sustainability

MDPI

Article

Development of Output Correction Methodology for Long Short Term Memory-Based Speech Recognition

Recep Sinan Arslan [1,*] and Necaattin Barışçı [2]

[1] Department of Computer Programming, Vocational School of Technical Science, Yozgat Bozok University, 66200 Yozgat, Turkey
[2] Department of Computer Engineering, Faculty of Technology, Gazi University, 06560 Ankara, Turkey
* Correspondence: sinanarslanemail@gmail.com

Received: 17 June 2019; Accepted: 5 August 2019; Published: 6 August 2019

check for updates

Abstract: This paper presents a correction methodology for Long Short Term Memory (LSTM) based speech recognition. A strategy that validates with a reference database was developed for LSTM. It is conceptually simple but requires a large keyword database to match test templates. The correction method is based on the "most matching method" that is finding the word in which the system output is closest among the "Referenced Template Database". Each LSTM model recognition output was corrected with the proposed new concept. Thus, system recognition performance was improved by correcting faulty outputs. The effectiveness, efficiency, and contribution of this approach to system performance were demonstrated by experiments. Tests carried out using different speech-text datasets and LSTM models yielded an average performance increase of 2.25%. With some advanced models, this ratio rises to 3.84%.

Keywords: speech recognition; Long Short Term Memory (LSTM); speech output correction; most-matching

1. Introduction

Speech is the most effective means of communication among people and is the most natural way of exchanging information. Therefore, the desire of people to interact vocally with computers is increasing day by day. To meet this desire, studies are being conducted in a number of fields intended for the simulation of humans' ability to talk, from carrying out of simple tasks by computers through machine-human interaction, to turning speech to text through Automatic Speech Recognition (ASR) systems [1]. In recent years, signal processing and recognition have been utilized in many areas such as human activity tracking and detection [2,3], computer engineering [4], physical sciences, health-related applications [5], and natural science and industrial [6]. Speech sounds may get mixed with another speaker's speech in the background, in a room with TV sound or with an external voice. All sounds except the speech signal are called noise. Therefore, it is necessary to filter this noise in speech recognition systems [7]. In order to reduce noise in the recording stage of speech, the environment and the sensors are very important. Many different sensor-based technologies have been proposed for signal capturing and processing [8–11].

Along with the developments in speech modelling, ASR systems have begun to be used in many different areas such as voice processing at call centers, tasks requiring human-machine interface, travel information, stock-exchange transactions, quotations, weather reports, data entry, speech dictation, and access to information. Although there has been considerable progress in ASR systems over the last 60 years, there are still many problems that need to be solved and many particulars to be improved [12]. In the modelling of speech recognition systems, the flow-chart given in Figure 1 is generally followed. Accordingly, after the input signal is received into the system via the microphone, extracting the samples of the sound, digitizing, putting through various filters, in some necessary cases, labelling and transforming into a format that can be modelled are all carried out at pre-processing stage. Here, the aim is to produce a state of sound that is less simple, and free of speech variations and noise. In the next step, the parameters of the sampled audio signal obtained are captured and calculations are made to extract the remaining properties of the sound at certain time intervals. Feature extraction operations are used in many applications such as Real-time feature extraction of large size satellite images [13], activity tracking applications [14], face recognition [15], real-time motion capture [16], as well as human detection and activity tracking [17]. Therefore, the data obtained in feature extraction are critical values for speech recognition and provide important clues. In the next step, the comparison of the estimated word from the parameters with the language models is performed. The outputs produced after the acoustic model is compared with the outputs of the language model and a search is performed on the language model. Thus, it is aimed to find the right word to correct the acoustic model outputs. A kind of forecasting process is operated. Therefore, the correct word is obtained by making the most appropriate selection for the acoustic output from the text data that were created with the help of the word log [2].

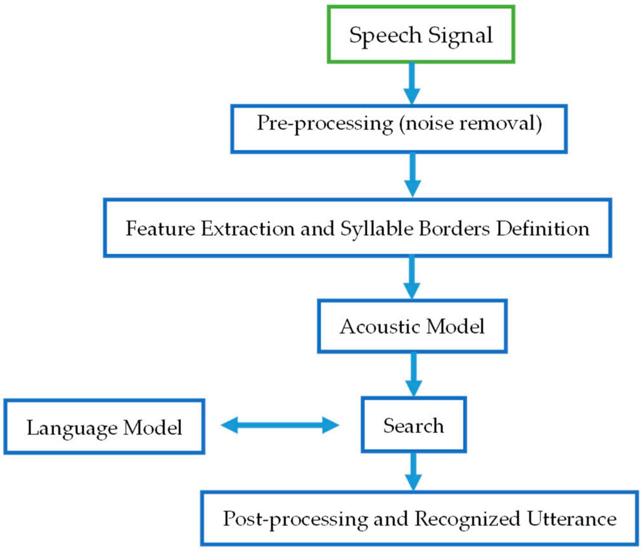

Figure 1. Basic flow of speech recognition systems.

This is the most basic flow of voice recognition techniques. Differences arise from the approaches used to create acoustic and language models.

In this study, a new method is proposed to improve the performance of Long Short Term Memory (LSTM) based Recurrent Neural Network (RNN)-trained speech recognition systems. The system is based on the principle of correcting recognition outputs with minimal increase in runtime.

Similar to the proposed method in this study, many different hybrid studies have been published in recent years. In the study [18] in which the Gaussian model was used with LSTM, a performance increase of approximately 0.5% to 0.2% was achieved compared to standard LSTM models. In the study [19], where a talking facial expression was intended to contribute to speech recognition performance, a certain increase was achieved for all situations. In the study by Kowari et al. [20], Deep Neural Network (DNN), Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) were used in combination. An increase in performance was achieved.

In Section 2 of this study, the working method of RNN-LSTM structure is explained, and, the structure of the proposed model and the working process are explained in Section 3 in detail. The comparative representation of the experiments performed to observe the contribution of the model and its results are described in Section 4. The evaluation of the results, some of the problems encountered and the studies that can be carried out for on the solving of these issues are presented in Sections 5 and 6.

2. Recurrent Neural Networks and Long Short Term Memory

Artificial neural network is a computation model based on the structure and functions of biological neural networks. It is like an artificial nerve used to receive process and transmit information. It is basically consists of 3 layers. The input layer communicates with the environment for values to be input to the neural network. The output layer is used to provide information out of the network. It collects and transmits the required information. The hidden layer is the layer that contains neurons with the activation function located on the input and output layer. It extracts and uses the properties of inputs coming from the previous layer. It can consist of multiple layers in itself. An artificial neural network includes an input layer, a certain number of hidden layers depending on the problem and an output layer. People produce new ideas by using their previous thoughts. Thus, the permanence of information is provided. Traditional neural network models, the first systems in which this type of thinking was model, cannot totally provide this idea. This forgotten old information constitutes a serious problem in the training of the network. RNN models were proposed to produce a solution to this problem. These models have loops that allow information to persist.

As shown in Figure 2, schema shows a chunk of neural network, Xn input value and y_k output value. A loop allows the information to be transmitted to the next step of the network.

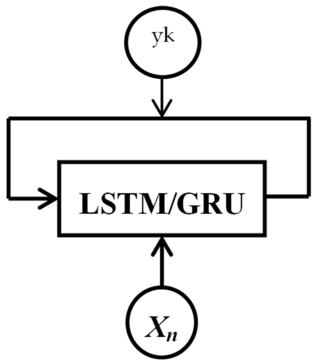

Figure 2. RNN give permission to recursion [21,22].

