Birzeit University

Speech-Based CALL System to Evaluate the Meaning and Grammar Errors in English Spoken Utterance

by

Mohammad Ateeq

supervisor

Dr. Abualsoud Hanani

A thesis submitted in partial fulfillment for the degree of Master of Computing

in the

Faculty of Engineering and Technology

September 2019

Declaration of Authorship

I, Mohammad Ateeq, declare that this thesis titled, ' Speech-Based CALL System to Evaluate the Meaning and Grammar Errors in the Student's Spoken Utterance' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Abstract

In this research, we are developing a CALL (Computer Assisted Language Learning) system to evaluate the English spoken sentences grammatically and linguistically. We give the user a certain prompt written in his native language, then the response is recorded as English audio file. The English spoken response is converted to text using baseline English DNN-HMM ASR and another two commercial ASRs (Google and Microsoft Bing). The produced transcription is assessed in terms of language and meaning errors. Grammatical errors are detected using English grammar checker, part of speech analysis and extracting incorrect bi-grams from grammatically incorrect responses. Errors related to the meaning are detected using novel approaches which measure the similarity between the given response and stored set of reference responses. The training and testing datasets of spoken CALL shared task 2017 and 2018 were used in all of our experiments presented in this thesis.

We propose three main approaches to build this CALL system. The first approach is rule-based, which take a final decision about the given response (accept or reject) by passing audio transcription given by ASR (text) through a sequence of pipelined stages and rules. Each rule checks if the response has a language error or not. If a rule can not detect any errors, it passes the response to the next rule, and so on. In the second approach, the genetic algorithm was combined with firs approach to tune the parameters and thresholds used in each rule. The third approach is a machine learning model which predicts the final decision, accept or reject. Different types of features were extracted from the response and used in these approaches. The universal sentence encoder was used to encode each sentence into 512-dimensional vector to represent the semantic features of the response. Also, we propose a binary embedding approach to produce 438 binary features vector from the response. To assess the grammatical errors, a set of features were extracted using the grammar checker tool and part of speech analysis from the text response. Finally, the best two DNN models have been fused together to enhance the system performance. D-score was used as a performance metric in all of our experiments. The D-score of our three proposed systems are 6.5, 14.4 and 13.87, respectively. Compared with the results of similar systems (spoken CALL shared task 2018) published in Interspeech 2018, our second and third systems outperform them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over time, the introduction of Computer-Assisted Language Learning (CALL) models is a pioneer factor in development of speech and language technology especially after integrating Automatic Speech Recognition (ASR) as one of the components. CALL system can better help improve language skill of the learners, when it understands what they are saying. To date, most of the common speech-based CALL systems focus on the pronunciation quality of the second language. A good and well-documented example of these systems is the EduSpeak system [5] which plays the student a recorded sentence, asks them to imitate it, and then rates them on the accuracy of their imitation, giving advice if appropriate on how to improve pronunciation or prosody. There is no doubt that this is useful, but doesn't give the student a real opportunity to practice spoken language skills. Rayner et. al. in [6] took this a further step by building a speech-based CALL system by which students can interact and response to the systems prompts. This system prompts the student in his/her L1 language indicating in an indirect way what he/she is supposed to say in the L2 language. Then, the systems automatically assess the spoken response, based on the grammar and linguistic, and provides a feedback. In this project, we are building a speech-based CALL system on this basic pattern and studying the impact of different techniques applied in the natural language processing of the system performance.

The main goal of our system is to help the students to practice their conversational skills where we give the student and a text prompt (in his language) and the response is recorded as English audio file. The task is to determine if the response is correct with respect to grammar and linguistic rules. Then we will provide the students with feedback about their level.

1.1 System Overview

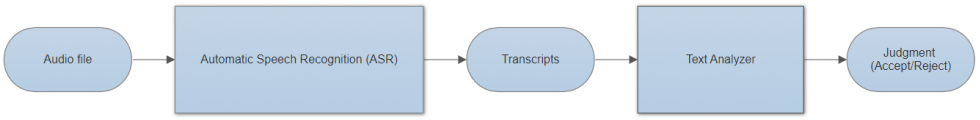

FIGURE 1.1: The System Overview.

