016).*

Submitted: 29/11/2020
Revised: 17/02/2021
Accepted: 17/02/2021
DOI: 10.48129/kjs.v49i1.11231

Copyright of Kuwait Journal of Science is the property of Kuwait University, Academic Publication Council and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.