Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2021) 29: 2298 – 2311
© TÜBİTAK
doi:10.3906/elk-2010-117

Detecting and correcting automatic speech recognition errors with a new model

Recep Sinan ARSLAN[1,*], **Necaattin BARIŞÇI**[2], **Nursal ARICI**[3], **Sabri KOÇER**[4]
[1,2,3]Department of Computer Engineering, Faculty of Technology, Gazi University, Ankara, Turkey
[4]Department of Computer Engineering, Faculty of Engineering, Necmettin ERBAKAN University, Konya, Turkey

Received: 28.10.2020 • **Accepted/Published Online:** 12.03.2021 • **Final Version:** 23.09.2021

Abstract: The purpose of automatic speech recognition (ASR) systems is to recognize speech signals obtained from people and convert them into text so that they can be processed by a computer. Although many ASR applications are versatile and widely used in the real world, they still generate relatively inaccurate results. They tend to generate spelling errors in recognized words, especially in noisy environments, in situations where the vocabulary size is increased, and at times when the input speech is of poor quality. The permanent presence of errors in ASR systems has led to the need to find alternative methods for automatic detection and correction of such errors. In this study, the basic principles of ASR evaluation are first summarized, and then a new approach based on the suggestion of an alternative hypothesis is proposed for the detection and correction of these errors generated by ASR systems. The proposed method involves a series of processes such as identifying incorrect words, selecting the ones that can be corrected, and identifying candidate words to replace these words. As a result of the tests carried out by creating different test environments, significant performance improvements for Turkish were achieved and an average of 4.60 % performance improvement was provided.

Key words: Automatic speech recognition, automatic speech recognition error correction, artificial ıntelligence, alternative hypothesis suggestion, natural language processing

1. Introduction

Along with the progress of information technologies, computers are no longer used just for mathematical and scientific transactions. Instead, it is possible to develop and run applications on computers, which could generate solutions to multidimensional problems in different areas. There has been a vast amount of interest in automatic speech recognition (ASR) systems in recent years, and a significant number of studies have been carried out by various scientists, universities, and research centers [1]. ASR is the process of mathematically representing the acoustic signal and converting it into words, which can be processed by computers. Many different applications such as voice-to-text conversion, automated telephone services, voiced user interface applications, and voice-oriented home automation systems are some examples of ASR systems. The basic flow chart shown in Figure 1 is followed in the modeling of speech recognition systems. Accordingly, after the input signal is received through the microphone, the process of extracting, digitizing, filtering various samples, labeling, and converting them into a format that can be modeled is performed in the preprocessing stage. The objective in this step is to achieve a simpler, noise-free, and easy-to-operate speech variation. In the next feature extraction step, the parameters of the resulting audio signal are extracted, and the required calculations are performed to obtain the properties of the audio at certain time intervals. The parameters obtained in this step are critical values for speech recognition systems and provide important clues for speech recognition. Speech recognition is performed

*Correspondence: sinanarslanemail@gmail.com using the parameters obtained in this step. Correction is required in the outputs, due to the fact that 100 % correct results cannot be achieved in the speech outputs obtained after this process. The most basic structure used for this is the language model [2]. Improvement is made on the results using these preformed structures, and the final results are thus obtained [3].

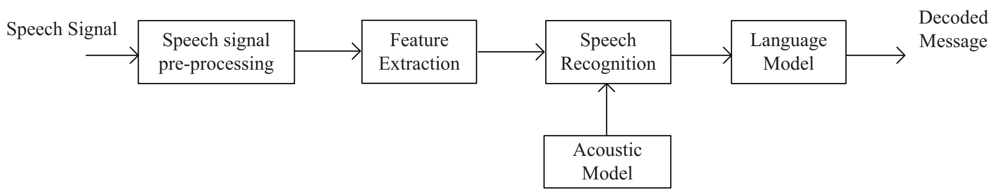

Figure 1. Block diagram of a speech recognition system [4].

Despite the various highly advantageous benefits of ASR systems, they are still considered to be defective structures due to typographical errors in the output texts. The primary errors in these systems are linguistic errors and misspellings. This is usually caused by noise in the environment, poor sound signals, overlapping sounds due to reciprocal conversations and vocabulary depending on the language [5, 6]. Numerous error correction methods and algorithms have been designed to help correct errors in ASR systems [7, 8]. While some of these process the text afterward or manually correct the text, others try to create more advanced acoustic models [9]. Although all these studies are aimed at reducing error rates, they are still not convincing enough, and speech recognition systems still do not generate results with 100 % accuracy [10]. In this study, a new method based on the suggestion of an alternative hypothesis is proposed to determine whether the output text generated for ASR systems contains any incorrect words and correct them, if any. Basically, it includes the comparison of incorrect words to Turkish reference words and correction of incorrect words. It is an effective approach in eliminating typographic errors. The proposed approach provides a significant reduction in error rates in testing with acoustic sets used in the real world, and as a result, it is an effective method for improving system performance.