These loop system is not different from that of the classical neural network. It can be thought of as an architecture that includes multiple copies of a neural segment and receives messages from the previous step in each step. They work in the same way as arrays or lists. RNN structure has been applied to many problems such as speech recognition, language modelling, translation and image analysis in recent years and very successful results has been achieved.

The most important idea of RNN is that the previous data can be used in the next steps of the network. The amount of previous data is an important consideration for RNN structures. This structure is well suited for problems where recent information is sufficient to make prediction about the future. However, more information may be needed to solve some problems. For example, consider a language model that tries to predict the next word using previous words. This prediction requires subjects or verb given previously than the last word of the sentence. RNN structures cannot provide learning in cases where addiction is increased. The most important reason for this is the theoretically limited number of nodes in the network.

LSTM based RNN networks have been proposed to solve this problem. Long Short Term Memory (LSTM) is an RNN network that can learn long-term dependencies. This model was firstly proposed by Hochriter and Schmidhuber in 1997 [4]. Its popularity has increased over time and achieved very good results for many problems. It is well suited for avoiding long-term dependence problems. It is an idea that long-term recall of knowledge is a natural behavior for people (Figure 3).

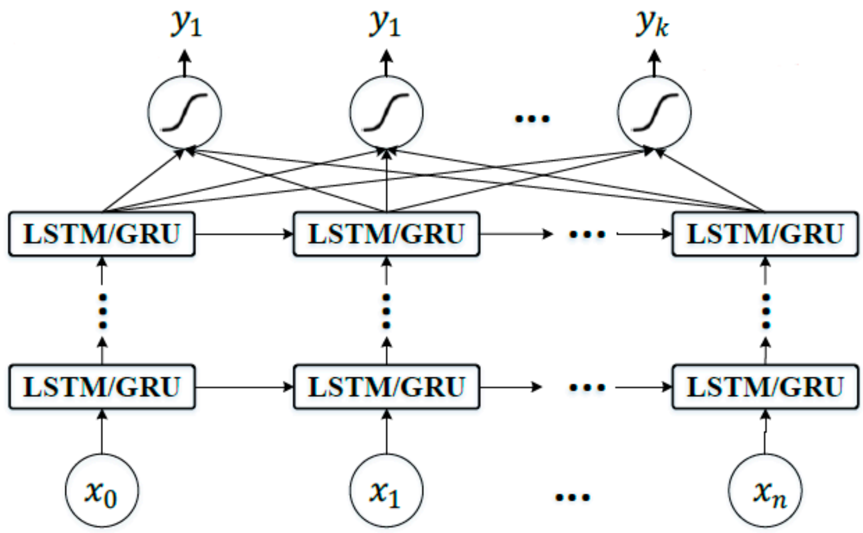

Figure 3. An Unrolled Recurrent Neural Network [21,22].

A standard RNN network is very simple and each layer can contain only one "tanh" function. However, in LSTM networks, this layer has a slightly different structure as shown in Figure 4.

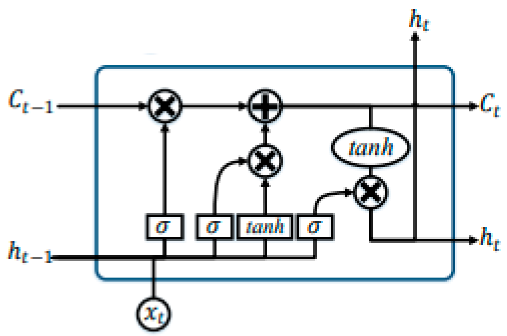

Figure 4. The repeating module in an LSTM contains four interacting layers [3].

The key structure of the LSTM is the line that runs over the diagram and works with small linear interactions. It is capable of adding and deleting information with the help of logic gates. LSTM works in a structure consisting of 6 steps.

Forget Gate Activation $$f_t = \sigma\left(W_f \,.\, [h_{t-1}, x_t] + b_f\right) \quad (1)$$
Input Gate: $$i_t = \sigma(W_i \,.\, [h_{t-1}, x_t] + b_i) \quad (2)$$
Candid Memory Cell Value: $$\check{C}_t = tanh(W_c \,.\, [h_{t-1}, x_t] + b_c) \quad (3)$$
New Memory Cell Value $$C_t = \left(f_t * C_{t-1} + i_t * \check{C}_t\right) \quad (4)$$
Final Output Gate Values: $$o_t = \sigma(W_o \,.\, [h_{t-1}, x_t] + b_o) \quad (5)$$
Final Output Gate Values: $$h_t = (o_t * \tan\mathrm{h}(C_t) \quad (6)$$

In the above formulas, each b is a bias vector, W is a weight matrix, and x_t is the input to the memory cell. i,c,f,o indices refer to input, cell memory, forget and output gates respectively.

The Step (1) is to decide which information is thrown away from the cell state. This decision is made using the sigmoid layer called the forget gate layer. It looks at h_{t-1} and x_t and generates an output of 0–1 for each c_{t-1} cell state. If it is 1, it means to completely keep it and 0 is to completely get rid of it.

Step (2) is to decide what new information to be kept in memory. It has two steps. First one is, decide which values to update with the sigmoid layer. Then, the tanh layer generates a vector for new candidate values(C~) in step (3).

It is necessary to update the old C_{t-1} status with the new C_t status. To forget the previous decision, it is multiplied by the f_t value and a new candidate value is created by i_t value in step (4).

Finally, it is necessary to decide what the cell output will be. This output depends on the current state of the cell but is a filtered version. Using the sigmoid function, the data to be extracted from the cell is selected in step (5). Then, the data is normalized with the tanh function in step (6). So the values between −1 and 1 are pushed. and multiplied by the output of the sigmoid function. Thus, only the part of the data that is decided is produced as output [21,22].

Using LSTM with RNN, remarkable results were obtained for many different problems. The next step from this stage is the structures that allow much larger data to be learned in each learning step. Many studies were carried out in relation to LSTM and derivatives and successful results were obtained on better learning systems [23–29]. Similar to LSTM structure, The Gated Recurrent Unit (GRU) is preferable for this study, as well. In the study of [30], comparing the LSTM and GRU approach, both models were compared and reported to produce similar results. When LSTM and GRU trainings were performed for the model proposed in this study, it was observed that more successful results could be obtained with LSTM. For this reason, the success of the hybrid model was demonstrated by creating a model with LSTM structure.

3. Proposed Correction Approaches for LSTM Speech Recognition

The LSTM based RNN structure can perform speech recognition with a certain level of performance on a test data. There are many factors such as dataset, model complexity, and the training time that affects this recognition process. For this reason, it is generally not possible to maintain satisfactory levels of system performance. Many different approaches have been proposed in the literature to improve the level of performance [31–35]. In addition to speech signal, features of human activities contribute to recognition processes. With these models, it is aimed to achieve significant increases such as our proposed model.

Using the comparison method with reference words together with the LSTM structure, an increase in performance can be achieved. Figure 5 shows the structure of proposed model.