The abstract view of our system is shown in the fig1.1. It consists of ASR (in our case English ASR) which converts spoken response into English text, and text processing unit which compares the transcript of spoken response (output of ASR) with a pre-defined expected and correct responses to find out if the user response is accepted for the displayed prompt or not. For ASR component, we used state-of-the-art English ASR (based on the Deep Neural Network, DNN technology). Regarding to text analyzer component, various techniques in the natural language processing were investigated for estimating similarity between the user response and the stored correct responses. A list of possible responses for each prompt is defined in the XML file .For example, suppose the prompt قل: اسمي محمد has three responses that defined the XML file: I am mohammad, I am called mohammad and my name is mohammad. Let us consider some easy cases:

- If the sentence in the audio file is I am Mohammad and the ASR component gets all the words right, then the Text Analyzer will accept this response.
- If the sentence in the audio file is I dont understand and the ASR component gets all the words right, then the Text Analyzer will reject this response(there is no match to any defined response).

It is a bit easy to build a system which can handle such a simple case, but it is more challenging to handle more complicated cases as in the following examples:

- If the sentence in the audio file is Why name is Mohammad and the output of the ASR component is my name is Mohammad(the language model adds some modification) then the text analyzer will matche the string with the defined response, but it should produce a false accept.
- If the sentence in the audio file is I'm Mohammad and the ASR component gets all the words right then the text analyzer will not find an exact match with defined responses. In this case it should accept the response.

The response is considered as Linguistically correct one if both vocabulary and grammar are accepted. The response has a correct meaning if the answer is meaningful related to the provided prompt.

1.2 Objectives

The main goal of this project is to develop a CALL system which helps English learners to exercise and improve speaking skills in English conservation. In particular, we aim to perform the following subgoals:

- Develop an algorithm to measure the similarity between the student response and each possible reference.
- Integrate existing embedding models to produce a high dimensional vector from the text. Also, developing different models to produce embedded vectors.
- Apply the state-of-the-art techniques in natural language processing to enhance the overall performance in our system.
- Determining the best set of syntactic and semantic features to increase the D score of our system.
- Studying the possibility of applying the optimization technique in this problem.
- improve the grammar checker by analyzing the POS tagging for all possible responses that are defined in the grammar XML file.
- Make the system online and accessible to the public.

1.3 Research Questions

In this project, we will try to find an answer to the following research questions:

- To what extent this system can help English learners to improve their speaking skills?
- What is the impact of ASR accuracy on the performance of the overall system?
- What is the impact of the text analyzer on the system performance?
- What are the best embedding technique to produce the feature vector from the spoken utterances ?

- Is it possible to use more than one ASR to improve the decision accuracy.
- What is the best algorithm to determine the similarity between the student response and its possible references.
- What is the best machine learning model to predict the decision?
- What is the effect of training data size on building the DNN model?
- what are the most effective and representative features to detect the semantic and linguistic errors ?
- Can the genetic algorithm be applied on such problem as feature selection technique to select most effective features ?

1.4 Thesis outline

Chapter 2 presents the CALL shared task in details. First, it shows how the data were collected for the first and second edition of this task. Then, it shows how to evaluate the system performance. Finally, it explains the components of the baseline system and many enhancements to these components. In Chapter 3, we explain the genetic algorithm which is used in our proposed system. Chapter 4 provides many technical details about the ASR component. In Chapter 5, we explain the machine learning models that used in our experiments. The technical details for our proposed system are described in chapter 6. The experiments results are reported in chapter 7. Also, it covers the discussion and comparison between the proposed systems that described in chapter 6. Finally, we conclude our results in chapter 8.

1.5 Key contributions of this thesis

The contribution for this paper is as follows:

- Developing three systems to solve the CALL shared task. Each system follows different approach to take the final decision. Our system achieved competitive results compared with other published results.
- Employ the universal sentence to encode each response into 512-dimensional vector which can capture the semantic of the response.
- Applying a genetic algorithm to optimize the parameters of our developed model to enhance the system performance. This contribution was published in INTERSPEECH2018 [7].

– We proposed a binary embedding approach to produce 438 binary features vector from the student response. This contribution was submitted to IEEE-ICASSP-2019.

Chapter 2

Overview of the CALL shared task

University of Geneva released a Computer Assisted Language Learning (CALL) Shared Task System [8] to build a system gives the student a German prompt and then decide to accept or reject the response after the processing the recorded English audio file. There are two editions of CALL shared task. The first one was organized in 2017 and the second edition was released in 2018 [1]. The results of the first edition of the task were presented at the SLaTE workshop in August 2017, whereas the best achieved systems of the second edition were published in INTERSPEECH2018. The highest D score of 4.766 and 19.088 was achieved in the first and second edition respectively. In addition, each entry in the dataset includes a transcription of recorded wav file by human experts, prompt, a meaning evaluation (correct/incorrect), and an overall evaluation (correct/incorrect) done by human experts. The basic assessment method performs speech recognition on spoken response and then accepts it if the recognized text matches at least one of the reference responses corresponding to the given prompt [9]. The system accepts responses that are correct in terms of grammar and meaning. Also, it rejects grammatically and linguistically incorrect responses according to the experts judgments [10].