2. Turkish language structure and the problem of out-of-vocabulary words

The Turkish language belongs to Oghuz the group of Ural – Altaic language family. It is spoken in a geographical area covering Turkey, Cyprus, Iraq, the Balkans, the Middle East, and some of the European countries. It is an agglutinative language [11]. With the development of today's speech, communication and voice processing technologies, it is expected to develop robust speech recognition structures based on the word structure, acoustic parameters, and language structure of the Turkish language. The development of the model in speech recognition systems is a very laborious process and requires the resolution of problems specific to each language [12]. Since the Turkish language is an agglutinative language, the number of words that can be derived is enormous compared to other languages, which is a feature of the agglutinative languages. Hundreds of thousands of different morphological variants can be generated from a verb stem. Many of these variants have different meanings, as well [13]. The same problems exist for many languages, such as Czech, Finnish, Hungarian, and some Asian languages. This is a head-aching problem for speech recognition systems. Increased vocabulary requires collaboration with larger word sets and audio sources. For a model with better performance, fewer nonvocabulary words are required [14]. This phenomenon makes it difficult to develop speech recognition systems in the Turkish language.

3. Automatic speech recognition error correction algorithms

Different error correction approaches have been proposed to detect and correct incorrect words in the outputs generated by ASR systems. These approaches can be generally divided into the categories of manual error correction, alternative hypothesis-based error correction, pattern learning, or post-output error correction [10]. In the manual error correction algorithm, the words generated by the system are reviewed and manual correction process is assigned to a person. Besides the probability of overlooking some errors, this process is very laborious and time-consuming [10]. Another error correction method is the method of correcting an error by replacing it with an alternative word. The main disadvantage of this method is that the alternative word is derived from a vocabulary dictionary and it therefore requires a vast amount of vocabulary sources. In many different studies, different algorithmic approaches have been proposed for this category [15, 16]. Pattern learning is another error correction approach in which patterns considered to be erroneous are spotted and errors are detected. The system is first trained using a set of erroneous words from a particular domain. Next, error rules that can be detected after system errors occur are created. During the recognition process, the ASR system detects errors by verifying the output text according to predefined rules. The disadvantage of this approach is that it is subject-specific, and the number of words that can be recognized by the system is quite limited [10]. The final error correction algorithm is the method of correcting the output after the output is generated. In this approach, an extra layer is added to the ASR system to detect and correct spelling errors in the final output text after recognition. The advantage of this technique is that the recognition and correction processes of the ASR system can be combined and thus easily integrated into the general recognition system. In this approach, NLP can be evaluated together with different fields such as machine translation [10].

4. Methods

4.1. Deep neural networks

Deep learning is a special case of machine learning that incorporates artificial intelligence. The first neural networks are perception algorithms [17]. This network consists of an input and output layer. It is possible to model more complex relationships by adding multiple layers to neural networks. Structures that contain more than one layer in such manner are called deep neural networks. It defines a multilayered and multineuronal neural network. With the development of faster GPU systems and their significant increases in performance, the use of deep neural networks has become more widespread [18, 19].

4.2. Recurrent neural network (RNN)

In neural network structures, people's real-life learning systematic is followed. People use their knowledge to acquire more knowledge. There is a structure of thinking where given information is superimposed. In conventional neural networks, this cumulative learning pattern cannot be put into practice. This drawback is the most important reason why conventional neural networks cannot be used especially in solving problems such as speech recognition. For example, in speech recognition applications, the previous voice is very important in recognizing the next voice. Evaluating each sound independently would result in erroneous results. Since such dependence cannot be modeled in conventional neural networks, recurrent neural networks (RNNs) have been proposed to address this problem [20].

4.3. Long short term memory (LSTM)

It is a type of recurrent neural network that can model long-term dependencies [21]. Its purpose is to keep information in memory for a long time. It is the duty of the neural network to decide what information is to be learned and to do the network training. The use of this memory structure is increasing each day, and better results are obtained for many problems [22]. The LSTM structure is chain-shaped and has four gates instead of a single layer, as shown on the left side of Figure 2. Cell state is the channel through which information is transmitted continuously from one cell to another. Given that LSTM cells are sequenced one after the other, the flow of information between the cells is thus achieved in such a manner. In the LSTM structure, the decision on which information is to be forgotten and which information is to be transferred to a next cell is made using the forget gate. The value to be forgotten is decided upon by applying a Sigmoid Function on the new information that arrives as an input together with the information coming from the previous cells [23]. Using the input gate, it is decided which information is stored with cell states and which is transmitted to other cells. After completing the operations at this stage, the cell states are also updated and prepared to be transferred to the next cell. The output gate determines the output value of the cell. However, not all values coming from the cell state are used as outputs. This filtering process, too, is performed using the "tanh" function [24].

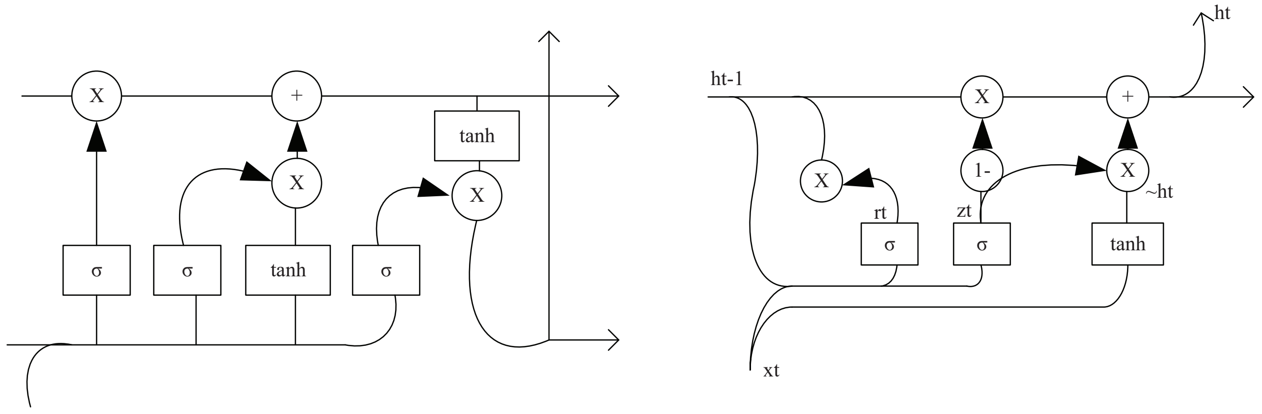

Figure 2. Schema of long short term memory (left) and gated recurrent unit (right) [25].