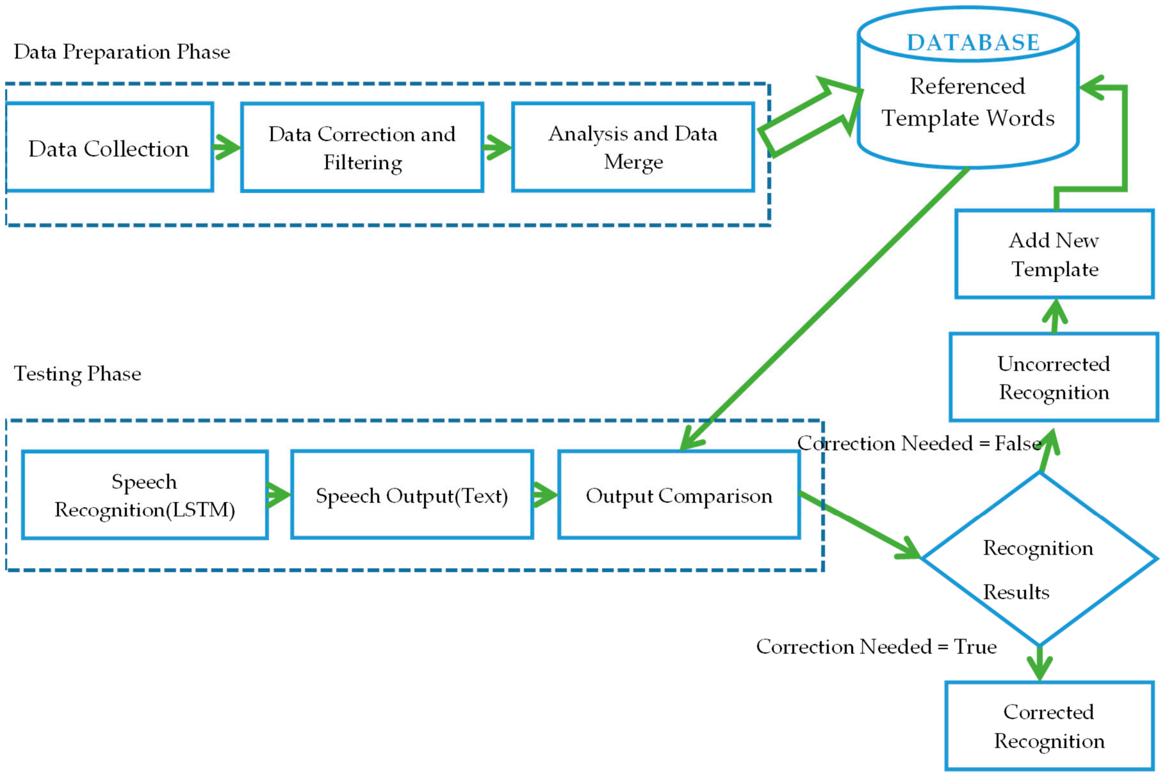

Figure 5. Framework of Speech Recognition with Correction Methodology.

There are basically two phases in this model. The creation of a database of "Referenced Template Words" is the first phase. Then, in the second phase, this database is compared with the LSTM system outputs and the correction of the system outputs is performed.

3.1. Data Preparation Phase

In the data preparation phase of the model proposed, primarily Turkish data collection process was performed. For this, 3 distributed datasets in Turkish were used. All of these datasets are created for different purposes. Therefore, in order to be used in this model, data correction and filtering operations are required. For instance, many editing procedures are needed, such as deleting schemas showing the suffix and roots of the words or correcting Turkish character problems. After these operations, 3 datasets were combined, the whole data were analyzed and the duplicated words were deleted and the type conversions were made to keep them in the same database tables. As a result, a "Referenced Template Words" database containing approximately 3 million unique Turkish words was created.

After the preparation of the text dataset, it is necessary to prepare an audio dataset. In this study, audio data of Middle East Technical University Microphone Speech v 1.0 [36,37] was used. It is a dataset prepared in Turkish language. Turkish is a phoneme based agglutinative language. Each letter is represented by a phoneme. However, in some cases, vowels and consonants may vary depending on where they are produced. Twenty letters in Turkish alphabet are represented by 45 phonetic symbols. In this dataset, a 193 speaker audio corpus and a pronunciation lexicon were developed. A new corpus and audio tools were created to ensure the accuracy of phonetic alignment and phoneme recognition. 91.2% of the automatically labelled phoneme boundaries are placed within 20 ms of hand-labelled locations for the Turkish audio corpus. The corpus is about the size of 600 Mbytes. The data has been digitally recorded with a Sound Balaster sound card on a PC at a 16 KHz sampling rate. The first 2000 sentences of the TIMIT [38] dataset were translated into Turkish. Afterwards, some studies aimed to improving the dataset yielded 2462 sentences with 9165 different words.

3.2. Testing Phase

In testing phase, an exemplary speech recognition model in RNN-LSTM structure was created. METU 1.0 Turkish speech set, which was suggested in the studies [36,37] and distributed over Linguistic Data Consortium (LDC) [39], was used for training of the model. The outputs of the standard recognition model were subjected to the comparison model process, of which the details are in Figure 5, and an increase in system performance was intended to be achieved.

The procedure for correcting the output of the LSTM speech recognition model within the proposed new model is shown in Figure 6. Accordingly, all outputs are subjected to a verification and correction process. Firstly, the outputs are checked for any corrections. If results are generated with 0% error rate, recognition process is terminated and system output is generated. In addition, this output is added into the database as a new "Referenced Template" if the word does not exist in the database. Moreover, if the recognition output is produced with uncorrected characters, it is ensured that these outputs are subjected to a correction process. "Referenced Template DB" that was previously prepared is used for this correction. Each output is subjected to different string distance calculation algorithms, some tests are subjected to 1 and some others to 11 different algorithms, in order to find the closest word in the database. Each output is subjected to this search and distance calculation process without exception and it is ensured that the closest reference word is found. This reference word is used as the system output instead of the incorrectly generated word. That is, instead of the text output generated using audio data, the new referenced word found by matching the incorrect outputs in the database is used.

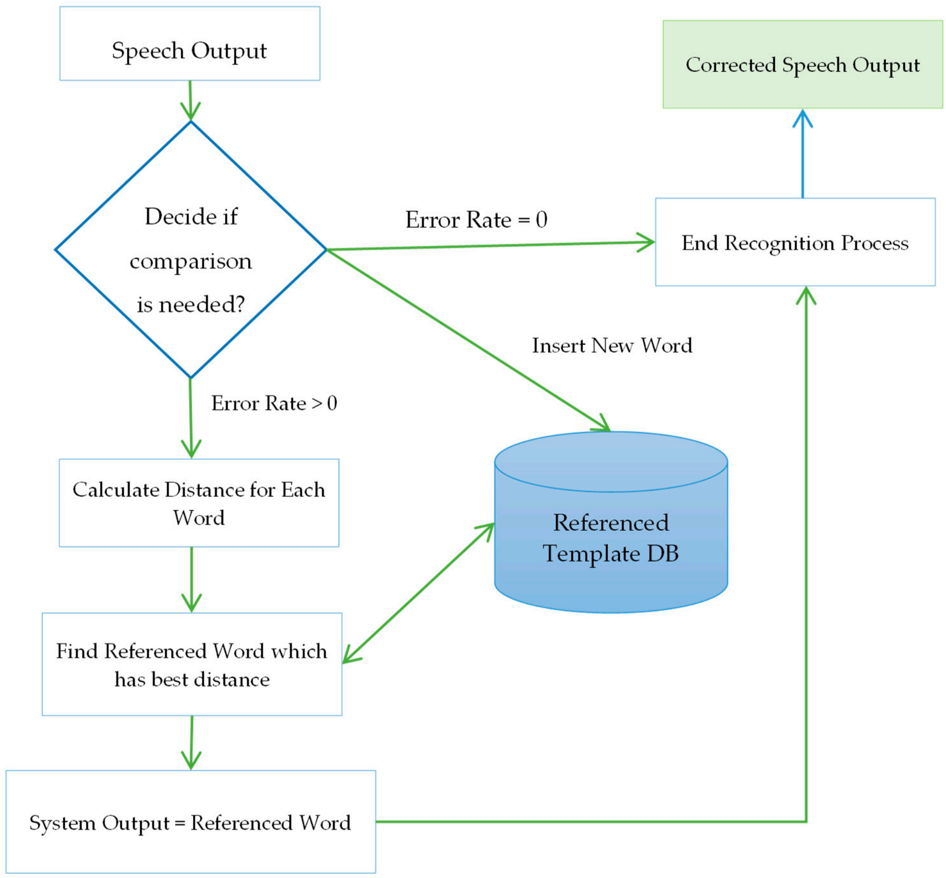

Figure 6. The proposed correction method for LSTM speech recognition.