CALL shared task 2018 is similar to the original one, in which it consists of two phases; the first one focuses on the effect of speech recognizer and the other focuses on the effect of text processing at the overall system performance. Consequently, the input of the speech-processing version consists of an a German text prompt, identifier, and a speech file containing an English language response. Regarding to the text-processing version, the text obtained from a baseline ASR system [11] was added. In addition, it provides a set of possible responses corresponding to each prompt.

[1] http://regulus.unige.ch/spokencallsharedtask

2.1 Data description

The data provided for the first and second edition of this CALL shared task were collected from different German speaking schools in 2014 and 2015. The speakers are German-speaking Swiss students. The speakers age is ranging from 12 to 15 years. Each participant was asked to response verbally in English to a given German text prompt. Three native English language experts judged each response as accepted or rejected in terms of both language grammar and meaning. For each prompt, a number of possible accepted responses were added by the experts and used as a correct reference responses. The data was divided into training set and testing set. In the first edition, the responses in the training data were divides as following: 3880 responses are correct in term of meaning and grammars, 540 responses are incorrect in terms of meaning or grammars and 802 responses are correct in term of meaning but not correct in term of grammar. According to the test data, it consists of 996 responses that were divided into: 716 responses are correct in term of meaning and grammars, 121 responses are incorrect in terms of meaning or grammars and 159 responses are correct in term of meaning but not correct in term of grammar. Regarding to the second edition, the training set contains 6698 utterances and the testing set contains 1000 utterances. The gender, age, English proficiency and motivation of the participants made balanced in the training and testing subsets. The recording environment is not perfect due to the background noises in schools. Table 2.1 shows the information of the data from the 2017 and 2018 tasks [12].

TABLE 2.1: Numbers of accepts/rejects in different datasets.

Dataset	**No of accepts**	**No of rejects**	**Total**
2017 Training	3,880	1,342	5,222
2017 Test	716	279	995
2018 Training	4,418	2,281	6,698
2018 Test	750	250	1,000

2.2 Evaluation metric

The annotators labeled each data item according to its linguistic correctness and its meaning. The correctness of the vocabulary and grammar were used for the linguistic assessment. On the other hand, the meaning was judged according to the context of the given prompt. Consequently, the response is rejected if either the linguistic or the meaning is incorrect, and accepted when both are correct. However, it makes more 'sense' for the system to accept meaningful responses with language mistakes, than to accept linguistically correct with meaningless responses. Thus, for each prompt, the

system's output (correct or incorrect) falls into one of the following four categories compared with the language and meaning gold standards provided by the annotators:

1. **Correct Reject (CR)**: It represents the number of utterances where the system rejects students response which is incorrect in term of meaning or language.
2. **Correct Accept (CA)**: It represents the number of utterances where the system accepts students response which is correct in meaning and it has no linguistic error.
3. **False Reject (FR)**: It represents the number of utterances where the system rejects students response which is correct in meaning and it has no linguistic error.
4. **False Accept (FA)** and calulated by $FA = PFA + k : GFA$, where PFA Plain False Accept represent the number of utterances where the students response is correct in meaning but has a linguistic error, and the system accept it, and GFA represent the number of incorrect responses in terms of meaning or linguistic where the system accept it. The parameter k is set to 3 to make gross false accepts relatively more important.

According to the above four categories, the performance of the overall system is calculated by the Differential (D) score [10] which is mathematically defined by the following equation:

$$D = \frac{CR(FR + CA)}{FR(CR + FA)} \tag{2.1}$$

2.3 Baseline system

The introduced baseline system consists of main three components: DNN-HMM ASR developed using the Kaldi toolkit[13], text Processor and Grammar. The DNN-HMM ASR was trained using WSJCAM0 corpus [14] in addition to the 5,500 utterances that was used to train the whole system. Each frame is represented by 13-dimensional MFCCs features, also the delta and delta-delta coefficients were appended to construct a 39-dimensional feature vector. Moreover,Some context information were included by taking seven frames before and after of the current frame. Hybrid HMM-DNN model was used to create the acoustic model. The neural network consists of four hidden layers, where each layer includes 1024 neurons. The output layer represents a softmax layer, where each node denotes to posterior probability of the context-dependent HMM states. The following steps were used to train the DNN-HMM system:

1. Train the mono-phone GMM-HMM system from WSJCAM0 corpus [14] and Shared Task training data.
2. Alignment of the speech data using mono-phone model.
3. Train the tri-phone GMM-HMM system using the alignments obtained from step2.
4. Alignment of the speech data using tri-phone model.
5. Initial training of the DNN-HMM system using the alignments obtained from step4.
6. Fine-tuning phase to DNN-HMM system using shared task data.