4.4. Gated recurrent unit (GRU)

GRU is the type of LSTM that is most widely used. The most significant feature that distinguishes this structure from the LSTM structure is the elimination of deletion of cell state. The second most significant distinctive feature, however, is the merging of two gates, the forget and the update gates, into one. With these two changes, a simpler structure is obtained compared to LSTM, as shown on the right in Figure 2 [25].

4.5. Levenshtein algorithm

In computer science, the Levenshtein algorithm is a method used to measure the distance between two sequences. It refers to the minimum number of single character edits required to replace one word with another. It was proposed in 1965 [26]. Where the function "lev(a, b)" is used to calculate the distance between the text strings a and b,

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) \quad if min(i,j) = 0 \\ min = \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j)+i_{a_i != b_i} \end{cases} \end{cases}$$

When "$a_i = b_i$", the display function is taken as 0, in other cases it is taken as 1. With the "lev" function the distance between the section of the string "a" up to its index "i" and the section of the string "b" up to its index "j" is calculated. The values "i and j" represent values that increase by 1. Using the Levenshtein algorithm, the distance between the two strings is calculated. If it is later requested to be used by being normalized to 0.0–1.0 interval, then the "normalized Levenshtein" algorithm should be used. In this case, however, the results are no longer a metric. The similarity is used as normalized to 1 [26].

4.6. System performance evaluation criteria

There are several approaches used to measure the improvement in system performance for voice recognition applications. One of these measurement methods is to calculate the word recognition rate as per the formula given below (Eq-2).

$$Wordrecognitionratio(WRR) = (N - S - D)/N * 100$$

In the equation, N is the total number of words, S is the number of misspelled words that need any character addition and D is the number of misspelled words that need any character deletion. Accordingly, when these value are subtracted from the total number of words, the number of correctly recognized words is found. In the equation, the recognition performance of the system is calculated as a percentage. The second method that can be used to measure the success of the performance improvement method proposed in this study is to calculate the above process for each test and numerically proportion it for each group of tests. For example, in each step for 10000 tests, it is the calculation of the results according to the method given above and the comparison of the traditional model with the proposed model output and numerically measuring which is more successful. This result demonstrates whether or not the proposed method generates a model that has a better performance in all cases.

5. Materials

In order to test the proposed model in this study, acoustic and text sets are needed. Therefore, acoustic and text sets were prepared for use. For the acoustic model, Metu 1.0 acoustic dataset was used, which has already been used in many studies. It includes a significant amount of speech records to be used in the Turkish speech recognition systems, and it is distributed over the Linguistic Data Consortium. It contains an average of 500 minutes of speech, with 60 male and 60 female speakers voicing 40 sentences, each of which contains 300 words on average. It contains randomly selected 2462 Turkish words. It is a systematically prepared acoustic dataset [27, 28]. In order to correct the recognition outputs of the traditional model, a text dataset containing different words is needed to form the Turkish reference words database. It is required to contain all Turkish words where possible. Otherwise, a match and required corrections to the results cannot be made. For this purpose, three commonly used Turkish word groups were preferred in this study. These were Zemberek [29], Boun Corpus [30] and Metu 1.0 [27, 28] datasets. Zemberek is an open source library published in 2010. It contains Turkish grammar features. It contains approximately 1.15 million unique Turkish words. Boun Corpus, published in 2008, is a study carried out to create a Turkish language resource. It contains approximately 1.4 million unique Turkish words. The Metu v1.0 dataset was published in 2002. It contains around seven thousand unique

Turkish words. These three text sets were combined to create a database of approximately 1.6 million words for reference.

6. The proposed model, experiments, and results

For a Turkish speech recognition system using LSTM, GRU, or RNN, a model was proposed for performance improvement. Thus, it was aimed to reduce the system error rate and to develop a speech recognition model with better output.

6.1. Model suggestion

The flow diagram of the proposed model is shown in detail in Figure 3. Accordingly, firstly, a conventional speech recognition model was developed and its outputs were taken. Whether the model is based on LSTM, GRU, or RNN does not cause any change in the application method. After the output of the model was taken, it was included into the error correction mechanism within the flow given below instead of using a language model used in conventional methods. Each word in the output was compared with the words in the previously created database, and it was aimed to find the word closest to that word. At this point, however, a few things must be checked in this comparison process. The most important of these is whether a word that is close enough to the model output is found in the database. When the database does not contain a close-enough word, the erroneous system output is replaced with a more erroneous one, which, in turn increases the error rate of the system. Therefore, if a word with a closeness level below a certain threshold can be found in the database, the second control stage is initiated. Otherwise, no correction is made in the model output. The second control stage is to determine whether the model output is able to generate a sufficiently long word. Because, in some tests, text outputs that are long enough cannot be generated in the process of converting to text. In this case, the estimation of words containing a single letter is not possible. Therefore, it is necessary to check whether the word is long enough. After these two control processes, the model output is corrected. Thus, significant reductions are provided in the system error rate.