The purpose of this proposed model is to correct system outputs with a few character errors and thereby improve system recognition performance. The pseudo code version of the algorithm is also presented in Figure 7. In the first step of this method, it is very important to reset the output value to be produced by LSTM model and the values that keep the distance of the words between LSTM model output and reference words. The LSTM-based speech recognition system is then performed and the output is produced. This output is checked for error rate is equal to 0%. This control has been put in order to contribute to a faster operation. The results that have 0% error rate, are not included the process. However, if recognition is performed incorrectly, all words in the reference table are sequentially compared and the distance of each word to the LSTM model output is calculated. After this calculation, an index with the minimum distance is determined and the related word is generated as a system output. A type of output correction is performed.

There are many different distance calculation algorithms for finding the nearest word. Although each algorithm has a different approach, the main goal is to make the correct estimation. At this stage, the most accurate approach would be to calculate each word with more than one distance algorithm and get the best value. However, it is natural that this is reflected in the process time as a significant increase. Therefore, it this study, the most suitable distance algorithm for the dataset was determined firstly by the tests and this method was used in the comparison process. The reference word closest to the recognition output with some errors is found by searching with the code given above.

```
Procedure Correction_Most_Matching_Method();
Output=""
Bestindex=0
If(Recognition Error Rate >0) then
/*Start Find Best Distance Word*/
    For each word of references template db
        Calculate distance with word and system output
    End For
    For each n
    /*n is the number of distance with word*/
    Search the best distance index(i)
    End For
    /*output correction process*/
    output = word(i)
Else
    /* End of learning process(No correction needed)*/
End If
```

Figure 7. PSEUDOCODE: The pseudo code of most-matching learning.

3.3. Datasets, Tools and Algorithms Used For Testing Proposed Model

In order to test the model proposed in different ways, audio data, text data, database tools, LSTM model and other libraries were needed.

LSTM models require audio data for training. In this study, audio data of "Middle East Technical University Turkish Microphone Speech v. 1.0" [36,37] audio data, which contain a comprehensive data that can be used in Turkish speech recognition studies, were used. 120 speakers (60 males and 60 females) speak 40 sentences each (approximately 300 words per speaker), which makes approximately 500 min of speech in total. The 40 sentences are selected randomly for each speaker from a triphone-balanced set of 2,462 Turkish sentences. The ages range from 19 to 50 years, with an average of 23.9 years.

It is essential to create a text data set that can be used as a template for the correction of LSTM speech recognition outputs. It needs to be able to cover almost all words in Turkish. For this purpose, 3 commonly used Turkish data group were preferred. These are Zemberek [40], BOUN Corpus [41] and METU 1.0 Speech Dataset [36,37]. Zemberek, published in 2010, is an open source library containing grammatical features specific to Turkish language and can be used in Natural Language Processing (NLP) studies. It contains approximately 1.15 million unique Turkish words. BOUN Corpus, published in 2008, is a Project to create a Turkish language resource. It contains approximately 1.4 million unique Turkish words. METU 1.0 audio data set was prepared for use in speech recognition studies and published in 2002. This data set contains approximately 7 thousand unique Turkish words.

The PostGreSql database tool [42] was used to store "Referenced Template Words" and perform a quick search. Python [43] programming language was used for training of LSTM speech recognition system and Java libraries were used for testing. Dill [44], librosa [45], namedtupled [46], numpy [47], python_speech_features [48], tensorflow [49] libraries were used for speech recognition with Python.

There are many algorithms available for string comparison of the model. In this work, 11 different distance calculation algorithm were used to test the system performance in a different ways, which are Levenshtein [50], Damerau Levenshtein [51], Jaro Winkler [52,53], Longest Common Subsequence [54], Metric LCS [54], Normalized Levenshtein [50], Optimal String Alignment [51], Precomputed Cosine [55], Qgram [56], Sift4 [57] and Weighted Levenshtein [24]. Thus, it was possible to observe the effects of different algorithms on the proposed model and make comparison in order to contribute to performance improvements.

4. Experiment and Results

In order to test the proposed data correction model, a medium scale Turkish training set was prepared. An LSTM RNN based model was prepared using an acoustic model with using Connectionist Temporal Classification (CTC) and deep learning. METU 1.0 audio dataset was used to test the application. The audio data were recorded in a quiet studio. The speech signal was recorded as a single channel at 16 Khz 16 bit resolution. The analysis frames are 20 ms wide and there is a 10 ms overlap. Different sample sets were created and the model proposed was tested. Three different text set were used for the tests. Zemberek Library [40], Boun Text Corpus [41] and Metu 1.0 text data [36,37] were preferred. All these datasets were examined and a text dataset containing more than 2M different Turkish words was obtained by combining them. For training and testing of the model, multiple training and test sets were prepared from audio dataset and the results were observed comparatively.

Two different methods were used to evaluate the proposed model in this paper. Firstly, the method was applied on more than one test, and as a result, the average contribution to system performance was observed. In the second method, the original sentence was compared with the LSTM model output and the model proposed output at each step of the 10000 Epoch tests, and the number of sentences were closer to the original sentence was counted. Thus, although the average value of the increase is important. How many sentences the proposed model had produced better results for in 10000 tests was also evaluated. Figure 8 shows the process of second method as an example.

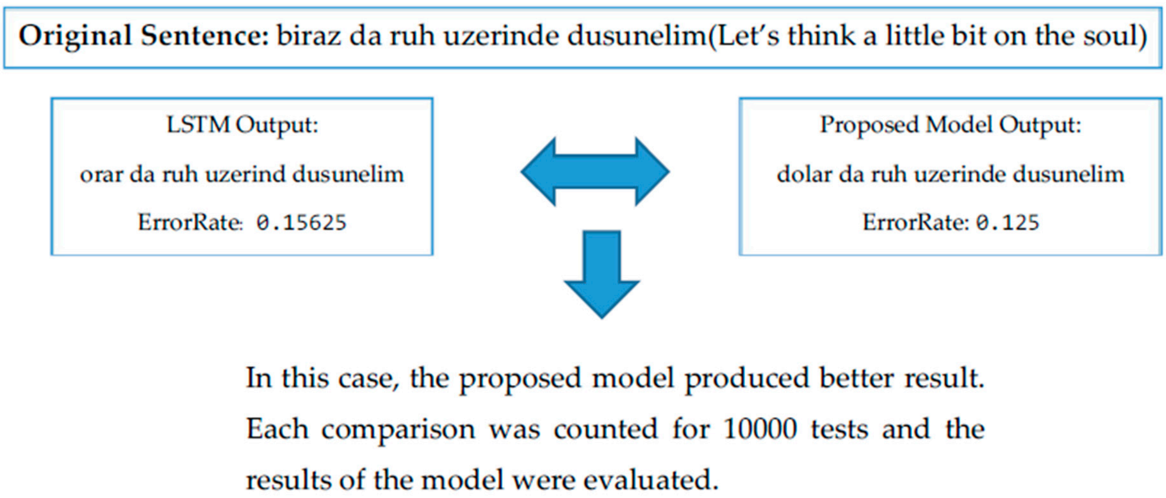

Figure 8. Sample test scheme of the proposed model.