BEEP dictionary[15] was used in the baseline system. The backed-off bi-gram language model was used and trained on the transcriptions of the shared task data.

The text processing component in the baseline system compare the answer of a certain prompt with a set of responses defend in the grammar which includes a group of templates for each prompt. The baseline grammar has 565 prompts, each one is a German text prompt with a set of possible responses to it. The ASR transcriptions is matched with a list of valid responses to be labelled as accept/language, accept/meaning, reject/language or reject/meaning. The base-system follow a simple assessment method which compare the recognized text with all possible responses and accept it if there is one possible response exactly matched with recognized text.

To train the whole system, 15 school classes participated to collect the data. Human annotators judge each response of the given prompt to decide if it should be accepted by the system or not. The training data includes 5,000 utterances, where the test data contains 996 utterances. The responses in the training data were recorded from both genders, and balanced across different age levels. Also, the background noises in the school environment introduce additional challenge to this task. The system has an WER of 14.81%. D score of the system is 1.694 on the training set and 1.0 on the test set.

2.4 Published systems for CALL 2017 shared task

Many researchers [1, 11, 16–18] introduced many enhancements to the baseline system to solve this shared task.

2.4.1 "The University of Birmingham 2017SLaTE CALL Shared Task Systems"

Mengjie Qian et al. [11] propose the following to enhance the baseline system:

- Using the AMI [19]and German PF-STAR [20] datasets in addition to Shared Task development data to train a hybrid DNN-HMM ASR system.
- Using tri-gram language model instead of bi-gram language model.
- They changed some parameters during the experiments. Fore example, they used 6 hidden layers neural network and 1024 neurons for each layer.
- Feature normalization and adaptation: They applied Cepstral Mean Normalisation (CMN) and feature-space maximum likelihood linear regression (fMLLR)[21, 22]. Because the training data of the shared task does not have any information about the speaker, each utterance was considered to be from a different speaker. First, the LDA was applied on 143-dimensional vector of MFCCs to reduce the dimensionality of the data to 40-dimensional features and decorrelate the classes from each other. After that, fMLLR transformation was applied on 40-dimensional features that extracted from the LDA stage.
- They enhanced the grammar in the text processing phase by including additional responses that were extracted from the transcriptions of the Shared Task.
- Adding two pre-processing steps before comparing the response with the grammar. The first step to delete some extra words like um and uh that may appear due to the uncertainty in the student answer. The second one to remove the words that repeated more than one time when the student tries to modify his answer.

After the proposed enhancement on the ASR component of the system,they achieved WER of 9.27% compared with the baseline CALL Shared Task DNN-HMM System which has WER of 14%. To insure that the parameters value will be the optimal for the test set. They employed the weighted summation fusion approach to take advantage of the multiple systems. The experiment showed that the expanding the Grammar XML file has improved the D score to 4.710 when system was applied on the test set.

2.4.2 "The CSU-K Rule-Based PipelineSystem for Spoken CALL Shared-Task"

Axtmann et .al[16] developed a system based on a pipeline of predefined rules. Also,they introduced many enhancements to the baseline system of the shared task:

- Improving the recognition rate by Appling (LDA+MLLT) and Speaker Adaptive Training (SAT) [8] technique to reduce the effect of speakers variation. Also, the did many experiments to determine the best number of gaussians. This achieved a WER of 13.05%(1.76% improvement).
- Replace the backed-off bigram language model of the baseline system with new interpolated trigram model which added the responses that defined in the grammar train the model. This achieved a WER of 10.72%. The language model returns a sentence score of log probability. This score can be used to measure the acceptance of the sentence syntax.
- Extending the phonetic dictionary to include some expected pronunciation. They followed different rules [23] that cover German mispronunciations in English.
- Some pre-processing steps were applied on the transcript including: remove all irregular white spaces, remove all the words that have no effect on the meaning and language correctness of the system like please at the start of the response, expanding the abbreviations (Im becomes I am) and remove duplicate words.
- Expanding the grammar by adding the correct answers in training data and generate new responses by substituting some words with its synonyms such as the word want was replaced with need and add new generated response.
- Creating a table for each prompt to indicate the words POS (Part of Speech) level for each answer of a certain response. The text of a new response is presented in terms of POS and compared with this table to find allowed words.
- Clustering the prompts into five clusters according to their POS level similarity. Then, the new utterances are matched to the words that carry the meaning and extracted from the same cluster.