Experimental sets were established to find the best values (threshold values) for the two control stages, which should be decided upon for the above model and the results are given in detail in this section. Thus, a better improvement in system performance was obtained using a better threshold value. In the proposed model, it was aimed to recommend the correct words instead of the erroneous ones, thus increase the performance of the system. The pseudo-code version of the method is shown in Algorithm-1. Accordingly, all variables must first be reset and prepared for use in the algorithm. Then, an automatic speech recognition system output based on RNN, LSTM, or GRU should be obtained. It is assumed that these outputs contain erroneous words. Then all words in the database are compared with incorrect system outputs, and their distances are calculated. If there is a word with a distance of 0, the process is terminated. Otherwise, it continues to investigate whether there is a threshold value or not. If it is found that the word is replaced with the reference word in the database. Except for these two cases, no correction is made in the wrong output. Thus, correctable ones are selected from faulty outputs, and it is aimed to contribute to the system performance

6.2. Determination of the threshold value when calculating the closeness of a word

In the process of calculating the distance between two words, it was assessed which text distance calculation algorithm could be used to get better results. In the studies carried out in this field [31] it was confirmed that the ”normalized Levenshtein (NL)” algorithm performed more successful computations. Therefore, in this

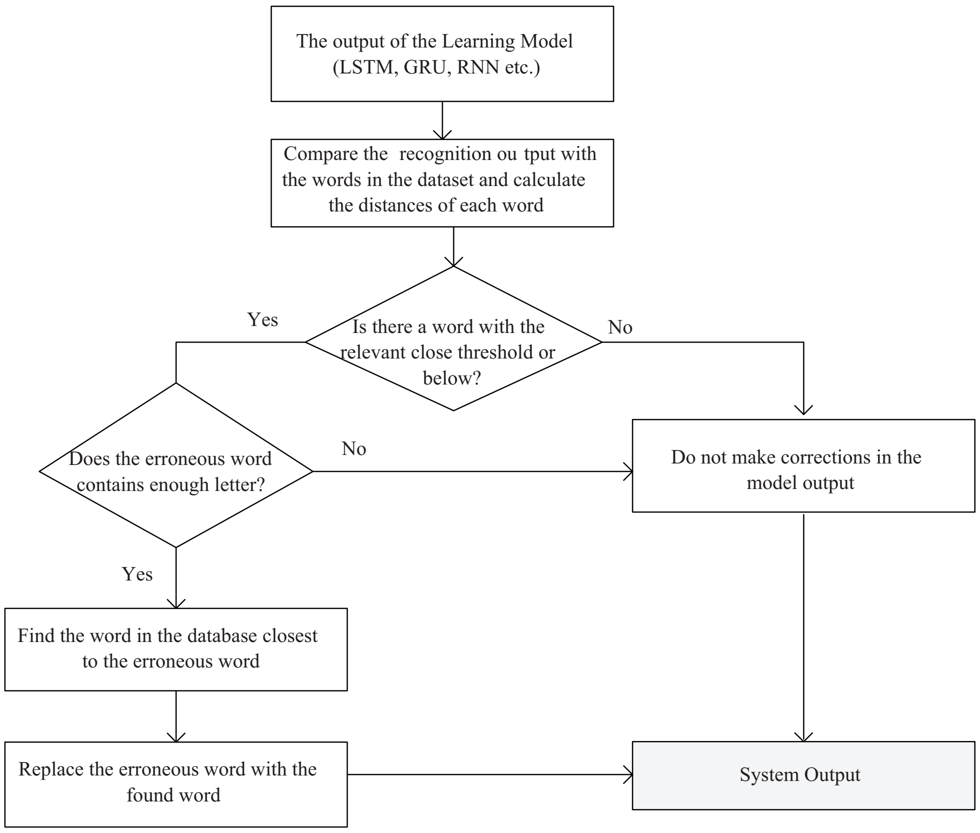
The output of the Learning Model (LSTM, GRU, RNN etc.)
Compare the recognition ou tput with the words in the dataset and calculate the distances of each word
Is there a word with the relevant close threshold or below?
Yes
No
Does the erroneous word contains enough letter?
No
Yes
Do not make corrections in the model output
Find the word in the database closest to the erroneous word
Replace the erroneous word with the found word
System Output

Figure 3. Proposed ASR error correction method.

study, this algorithm was chosen as the distance calculation algorithm. In the NL algorithm, the distance between two texts is in the range of 0–1. In this case, the threshold value to be determined must be within this range. In the process of comparing the words in the word database with the system output, it was confirmed that there were some problems in finding the word close to the erroneous word. The main problem is that the model contribution level drops with the replacement of an erroneous word in cases where a close-enough word is not found. For this reason, it was confirmed that a specific threshold value should be determined, and the replacement should be made if a word in a distance under this threshold value is reached in the database, otherwise, it would be useful to leave the output as erroneous. Figure 4 demonstrates the effect of changing the threshold value on the contribution level to the overall system performance of the proposed model.