Table 1 shows the test results of different distance algorithms. These results were obtained as a result of 10000 Epoch tests. When the results are examined, it is seen that different increase values in performance are obtained when different algorithms are used. This is due to the fact that distance algorithms produce different results for finding the closest words. The proposed model achieved maximum 3.84% increase in overall system recognition performance. In addition, a significant contribution is the before and after correction count when a comparison is made for each test in 10000 Epochs. Accordingly, after 10000 tests performed in the best condition, the model produced better results in 7711 after correction and 2289 before correction. This shows that the correction process is very useful not only for the overall performance of the system, but also in correcting inaccurate outputs throughout the system. A proportional difference of 27.21% is achieved on average.

Table 1. Recognition Results of LSTM Speech Recognition with Different Distance Algorithm.

Distance Algorithm	Error Rate Before Correction	Error Rate After Correction	Count of Better Performance(Not Corrected-Corrected)	Difference in Success Count (%)	Overall Performance Increase (%)
NormalizedLevenstain	0.3062	0.2730	2390-7610	52.20%	3.32%
DamerauLevenstein	14.8955	14.8143	3831-6169	23.38%	0.54%
JaroWinkler	0.2315	0.2093	3499-6501	30.02%	2.22%
LongestCommonSubsequence	17.7432	17.1749	3476-6524	30.48%	3.20%
MetricLCS	0.2835	0.2558	**2289-7711**	54.22%	2.76%
OptimalStringAlignment	14.9061	14.8344	3831-6169	23.38%	0.46%
PrecomputedCosine	0.2860	0.2798	4397-5703	13.06%	0.6%
Qgram	**27.5372**	**26.4720**	**3879-6121**	**22.42%**	**3.84%**
Levensthein	14.9278	14.8536	3851-6149	22.98%	0.49%
Sift4	19.8155	19.8039	4402-5598	11.96%	0.58%
WeightedLevensthein	14.9278	14.8536	3851-6149	22.98%	0.43%

Table 2 shows the basic parameters of the test models. 10 test sets were created with these parameters but working with different data. With the proposed model, Normalized Levensthein algorithm, which is one of the distance algorithms that provide the best contribution to the overall performance of the system, was kept constant and tests were performed for the increase in system performance for 10 different test sets prepared differently from each other. The test results are presented in Table 3.

Table 2. Experiment Parameters of Designed Model for LSTM.

Test Set	Optimization Algorithm	Learning Rate	Standard Deviation	Epoch Number	Batch Size	NOE	Mean
Set 1-10	GradientDescent	0.01	0.1	10000	1	1	0

NOE: Number of Examples.

Table 3. Recognition Results of LSTM Speech Recognition with Correction Method.

Test Set	Error Rate Before Correction	Error Rate After Correction	Count of Better Performance(Not Corrected-Corrected)	Difference in Success Count (%)	Overall Performance Increase (%)
NL-Test Set1	0.3062	0.2734	2417-7583	51.66%	3.28%
NL-Test Set2	0.2717	0.2572	3504-6496	29.92%	1.45%
NL-Test Set3	0.1934	0.1677	1810-8190	63.80%	2.57%
NL-Test Set4	0.2516	0.2321	2713-7287	45.74%	2.04%
NL-Test Set5	0.3758	0.3602	3166-6834	36.68%	1.56%
NL-Test Set6	0.3074	0.2719	3040-6960	39.20%	3.55%
NL-Test Set7	0.0934	0.0789	2557-7443	48.86%	1.45%
NL-Test Set8	0.3149	0.3007	4138-5862	17.24%	1.42%
NL-Test Set9	0.4265	0.4097	3883-6117	22.34%	1.68%
NL-Test Set10	0.1461	0.1307	2463-7537	50.96%	1.54%

Average Difference in Success Count: 40.64%; Average Performance Increase: 2.25%; Maximum Performance Increase: 3.55%; Minimum Performance Increase: 1.42%.

As shown in Table 3, the proposed model produced different results for different data sets. The purpose of these tests is to observe the contribution of the proposed model to the system performance in different test environments. According to this, the change of the test sets did not cause any change in the contribution to the system, but there was a difference in the contribution rate. An average system performance increase of 2.25% was achieved. The variability of the improvement ratio is directly proportional to the output produced by the system before correction. The model offers a contribution of more than 3% in cases where there are less character errors in a word but there are character errors in more than one word in a sentence. However, the improvement level drops to around 1.5% in cases where a word has more than one character errors.

Changes in the number of Epoch and the number of hidden layers of the model lead to changes in the level of contribution.

It improves the performance of system in any case but acts at levels ranging from 0.5% to 3.55% (Figure 9). Furthermore, the proportional difference in the number of recognition's before and after correction is much higher. On average, it produces more accurate outputs at 40.64%. It is seen that it makes a significant contribution. The model makes visible corrections in each test sentence. This process fixes many character errors. Numerical difference exceeds 50% indicates situations where the LSTM model does not perform learning or overall recognition performance of model remains below 20%. With the increase Epoch number, the differences in the numbers of successes decrease to more reasonable levels. This is the case where real learning takes place and numerical success goes down to average levels. In the case of realistic and good learning, both the contribution of the system to the overall performance exceeds over 3% and the difference in the number of success reaches over 40%. This shows that the proposed model can contribute more with a good learning network.