Text processing component uses recognized text to accept or reject the user response based on rule-based system. This system achieved D score of 3.21 when was submitted to the competition and 4.79 after improving the system by variety of changes on the defined rules to accept or reject the response.

2.4.3 "Syntactic and semantic features for human like judgment in spokenCALL"

Magooda and Litman[17] enhanced the base-system by extracting syntactic and semantic features from the transcripts to measure the inconsistency on the language level and

on the meaning respectively. They extracted 10 features to detect if the response is linguistically correct or not:

- One feature to detect the spelling mistakes by using NLTK English spell checker [24].
- One feature to measure how much if the response is a real sentence or not, by using Stanford part of speech tagger [25].
- Extract eight features from multiple 5-gram language models.

To measure relatedness between the prompt which are defined in the grammar XML file and the speaker response, they used the following features:

- Two syntactic features to capture the matching between the user's response and the set of possible responses that were defined in the XML file. The first feature is the number of unigrams, bigrams and trigrams that matched with a set of possible responses. The second feature is the probability of the response text which extracted from a language model trained on all possible responses.
- Four semantic features to capture the relatedness between the user's response and the set of all possible responses that defined in the XML file. Each word in the response text was represented with a vector, where each dimension capture a semantic feature [26, 27]. The response vector is the summation of all vectors corresponding to each word. Then cosine similarity was used to measure the similarity between the response vector and the vector representation of each possible response defined in the Grammar file.
- One feature to indicate if the response length is reasonable or not. This feature can be computed by dividing the length of the student response on the average length of all possible responses.

Two classification techniques were used to evaluate the extracted features experimentally: K-nearest neighbor (KNN) [28] and Support vector machines (SVM) [29]. The system achieved the third position in SpokenCALL shared task competition and the best value of the D-score was 3.047.

2.4.4 "Using an Automated Content Scoring System for Spoken CALL Responses:The ETS submission for the Spoken CALL Challenge"

The performance of pre-existing automated content scoring system was investigated in [18]. This system extracts the features from word n-gram and use them to accept or

reject the responses in a spoken CALL task. Different types of features were explored to increase the performance of spoken CALL system:

- Features were extracted from automated content scoring system. These features were included in all experiments of the study and include: the logarithimic length of the response in term of its charachters, character 2-gram and 5- gram, token unigram and bigram and syntactic dependency features that extracted using ZPar tool[30].
- Features related to the prompt and prompt category: There are many prompts in the Grammar XML file does not include enough number of responses to train a robust model for each prompt. So, a single model was trained using the responses from all prompts and the prompt bias features were used to cover the information about prompt-specific grammar and vocabulary patterns.
- Features extracted based on the similarity between the user response and the set of possible responses that defined in the grammar XML file. These features include: Minimum word error rate (WER), average WER, Maximum WER and BLEU score. WER is obtained by calculating the number of operations(Add word,delete word or substitute word1 by word2) that required to convert the user response to the correct response which is defined in the Grammar file. The WER is calculated between the user response and each possible response defined in the Grammar file and then the maximum, average and the minimum were obtained. RELU metric is used to quality of a machine translation with respect to human translation [31].
- Features represent the grammatical errors: The language-check Python wrapper tool was used to check if the user response contains any grammatical errors.

Kaldi ASR output was used in all experiments to investigate the effect of the proposed features. Support vector regression was used to train several models to evaluate different sets of proposed features. The experiments showed that the features based on the similarity between the user response and the possible responses are more effective for this task. The system achieved a D score of 4.353 when the test data was applied on the trained model.

2.4.5 "Deep-Learning Based Automatic Spontaneous Speech Assessment in a Data-Driven Approach for the 2017 SLaTE CALL Shared Challenge"

A deep learning based approach was proposed by [1] to evaluate the grammar and semantic errors of the user response. Fig2.1 explains the proposed approach. Three

versions of ASR system were investigated: Kaldi, Nuance and SLaTE2017. Kaldi and Nuance were developed by the spoken shared task organizers and released as a base-system. SLaTE2017 ASR was developed by the authors for this challenge and it has the best performance (compared with Kaldi and Nuance) by adapting the acoustic model to the training data of this task. The output of each ASR system is provided to the feature extraction phase. Two main kinds of features have been extracted from the output text:

- Features to extract information related to the meaning in the user response(9 features from each ASR output text): Four features were extracted from four language models, three features were extracted using sentence-embedding approach [32], two features to measure the similarity between the the user response and all possible responses were extracted using word -embedding approach[33].
- Grammar features: The user response is parsed into dependency tree to identify incorrect grammar. Two grammar features were extracted from an input text by computing the log-probability from 3-gram language model. Twenty grammar features from the parsed text to do a grammar check using three language models: the first one includes correct distribution, the second includes incorrect distribution and the last one trained on the raw text of the shared task.