The first graph below shows, the change in the contribution to the general system performance when the threshold value was changed. When the graph is examined, it is understood that the level of system performance is increased by between 2 % and 4.60 % if the threshold value is changed between 0 and 1. In any case, the erroneous characters are corrected at a certain level, but the proportional effect differs depending on the threshold value. The second graph shows that what percentage of the words used in the tests should be corrected. The results show a similar trend to first graph. Accordingly, when the two graphs were evaluated together, it was determined that the proposed model reached the highest error correction level when the threshold value was determined as ”0.33”. Thus, it was possible to find and correct more errors in the proposed model. As a result of this comparison, different test sets were arranged based on the threshold value determined as ”0.33” and the results were evaluated and given in Table 1.

Algorithm 1 Error correction algorithm.

```
1:  procedure ErrorCorrection
2:    Reset all variables
3:    Get outputs from LSTM, GRU or RNN ASR system with some erroneous outputs
4:    Format all outputs
5:
6:    While the current position is inside the db
7:        Calculate the distance with the word in the db and system output
8:        if the distance = 0
9:            Return the word in the db
10:           Break;
11:       Else if distance (lt) threshold value && length of system output (gt) word length threshold
12:           Change the system output with a word in the db
13:       Else
14:           Don't change the system output
15:   end
16:
17:   Return the system output
```

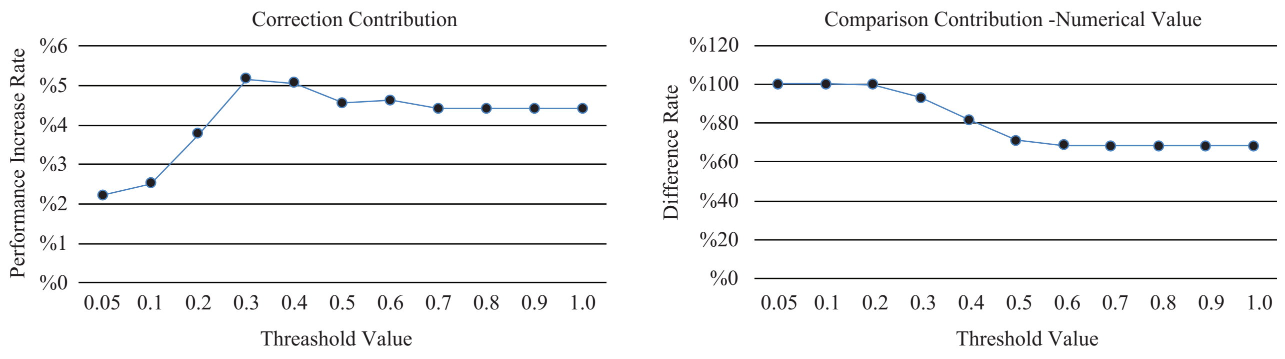

Figure 4. Threshold change graph in the optimization of the model proposed.

The performance contribution of the tests performed with five different sets was 3.01 % on average and increased to 4.60 % with the use of a threshold of value 0.33. The key point here is that the use of the threshold value in all tests makes an extra contribution to performance improvements. In addition, when the difference in the numerical value was considered, the average difference value of 55.42 % increased significantly to 90.24 %. This shows that nine out of every 10 tests performed according to the standard results of the proposed model generated better results. In that sense, the use of a threshold value is highly critical and directly affects the results. Similarly, in the experiments conducted with the test sets in the GRU architecture, the performance improvement increases from 3.13 % to 4.55 % and the numerical contribution level from 57.82 % to 84.62 %. The aim of testing by establishing two different models is to show that the proposed error correction algorithm can increase the success of different models. This is mainly due to the fact that the Turkish language is an agglutinating language and because there are no databases containing enough readily available words. The best way to solve this situation is to leave the erroneous word unchanged instead of replacing it with the wrong word if a word close to the erroneous word is not available in the database. This condition is shown in Figure 5 for a sample Turkish word. The word expected to be “kavuniçi [orange color]” was generated as “kvcni” in the standard model. Closest to this word was found in the database as ”avci [hunter]”. In this case, the error rate

Table 1. Test results for performance improvement in threshold usage.

Test set	Model	Mean performance improvement in the standard model	Standard model + proposed model with the threshold value = 0.33	Numerical value difference in the standard model	Numerical value difference in the standard model + proposed model with the threshold value = 0.33
Set1	LSTM	4.27%	5.18%	64.90%	91.64%
Set2	LSTM	1.85%	4.38%	38.58%	87.60%
Set3	LSTM	3.96%	4.64%	81.62%	93.04%
Set4	LSTM	3.30%	4.52%	61.12%	90.96%
Set5	LSTM	1.70%	4.30%	30.88%	87.96%
Mean	LSTM	3.01%	4.60%	55.42%	90.24%
Set1	GRU	3.37%	4.26%	59.83%	75.26%
Set2	GRU	3.85%	6.16%	45.28%	88.84%
Set3	GRU	4.01%	4.25%	62.57%	87.50%
Set4	GRU	2.27%	3.76%	80.79%	88.42%
Set5	GRU	2.18%	4.32%	40.63%	83.06%
Mean	GRU	3.13%	4.55%	57.82%	84.62%

of 0.5 increases to 0.6. The best way to prevent this is to enrich the word set, and the performance level can be increased with the use of threshold values.

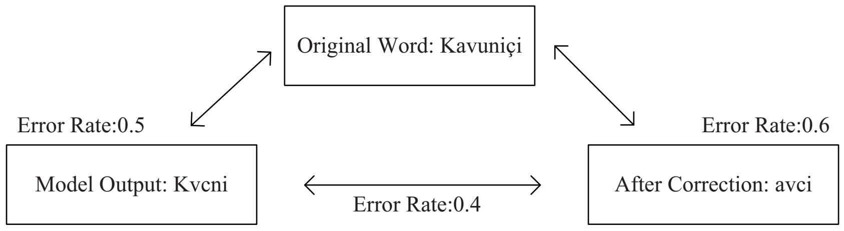

Figure 5. Incorrect output correction problem of the proposed model.