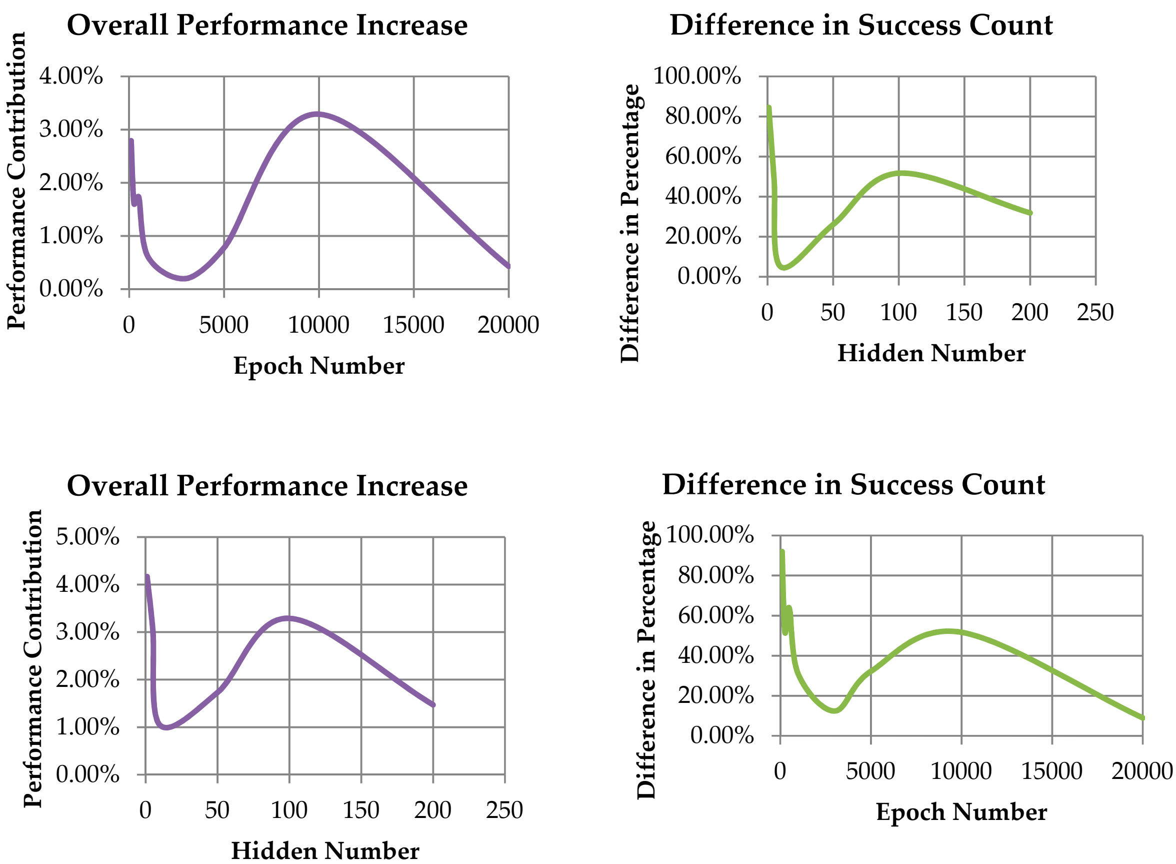

Figure 9. Effect of Epoch Number and Hidden Layer Count on Recognition Performance.

With the increase of performance, it is natural that the model proposed makes an increase in system calculation time. Since, after the system output, an output correction process is performed. After the output of the system was produced to observe how much the system was reflected as an increase of the overall runtime, we observe how long the verification method completed its operations and as a result we obtained the results as in Table 4. These results were achieved based on an Intel i5 processor and 8 GB ram memory. Approximately 0.04 s is required to correct each word during recognition. Assuming that an adult can speak an average of 200 words in 1 min, an extra 8 s is needed for a recognition model using the model proposed for creating speech data containing 1 min of continuous speech. These times are slightly longer in multiple comparison models using multiple distance calculation algorithms to find the closest word.

Table 4. Results of different recognition approach with METU 1.0.

Name of Authors	Recognition Approach	Vocabulary	Error Rate
Keser and Edizkan [58]	Common Vector Approach(CVA)	METU 1.0 Dataset	70 %
Aksoylar et al. [59]	HMM	METU 1.0 Dataset, SUVoice	Sports News: 37 %
Salor et al. [37]	HMM	METU 1.0 Dataset	29.2%
Çiloğlu et al. [60]	HMM, N-gram	METU 1.0 Dataset	35.91%
Büyük et al. [61]	HMM	METU 1.0 Dataset-2151 Test Data	3% increase performance
Proposed Model	LSTM	METU 1.0 Dataset	24.82%

Acceleration of the given processing times is possible with the necessary optimization techniques or faster computer infrastructure. However, we want to show here is that the proposed model requires an extra time in a standard speech recognition systems. It is a method that can be applied to many recognition systems with its effect on system result and run time comparison.

In this study, an LSTM based approach was preferred in speech recognition process and the proposed new approach contributed to a standard model. In addition to LSTM, many different algorithms such as Hidden Markov Model (HMM), Modified HMM, Embedded HMM, Gaussian Mixture Model (GMM), and Support Vector Machines (SVM) can be used. There have been many studies using these methods from past to present [58–65]. In order to compare the results obtained in this study, the results of the studies using modelling approaches other than LSTM and performing tests with METU 1.0 [36,37] dataset are shown in Table 4.

When the results shown in Table 4 are examined, the error rate is approximately 30%. However, small differences in the text data used in language model had effects on the results. The LSTM based model has an average error rate of 26.87%. In general, better results are produced than that of the models using HMM, CVA or N-gram. But, with the proposed new approach to reduce error rate, this performance level was further improved and reduced to 24.82%.

5. Conclusions

In this study, a viable output correction mechanism for speech recognition systems using LSTM modelling is proposed. Thus, an increase in system performance is aimed.

This proposed hybrid model provides a performance increase of approximately 2.25% with small increases in system runtime. Once the system requirements are established, it is very easy to implement.

The diversification of the audio data set and the enrichment of the word set will contribute to the performance. In addition, optimization of the distance calculation algorithms and improving database schemas would allow a considerable reduction in the processing times.

Nowadays, a speech recognition system that can operate in all situations, that has a 100% output performance and that can be used for all languages has not yet been developed. So, the model proposed can be used for many studies and structures to allow performance improvements at certain levels.

6. Future Works and Restriction

Although the new model proposed generally produces successful results, some limitations were encountered during the tests. The most important reason why the system cannot contribute more to the overall recognition performance is that the issue with the space character have not been resolved. As shown in Figure 10, the output of the system, which was made with almost 100% accuracy, is corrupted by the correction process. Solving this problem would reflect a significant increase in overall recognition performance.

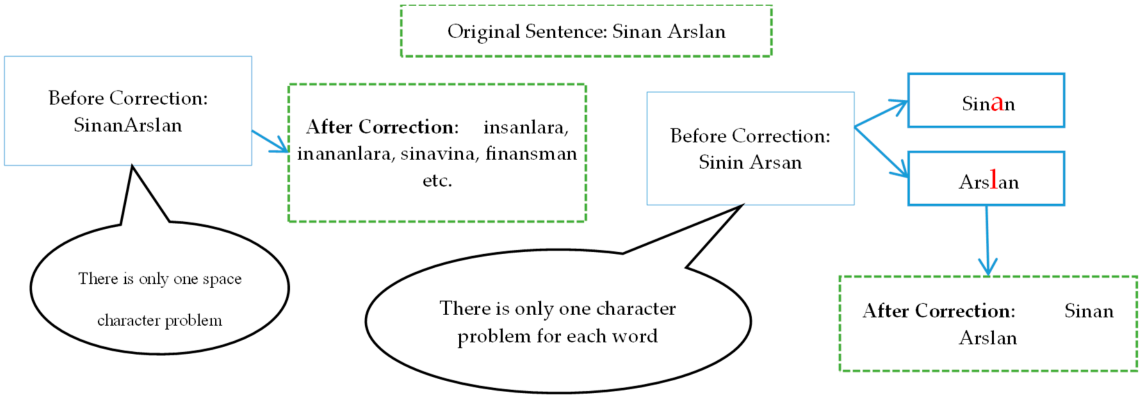

Figure 10. Restriction of Method – space character problem.

The second problem is that different distance calculation algorithms cannot be used together because of different calculation structures. If the necessary optimization and matching studies were performed on the distance values of the algorithms, it would be possible to find the correct template words. The optimization process would contribute to the system recognition performance.

Finally, the biggest problem of the system is the delays in the operating time. Since this method is a hybrid model, it requires a longer run time than that of the standard models. It is necessary to reduce this time to tolerable levels. Otherwise, it cannot be used in real-time speech recognition applications.