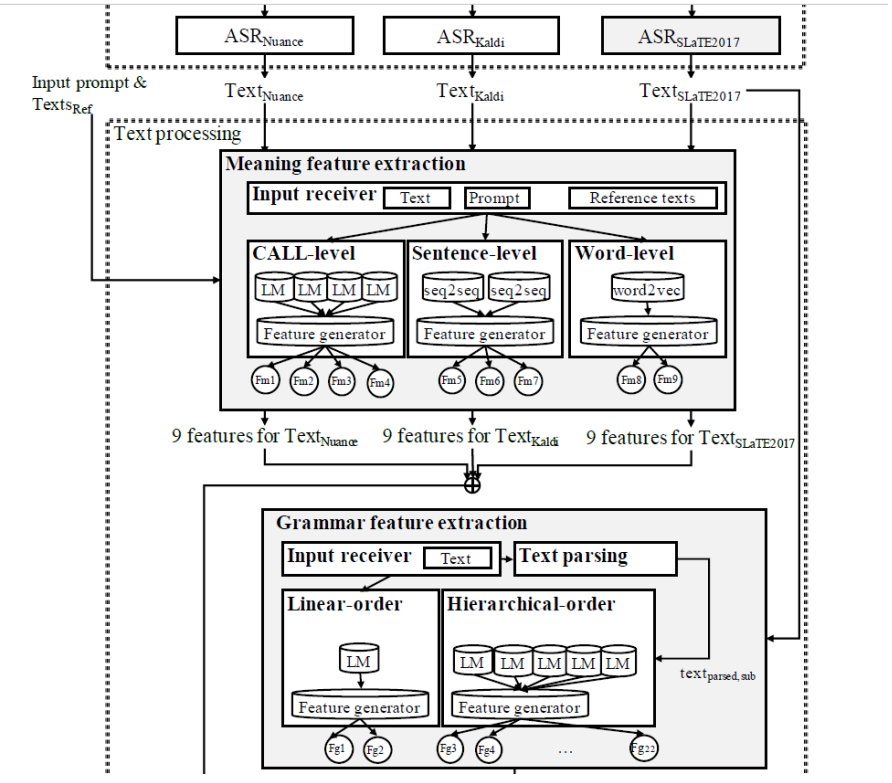

FIGURE 2.1: The deep learning based approach for spoken shared task [1] .

Deep Neural Network(DNN) with four hidden layers was used to evaluate the grammar and meaning features.The output layer is a softmax layer which give accept or reject decision. The experiments showed that the D score is 4.37 when the proposed method was evaluated on the test set.

Table2.2 summarize the main contributions for five approaches that published at SLaTE 2017 conference.

Research Paper	Main Contribution	D score
"The University of Birmingham 2017 SLaTE CALL Shared Task Systems"	Improving the grammar file, improve DNN-HMM System	4.710
"The CSU-K Rule-Based Pipeline System for Spoken CALL Shared Task"	Table for each prompt to indicate the words POS (Part of Speech), Rule-based system for classification	3.21
"Syntactic and semantic features for human like judgement in spoken CALL"	extracting syntactic and semantic features from the transcripts to measure the inconsistency on the language level and on the meaning respectively	3.047
"Using an Automated Content Scoring System for Spoken CALL Responses: The ETS submission for the Spoken CALL Challenge"	Extract Features from automated content scoring system, Features related to the prompt and prompt category and Features extracted based on the similarity between the user response and the set of possible responses that defined in the grammar XML file	4.353
"Deep-Learning Based Automatic Spontaneous Speech Assessment in a Data-Driven Approach for the 2017 SLaTE CALL Shared Challenge"	Extract 27 features to represent the meaning of user response in addition to 22 grammar features. Also, DNN based approach was used in the classification	4.37

TABLE 2.2: Comparison between shared task papers published at SLaTE 2017 conference.