6.3. The effect of word length on recognition performance

As the Turkish language is one of the agglutinating languages, the number of words that can be derived from one stem is quite high. For this reason, there is quite a large number of the same words and derivatives in the database. In this problem, it increases the number of template words with a particular number of characters and below. This increase makes it impossible to guess the correct word. The results of the experiments performed to test this theory are shown in the graphs given in Figure 6. The variability of the correction level with respect to the incorrect word length generated by this test is intended to be shown here comparatively.

When the two graphs given above are evaluated, the average performance increases by up to 5.27 % in cases where three or more characters are corrected. Similarly, the numerical contribution level generates the best results with a similar character length. In cases where erroneous outputs with three or fewer characters are corrected, the contribution level decreases below 5 %. Therefore, comparing the words that are not long enough with the words in the database reduces the contribution level of the model. The proposed model's level of correcting incorrect words varies depending on the overall recognition performance. At this point, the change in the effect level according to system performance was tested, and the results are given in Table 2.

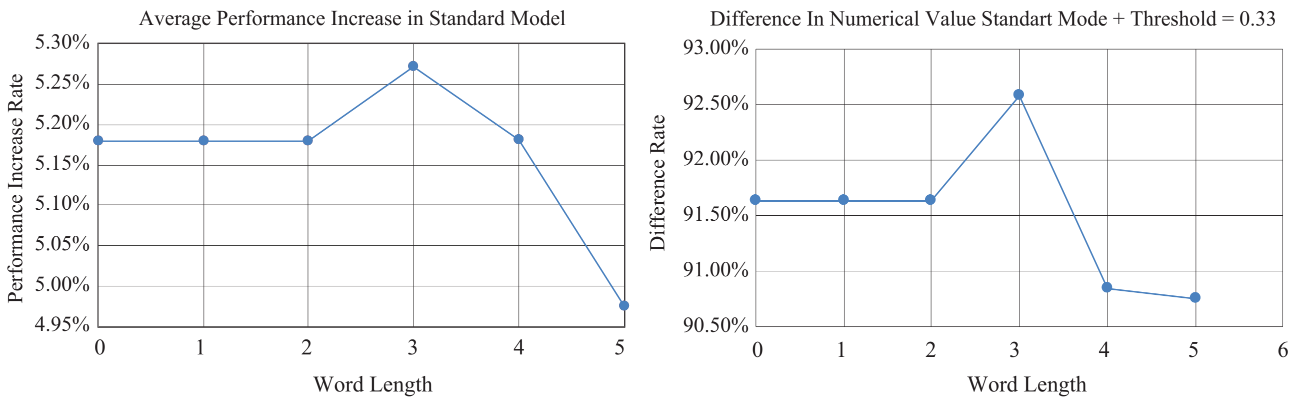

Figure 6. The effect of changing the length of a word on the model's contribution level.

Table 2. The effect of system error rate on the contribution of the proposed model.

Error rate of system	Standard model error rate	Standard model + proposed correction method	Standard model average performance improvement	Result of numerical comparison in standard model	Result of numerical comparison standard model + correction method	Result of numerical comparison standard model + correction method proportional
<0.025	0.01815	0	1.82%	0	43	100.00%
<0.05	0.03508	0.00175328	3.33%	0	236	100.00%
<0.1	0.06652	0.0148519	5.17%	3	900	99.34%
<0.2	0.126016	0.068501	5.75%	17	2888	98.83%
<0.3	0.186109	0.127689	5.84%	81	5745	97.22%
<0.4	0.224811	0.166771	5.80%	181	7584	95.34%
<0.5	0.243172	0.184974	5.82%	238	8257	94.40%
<0.6	0.260213	0.203749	5.65%	319	8689	92.92%
<0.9	0.30344	0.250572	5.29%	371	9584	9.55%

When Table 2 is examined, it is understood that the algorithm for the correction of incorrect words contributes more to the systems with better performance. In cases where the number of incorrect words is high and overall recognition performance decreases, correction cannot be performed sufficiently, which affects the contribution level of the proposed model. In systems with an accuracy greater than 98.2 %, all errors are corrected and system recognition performance is increased to 100 %. Similarly, the recognition level of a 96.5 % system can be increased to 99.9 %. The most successful results are obtained in speech recognition systems with an accuracy of 80 % or more and the model provides an improvement by up to 5.84 %.

6.4. Correction algorithm and variability of post-optimization processing time

The proposed model requires extra processing time as it performs the detection and correction of incorrect words after conventional speech recognition systems. The level of this need is critical. If the time is too long, it will not be possible to use the model in many real-time systems. For this reason, how much extra time this method may need was evaluated and tests were performed with the sample sets that had been established. The results are given separately for each test set in Table 3.

Table 3. The effect of correction algorithm and optimization processes on the model's processing time.