Author Contributions: Conceptualization, R.S.A. and N.B.; methodology, R.S.A. and N.B.; software, R.S.A.; validation, R.S.A. and N.B.; formal analysis, R.S.A. and N.B.; investigation, R.S.Arslan and N.B.; resources, R.S.A. and N.B.; data curation, R.S.A. and N.B. writing—original draft preparation, R.S.A. and N.B.; writing—review and editing, R.S.A. and N.B.; visualization, R.S.A.; supervision, N.B.; project administration, N.B."

Funding: There is no extra funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tran, D.T. Fuzzy Approaches to Speech and Speaker Recognition. Ph.D. Thesis, Canberra University, Canberra, Australia, 2000.
2. Uddin, M.T.; Uddiny, M.A. Human activity recognition from wearable sensors using extremely randomized trees. In Proceedings of the International Conference on Electrical Engineering and Information Communication Technology (ICEEICT), Dhaka, Bangladesh, 13–15 September 2015; pp. 1–6.
3. Jalal, A. Human activity recognition using the labelled depth body parts information of depth silhouettes. In Proceedings of the 6th International Symposium on Sustainable Healthy Buildings, Seoul, Korea, 27 February 2012; pp. 1–8.
4. Jalal, A.; Uddin, M.Z.; Kim, J.T.; Kim, T.-S. Daily Human Activity Recognition Using Depth Silhouettes and R Transformation for Smart Home. In *International Conference on Smart Homes and Health Telematics*; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6719, pp. 25–32.
5. Ahad, M.A.R.; Kobashi, S.; Tavares, J.M.R. Advancements of image processing and vision in healthcare. *J. Healthcare Eng.* **2018**, 1–3. [CrossRef] [PubMed]
6. Jalal, A.; Quaid, M.A.K.; Hasan, A.S. Wearable Sensor-Based Human Behaviour Understanding and Recognition in Daily Life for Smart Environments. In Proceedings of the International Conference on Frontiers of Information Technology (FIT), Islamabad, Pakistan, 18–20 December 2017; pp. 1–7.
7. Arora, S.J.; Singh, R.P. Automatic Speech Recognition: A Review. *Int. J. Comput. Appl.* **2012**, *60*, 34–44.
8. Chen, I.K.; Chi, C.Y.; Hsu, S.L.; Chen, L.G. A real-time system for object detection and location reminding with RGB-D camera. In Proceedings of the Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 10–13 January 2014; pp. 1–2.
9. Kamal, S.; Jalal, A.; Kim, D. Depth Images-based Human Detection, Tracking and Activity Recognition Using Spatiotemporal Features and Modified HMM. *J. Electr. Eng. Technol.* **2016**, *11*, 1857–1862. [CrossRef]

10. Fonseca, L.M.G. Digital image processing in remote sensing. In Proceedings of the International Proceedings Conference on Computer Graphics and Image Processing, Rio de Janerio, Brazil, 11–14 October 2009; pp. 59–71.
11. Jalal, A.; Kim, Y. Ridge body parts features for human pose estimation and recognition from RGB-D video data. In Proceedings of the 5th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Hefei, China, 11–13 July 2014; pp. 1–6.
12. Anusuya, M.A.; Katti, S.K. Speech Recognition by Machine: A Review. *Int. J. Comput. Sci. Inf. Secur.* **2009**, *4*, 181–205.
13. Rathore, M.M.U.; Ahmad, A.; Paul, A.; Wu, J. Real-time continuous feature extraction in large size satellite images. *J. Syst. Arch.* **2016**, *64*, 122–132. [CrossRef]
14. Farooq, A.; Jalal, A.; Kamal, S. Dense RGB-D Map-Based Human Tracking and Activity Recognition using Skin Joints Features and Self Organizing Map. *KSII Trans. Int. Inf. Syst.* **2015**, *9*, 1859–1869.
15. Jalal, A.; Kim, S. Global security using human face understanding under vision ubiquitous architecture system. *World Acad. Sci. Eng. Technol.* **2006**, *13*, 7–11.
16. Yoshimoto, H.; Date, N.; Yonemoto, S. Vision-based real-time motion capture system using multiple cameras. In Proceedings of the IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems, Las Vegas, NV, USA, 7 October 2003; pp. 247–251.
17. Kamal, S.; Jalal, A. A hybrid feature extraction approach for human detection, tracking and activity recognition using depth sensors. *Arab. J. Sci. Eng.* **2016**, *41*, 1043–1051. [CrossRef]
18. Lam, M.W.; Chen, X.; Hu, S.; Yu, J.; Liu, X.; Meng, H. Gaussian process LSTM recurrent neural network language models for speech recognition. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Brighton, UK, 12–17 May 2019; pp. 7235–7239.
19. Afouras, T. Deep audio-visual speech recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *12*, 1–13. [CrossRef]
20. Kowsari, K.; Heidarysafa, M.; Brown, D.E.; Meimandi, K.J.; Barnes, L.E. Rmdl: Random multi model deep learning for classification. In Proceedings of the 2nd International Conference on Information System and Data Mining, Lakeland, FL, USA, 9–11 April 2018; pp. 19–28.
21. Kowsari, K.; Brown, D.E.; Heidarysafa, M.; Meimandi, K.J.; Gerber, M.S.; Barnes, L.E. Hdltex: Hierarchical deep learning for text classification. In Proceedings of the 16th IEEE International Conference on Machine Learning and Applications(ICMLA), Cancun, Mexico, 18–21 December 2017; Available online: https://arxiv.org/abs/1709.08267 (accessed on 1 February 2019).
22. Kowsari, K. Text classification algorithms: A survey. *Information* **2019**, *10*, 1–68. [CrossRef]
23. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *98*, 1735–1780. [CrossRef]
24. Kaisheng, Y. ; Deep-Gated Recurrent Neural Networks. *arXiv* **2015**, arXiv:1508.03790.
25. Koutnik, J.; Greff, K.; Gomez, F.; Schmidhuber, J. A Clockwork RNN. *arXiv* **2014**, arXiv:1402.3511.
26. Greff, K.; Srivastava, R.K.; Koutník, J.; Steunebrink, B.R.; Schmidhuber, J. LSTM: A Search Space Odyssey. *Trans. Neural Netw. Learn. Syst.* **2016**, *28*, 2222–2232. [CrossRef] [PubMed]
27. Jozefowicz, R.; Zaremba, W.; Sutskever, I. An Empirical Exploration of Recurrent Network Architectures. *Int. Conf. Mach. Learn.* **2015**, *37*, 2342–2350.
28. Gers, F.A.; Schmidhuber, J. Recurrent Nets that Time and Count. In Proceedings of the IEEE-INNS-ENNS International Joint Conference, Como, Italy, 24–27 July 2000; pp. 1–6.
29. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv* **2014**, arXiv:1406.1078.
30. Irie, K.; Tüske, Z.; Alkhouli, T.; Schlüter, R.; Ney, H. LSTM, GRU, Highway and a Bit of Attention: An Empirical Overview for Language Modelling in Speech Recognition. *Interspeech* **2016**, *9*, 3519–3523.
31. Piyathilaka, L.; Kodagoda, S. Gaussian mixture based HMM for human daily activity recognition using 3D skeleton features. In Proceedings of the International Conference on Industrial Electronics and Applications, Melborne, Australia, 19–21 June 2013; pp. 1–6.
32. Jalal, A.; Kamal, S.; Kim, D. A Depth Video Sensor-Based Life-Logging Human Activity Recognition System for Elderly Care in Smart Indoor Environments. *Sensors* **2014**, *14*, 11735–11759. [CrossRef] [PubMed]
33. Jalal, A.; Kim, Y.-H.; Kim, Y.-J.; Kamal, S.; Kim, D. Robust human activity recognition from depth video using spatiotemporal multi-fused features. *Pattern Recognit.* **2017**, *61*, 295–308. [CrossRef]