Two approaches to measured the semantic relatedness were introduced by [2]. In the first approach, a weighted and directed semantic network was created, where the WordNets words and synsets were used as the nodes of the network. Each word is represented by the node in the network and the synsets of that word are connected to it by an edge. Also, each synset was connected to group of predicate arguments.

Fig2.2 shows a part of the network for the word car. The word "automobile" is the synonym of the word car. Moreover, automobile is connected to other synsets by extracting the semantic relations that included in WordNet.

The second approach represents the context of a certain word by a vector, where each dimension in that vector is the frequency of co-occurrence of the context words. If a

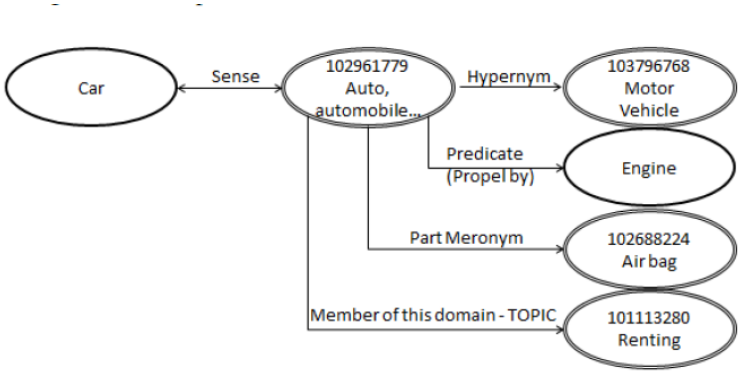

FIGURE 2.2: Semantic network example for the word car [2] .

phrase consists of number of words, the vector of phrase is computed by summation of its word vectors. Cosine similarity is used to measure the similarity between two vector.

Paragraph Vector approach was proposed in [34] to measure the semantic relatedness between two sentences, where each sentence or document is represented by a vector. Sentences with similar meaning have similar position in the vector space. Experiment results showed that this algorithm represented the text efficiently and achieved competitive results in text classification and sentiment analysis tasks.

Sentence embedding model was proposed in [35] to extract a semantic vector for each extracted word by using recurrent neural networks (RNN). This approach is more suitable to measure the semantic relatedness between two text strings. It takes the the relationship among words in a certain sentence to encode its semantic meaning.

A supervised learning approach was used in [9] to expand the grammar in Spoken CALL system. The annotated responses in the training data were used to add new responses to Grammar component of the system. First, a simple expanding method was followed by adding the correct responses in the training data. Then, a new approach was proposed to look for a more general way to add new responses using correct and incorrect samples.

2.5 Published systems for CALL 2018 shared task

Following the success of the first shared task with 20 submissions from 9 participant teams, the second edition with new resources and updated training data was announced in October 2017 and the test data was released in February 2018 [12]. Similar to the first edition, the task organizers provide the audio data, ASR outputs, and reference response grammar. There are two tasks: the text task where the ASR outputs for the spoken responses are provided by the organizers, and the speech task where participants

can use their own recognizes to process audio responses. For the second edition, a new subset was provided by task organizers. It consists of 6698 student utterances and was selected in a similar way to the first edition of CALL task. All wav files were processed through the two ASRs that achieved best results in the first shared task [1, 11] after cleaning the transcriptions at the University of Geneva.

Five of the participants in the 2018 CALL shared task [7, 36–39] presented their systems in the Interspeech 2018 conference which was held in 2-6 September 2018 in India. They introduced different ideas for improving to the baseline system at both the ASR and the text processing stages. In general. The worst submission in the second edition achieved better score than all submitted entries in first edition. The score of the best entry is (D = 19.088) compared with the baseline (D = 5.343).

2.5.1 "Liulishuo's System for the Spoken CALL Shared Task 2018"

The best D score (19) among the participating teams in the 2018 shared task, was achieved by Huy et. al. [36]. They improved the performance of the baseline speech recognition system provided by the shared task organizers. They developed a set of features to capture the linguistic and semantic meaning of the responses.

54 features were extracted from the transcription and include:

- All syntactic and semantic features proposed in [17].
- Language model scores: 14 language model were trained based on words from transcription and their part-of-speech (POS) tags. Also, CoreNLP [10] tool were used to train language models based on syntactic parsing results.
- Numbers of features represent the matching between the response and its references.
- different word embedding were trained and the maximum similarity value between the response and its responses is computed.
- Latent Dirichlet Allocation algorithm was used to learn a topic model. The minimum similarity value between the response and the topic distributions is considered.
- The number prompt words that does not exist in the response and then normalized by the its length.
- the number of grammar error when the tool 'our inhouse grammar' [40] was applied on the response.