Test set	Database reading time (ms)	Time needed for correction (ms)	Total number of words in test sets
Set1	228	102644	60853
Set2	218	109556	66016
Set3	288	97086	62839
Set4	396	83097	60632
Set5	235	92522	55382
Mean	273	96981	61144

When the Table 3 is examined, an average of 273 ms is needed in order to read 1.6 million words that are already registered in the database. If the number of different words is increased for languages that do not have sufficient data sources such as Turkish, this duration may increase but this will come with the contribution the proposed model offers. The slight changes in database reading time are due to the effect of other processes running at the time of the test. The second important process extending the general recognition time is the time required by the incorrect word correction method proposed in this study. There are 61144 words in five different test sets on average and 96981 ms of time is needed to correct these words. In this case, 1.50 ms on average is needed for each word. This indicates that an average of 0.2 s is needed for 1 min of speech. This is a very acceptable duration and can be used in real-time automated speech recognition systems. These results were obtained on an end-user computer with i7 2.0 GHz CPU and 8 GB of RAM; therefore, this model would accomplish these processes in a shorter time in more powerful machines.

7. Discussion

It is observed, nowadays, that speech recognition systems cannot generate outputs with 100 % accuracy. Their performance is affected by conditions such as noisy environments. Therefore, many different approaches have been proposed to correcting the outputs of automatic speech recognition systems. While each method has its own advantages and disadvantages, each contributes to the general recognition performances at varying rates. Table 4 gives some of the studies carried out in this field. As can be seen in the table, an average of 2.25-10.78 % performance improvement was achieved in experiments conducted with different languages and datasets. In this study, too, a specific model that was similar to the sample models was proposed, and the results obtained in different testing environments were evaluated. In the study carried out on the Turkish language, a performance improvement by 5.34 % on average was achieved, similar to the studies in the table.

8. Conclusion and future work

In this study, a unique error correction method is presented in order to detect and correct errors in speech recognition applications. The proposed model is based on a reference database and a correction algorithm with the necessary optimization processes completed. The reference database was organized to include verbs and nouns of the Turkish language, specific names, field-specific terms, technical terminologies, scientific

Table 4. Recommendation on similar error correction algorithms and contribution levels.

Publication info	Method used/proposed	Speech type	Dataset	Average performance improvement
Yohei et al. [32]	Ngram + Normalized relevance distance	Continuous speech recognition	Dataset: CSJ Speech database, MFCC, 2596 lecture video, 520 thousand words, Japanese	38.82% ->28.04% 10.78% performance increase.
Yusuke Nakashima et al. [33]	Proposed new model	Continuous speech recognition	Newspaper and web data, Japanese	8.6% ->4.3% for newspaper 4.3% performance increase. 9.8% - 6.5% for web data 3.3% performance increase.
Dong Yu et al. [34]	Adapt the lexicon, adjusting LM, adding new words, learning new pronunciation	Continuous speech recognition	Microsoft speech data, English	11% performance increase if word average WER is upper than 10%.
Yongmei Shi and Lina Zhou [35]	Noisy context, accurate context	Continuous speech recognition	32 sentences from English dictation corpus (sears et al. 2003, Feng and Sears 2004), 71800 words, English	%63 ->58% for accurate context 5% performance increase. 61% ->53% for noisy context 8% performance increase.
Bassil Youssef and Paul Semaan [10]	N-gram + proposed new model	Continuous speech recognition	Different English articles, 100 word by 5 different speakers. Microsoft N-gram dataset, English	21% ->14% 7% performance increase
Arslan, R.S. and Barışçı, N. [31]	Proposed new model	Continuous speech recognition	Metu 1.0 Dataset, Zemberek, Boun corpus, Turkish	2.25% performance increase.
This study	Alternative hypotheses based new proposed model	Continuous speech recognition	Metu 1.0 Dataset, Zemberek, Boun corpus, Turkish	4.60% performance increase.

abbreviations, and special expressions. As a result of the tests conducted with acoustic test sets and the LSTM, and GRU based sample model, a significant amount of decrease was achieved in the error rates of automatic speech recognition systems. In systems using the proposed model, an average performance improvement by 4.60% was achieved. The model is able to contribute more to the speech recognition applications having better system performances. In order to shorten the run time of the model in the future, it is very important to make the processing times simultaneous. It will also be useful to enrich the reference word set to increase its contribution to the system recognition performance. With the resolution of the resource problems in languages such as Turkish and wide spreading of the word databases with larger scales, it will be possible to make more error corrections and to develop automatic speech recognition systems having better performance levels.

References

[1] Dutoit T. An introduction to text-to-speech synthesis. Berlin, Germany: Springer Science and Business Media, 2001.

[2] Sak H, Saraçlar M, Güngör T. Morpholexical and discriminative language models for Turkish automatic speech recognition. IEEE Transactions on Audio, Speech and Language Processing 2012; 20 (8): 1-11.

[3] Anusuya MA, Katti SK. Speech recognition by machine: a review. International Journal of Computer Science and Information Security 2009; 4 (3): 181-205.

[4] Kandarpa KS, Mousmita S. Acoustic modelling of speech signal using artificial neural network: a review of techniques and current trends. Intelligent Applications for Heterogeneous System Modeling and Design 2015; 1 (12): 287-303.

[5] Deng L, Huang X. Challenges in adopting speech recognition. Communications of the ACM 2004; 47 (1): 60-75.

[6] Forsberg M. (2003). Why speech recognition is difficult [online]. Website http://www.speech.kth.se/ [accessed 01 Jan 2021]

[7] Jeong M, Lee GG. Improving speech recognition and understanding using error-corrective reranking. ACM Transactions on Asian Language Information Processing (TALLIP) 2008; 7 (1): 1-16.