34. Jalal, A.; Kamal, S.; Kim, D. Individual detection-tracking-recognition using depth activity images. In Proceedings of the 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), Goyang City, Korea, 28–30 October 2015; pp. 450–455.
35. Wu, H.; Pan, W.; Xiong, X.; Xu, S. Human activity recognition based on the combined SVM&HMM. In Proceedings of the IEEE International Conference on Information and Automation (ICIA), Hailar, China, 28–30 July 2014; pp. 219–224.
36. 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 Proceedings of the International Conference on Spoken Language Processing (ICSLP), Denver, CO, USA, 16–20 September 2002; pp. 1–5.
37. Salor, Ö.; Pellom, B.L.; Ciloglu, T.; Demirekler, M. Turkish speech corpora and recognition tools developed by porting SONIC: Towards multilingual speech recognition. *Comput. Speech Lang.* **2007**, *21*, 580–593. [CrossRef]
38. Garofolo, J.S.; Fisher, W.M.; Fiscus, J.G.; Pallett, D.S.; Dahlgren, N.L. *DARPA TIMIT: Acoustic-Phonetic Continuous Speech Corpus CD-ROM*; NIST: Gaithersburg, Maryland, USA, speech disc 1-1.1; 1993.
39. Liberman, M.; Cieri, C. The creation, distribution and use of linguistic data. In Proceedings of the First International Conference on Language Resources and Evaluation. European Language Resources Association, Granada, Spain, 28–30 May 1998; Available online: https://catalog.ldc.upenn.edu/ (accessed on 1 January 2017).
40. Akın, A.A.; Akın, M.D. Zemberek, an open source nlp framework for Turkic languages. *Structure* **2007**, *10*, 1–5.
41. Sak, H.; Güngör, T.; Saraçlar, M. Turkish language resources: Morphological parser, morphological disambiguator and web corpus. In *Advances in Natural Language Processing*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 417–427.
42. Koopmann, J. Database Journal–The Knowledge Center for Database Professionals; 2003. Available online: https://www.postgresql.org/ (accessed on 1 January 2015).
43. Van Rossum, G. *Python*; Corporation for National Research Initiatives(CNRI): Reston, VA, USA, 1995; Available online: https://www.python.org/ (accessed on 5 January 2015).
44. McKerns, M.M.; Strand, L.; Sullivan, T.; Fang, A.; Aivazis, M.A. Building a framework for predictive science. In Proceedings of the 10th Python in Science Conference, Austin, TX, USA, 11–16 July 2011; Available online: https://pypi.org/project/dill (accessed on 10 January 2015).
45. McKerns, M.M.; Aivazis, M. Pathos: A Framework for Heterogeneous Computing. 2010. Available online: https://librosa.github.io/librosa/ (accessed on 15 January 2015).
46. Hettinger, R. Namedtuple Factory Function for Tuples with Named Fields. Available online: https://docs.python.org/2/library/collections.html (accessed on 17 January 2015).
47. Olpihant, T.E. Guide to Numpy. Ph.D. Thesis, MIT University, Cambridge, MA, USA, 2006. Available online: https://www.numpy.org/ (accessed on 15 February 2017).
48. Lyons, J. Python Speech Features. 2013. Available online: https://python-speech-features.readthedocs.io/en/latest/ (accessed on 15 February 2017).
49. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. Tensorflow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, GA, USA, 2–4 November 2016; Available online: https://www.tensorflow.org/install/pip (accessed on 20 February 2017).
50. Levenshtein, V.I. Binary codes capable of correcting deletions, insertions, and reversals. *Sov. Phys. Dokl.* **1996**, *10*, 707–710.
51. Damerau, F.J. A technique for computer detection and correction of spelling errors. *Commun. ACM* **1964**, *7*, 171–176. [CrossRef]
52. Jaro, M.A. Advances in record linkage methodology as applied to the 1985 census of Tampa Florida. *J. Stat. Assoc.* **1989**, *84*, 414–420. [CrossRef]
53. Winkler, W.E. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. In Proceedings of the Section on Survey Research Methods. American Statistical Association, Washington, DC, USA, 1990, 10 January 1990; pp. 354–359.
54. Aldous, D.; Diaconis, P. Longest increasing subsequences: from patience sorting to the Baik–Deift–Johansson theorem. *Bull. Am. Math. Soc.* **1999**, *36*, 413–432. [CrossRef]

55. Singhal, A. Modern Information Retrieval: A Brief Overview. *Bull. IEEE Comput. Soc. Tech. Comm. Data Eng.* **2001**, *24*, 35–43.
56. Lu, J.; Lin, C.; Wang, W.; Li, C.; Wang, H. String similarity measures and joins with synonyms. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, New York, NY, USA, 22–27 June 2013; pp. 373–384.
57. Siderite Zackwehdex. Super Fast and Accurate String Distance Algorithm: Sift4. Available online: https://siderite.blogspot.com/2014/11/super-fast-and-accurate-string-distance.html (accessed on 11 November 2017).
58. Keser, S.; Edizkan, R. Phonem-Based Isolated Turkish Word Recognition with Subspace Classifier. In Proceedings of the IEEE Signal Processing and Communications Applications Conference (SIU), Antalya, Turkey, 9–11 April 2009; pp. 93–96.
59. Aksoylar, C.; Mutluergil, S.O.; Erdogan, H. The Anatomy of a Turkish Speech Recognition System. In Proceedings of the IEEE Signal Processing and Communications Applications Conference (SIU), Antalya, Turkey, 9–11 April 2009; pp. 512–515.
60. Çiloğlu, T.; Çömez, M.; Şahin, S. Language Modelling for Turkish as a Agglutinative Languages. In Proceedings of the IEEE Signal Processing and Communications Applications Conference (SIU), Kuşadası, Turkey, 28–30 April 2004; pp. 1–2.
61. Büyük, O.; Erdoğan, H.; Oflazer, K. Using Hybrid Lexicon Units and Incorporating Language Constraints in Speech Recognition. In Proceedings of the IEEE Signal Processing and Communications Applications Conference, Kayseri, Turkey, 16–18 May 2005; pp. 111–114.
62. Jalal, A.; Quaid, M.A.; Sidduqi, M.A. A Triaxial Acceleration-based Human Motion Detection for Ambient Smart Home System. *Int. Conf. Appl. Sci. Technol.* **2019**, *7*, 1–6.
63. Ahmad, J. Robust spatio-temporal features for human interaction recognition via artificial neural network. In Proceedings of the IEEE Conference on International Conference on Frontiers of Information Technology, Islamabad, Pakistan, 18–20 December 2018.
64. Jalal, A.; Uddin, M.Z.; Kim, J.T.; Kim, T.S. Recognition of Human Home Activities via Depth Silhouettes and R Transformation for Smart Homes. *Indoor Built Environ.* **2012**, *21*, 184–190. [CrossRef]
65. Jalal, A.; Rasheed, Y.A. Collaboration achievement along with performance maintenance in video streaming. In Proceedings of the IEEE Conference on Interactive Computer Aided Learning, Villach, Austria, 26–28 September 2007; pp. 1–8.