[8] Jeena HP, Golda BR, Hema AM. Importance of signal processing cues in transcription correction for low-resource Indian languages. ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP) 2020; 19 (1): 1-26.

[9] Rahhal E, Asmaa H, Hassan O. Automatic speech recognition errors detection and correction: a review. Procedia Computer Science 2018; 128: 32-37.

[10] Youssef B, Semaan P. ASR context-sensitive error correction based on Microsoft n-gram dataset. Journal of Computing 2012; 4 (1): 1-9.

[11] The formation of Turkish and the place of Turkish among world languages (2020). Home Page [online]. Website http://www.turkcede.org/turk-dili/729-turkcenin-olusumu-ve-turkcenin-dunya-dilleri-arasindaki-yeri.html [accessed 01 Jan 2021].

[12] Aksoylar C, Mutluergil SO, Erdoğan H. The anatomy of a Turkish speech recognition system. In: IEEE Signal Processing and Communications Applications Conference (SIU); Antalya, Turkey; 2009. pp. 512-515.

[13] Özbey C, Bayar S. Automatic speech recognition: generating and testing generic acoustic model for Turkish. In: 19. Academic Conference on Informatics; Aksaray, Turkey; 2017. pp. 1-6.

[14] Asefisaray B, Mengüşoğlu E, Hacıömeroğlu M, Sever H. How does language model size effects speech recognition accuracy for the Turkish language?. Pamukkale University Journal of Engineering Sciences 2016; 22 (2): 100-105.

[15] Setlur AR, Sukkar RA, Jacob J. Correcting recognition errors via discriminative utterance verification. In: Proceedings of the International Conference on Spoken Language Processing; Philadelphia, USA; 1996. pp. 602-605.

[16] Zhou Z, Meng HM, Lo WK. A multi-pass error detection and correction framework for Mandarin LVCSR. In: Proceedings of the International Conference on Spoken Language Processing; Pittsburgh, USA; 2006. pp. 1646-1649.

[17] Rosenblatt FF. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review 1958; 65 (6): 386-409.

[18] Socher R, Bengio Y, Manning C. Deep learning for NLP (without magic). In: ACL Annual Meeting of the Association for Computational Linguistics; Jeju Island, Korea; 2013. pp. 5-25.

[19] Arslan RS, Barışçı N. The effect of different optimization techniques on end-to-end Turkish speech recognition systems that use connectionist temporal classification. In: 2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT); Ankara, Turkey; 2018. pp. 1-6.

[20] Asefisaray B. End-to-end speech recognition model: tests in Turkish language. PhD, Hacettepe University, Ankara, Turkey, 2018.

[21] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computing 1997; 9 (8): 1735-1780.

[22] Schmidhuber J, Greff K, Srivasava RK, Kutnik J, Steunebrink BR. LSTM: a search space odyssey. IEEE Transactions on Neural Networks and Learning Systems 2017; 28 (10): 2222-2232.

[23] Sundermeyer M, Schlüter R, Ney H. LSTM neural networks for language modeling. In: Conference of the International Speech Communication Association (INTERSPEECH); Portland, USA; 2012. pp. 1-4.

[24] Sak H, Senior A, Beaufays F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: International Speech Communication Association; Singapore; 2014. pp. 1-5.

[25] Aydoğan M, Karci A. Improving the accuracy using pre-trained word embedding on deep neural networks for Turkish text classification. Physica A: Statistical Mechanics and its Applications 2020; 541: 1-17.

[26] Levenshtein VI. Binary codes capable of correcting deletions, insertions, and reversals. Cybernetics and Control Theory 1996; 10 (8): 707-710.

[27] Salor Ö, Pellom BL, Çiloğlu T, Demirekler M. Turkish speech corpora and recognition tools developed by porting sonic: towards multilingual speech recognition. Computer Speech and Language 2007; 21 (4): 580-593.

[28] Salor Ö, Pellom B, Çiloğlu T, Hacıoğlu K, Demirekler M. On developing new text and audio corpora and speech recognition tools for the Turkish language. In: International Conference on Spoken Language Processing (ICSLP); Denver, USA; 2002. pp. 1-5.

[29] Akın AA, Akın MD. Zemberek, an open source NLP framework for Turkish languages. Structure 2007; 10: 1-5.

[30] Sak H, Güngör T, Saraçlar M. Turkish language resources: morphological parser, morphological disambiguator and web corpus. In: 6th International Conference on Advances in Natural Language Processing; Gothenburg, Sweden; 2008. pp. 417-427.

[31] Arslan RS, Barışçı N. Development of output correction methodology for long short term memory-based speech recognition. Sustainability 2019; 11 (15): 4250-4266.

[32] Yohei F, Katsuyuki T, Tetsuya T, Yasua A. Word-error correction of continuous speech recognition based on normalized relevance distance. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence; Buenos Aires, Argentina; 2015. pp. 1-6.

[33] Yusuke N, Zhipeng Z, Nobuhiko N. Efficient speech-recognition error correction for more usable speech-to-text input. Ntt Docomo Technical Journal 2011; 11 (2): 1-8.

[34] Dong Y, Mei-Yuh H, Mau P, Alex A, Deng L. Unsupervised learning from users error correction in speech dictation. In: International Conference on Spoken Language Processing; Jeju Island, Korea; 2004. pp. 1969-1972.

[35] Yongmei S, Zhaou L. Supporting dictation speech recognition error correction: the impact of external information. Behaviour and Information Technology 2009; 30 (6): 761-774.