Four machine learning were used and compared between based on the results of the trained models. Also, a VotingClassifer was used to do a soft weighting of probability outputs from each model. The classification results were optimized for various factors (training set, n-best hypotheses of speech recognition, decision threshold, model ensemble).

2.5.2 "Improvements to an Automated Content Scoring System for Spoken CALL Responses: The ETS Submission to the Second Spoken CALL Shared Task"

Keelan et. al. in [37] used additional features extracted by comparing the input response to language models training on text written by English native speakers and L1-German English learners. In addition, they developed a set of sequence-to-label models using bidirectional LSTM-RNNs with an attention layer. The RNN model predictions were combined with the other feature sets using feature-level and score-level fusion approaches resulting in a best-performing system that achieved a D score of 7.397.

2.5.3 "An Optimization Based Approach for Solving Spoken CALL Shared Task"

In this paper [7], the text processing module is implemented as a rule-based, where optimized using the genetic algorithm. This system achieved D score of 14.4 in the 2018 spoken call shared task. Section 6.4.2 describe this system in details.

2.5.4 "The University of Birmingham 2018 Spoken CALL Shared Task Systems"

In this paper [38], authors proposed many improvements to the baseline system. They enhanced both components: automatic speech recognition and text processing unit. Regarding to ASR component, Long short-term memory (LSTM) network was used instead of DNN network, where the LSTM network was trained using the alignments that obtained from DNN-HMM system. Regarding to text processing, different methods were used to calculate the similarity between references and response. The Word Movers Distance (WMD) [41] was used to calculate a sentence-level distance between response and its references. Also, a two-class classifier was used to take the decision.

2.5.5 "The CSU-K Rule-Based System for the 2nd Edition Spoken CALL Shared Task"

A rule-based system was proposed in this paper[39]. This system predicts the judgment for grammars and meaning of the responses based on pipe-lined rules. First Doc2Vec [42] was trained using the training and all reference responses. Also, they enhanced the grammar by deleting any detected errors. They looked at meaning and grammar errors separately. Each response was judged in terms of grammar and meaning. Then, the final decision was taken based on a threshold value.

Chapter 3

Optimization Techniques Background

In the simple words, the optimization is maximizing or minimizing a real function by changing the input variables of a certain problem. The optimization targets to find the best values for some predefined objective functions. The most of optimization algorithms consider the problem as black box. In this case, the optimization algorithm does not require the mathematical model of the problem, because it only changes the inputs variables to maximize or minimize the outputs of the system.

Optimization algorithms are usually classified into evolutionary or swarm intelligence-based algorithms [43]. The optimization algorithm can be used to generate one solution and improve it over its iteration. This form is called individual-based algorithm. In the second form, the algorithm generates many solutions and enhances them during the iterations of the optimization.

3.1 The genetic algorithm (GA)

The genetic algorithm mainly depends on three operators to generate high-quality solutions: mutation, crossover and selection.

3.1.1 Crossover

This operator takes two solutions (called parents) to generate another two solutions (offspring) by inheriting the genes from both parents. For example, the offspring may

take 30% of its genes from the first parent and 70% from the second parent, where the crossover points are randomly selected.

3.1.2 Mutation

The function of this operator is selecting some genes and changes its values. There are many forms of this operator depending on the chromosome representation. For example, if the chromosome is encoded as a list of binary digits, the mutation operator can be applied by flipping the bit value of some selected genes. However, if the chromosome is presented as a list of integers, we can not apply the binary mutation on the selected genes. In this case, we should find another form of mutation such as performing permutation operator on the selected genes.

3.1.3 Chromosome Representation and Evaluation

The representation of the chromosome basically depends on the problem to be solved. For example, it could be represented in binary as list of of 0s and 1s in some problems. A suitable representation should be found after studying the problem domain. The chromosome should be well formulated as a good representation will enhance the search process. The mutation and crossover operators are heavily depend on the chromosome design.

3.1.4 The basic procure of genetic algorithm

Genetic algorithm 1 starts with a population which is defined as a set of initial solutions. The chromosome is one solution in that set. Two parents from the population are selected to form the offspring (new solutions) using crossover and mutation operators. If the fitness value in the offspring is better than the value in previous iteration, the

algorithm updates the fitness and keeps with the solution which has better fitness value.

Algorithm 1: The main procedure of the Genetic algorithm. Details in [44]

Input crossover probability
mutation probability
maximum number of iterations (MAXITERATIONS)
Initialize current iteration=0
Create initial set of solutions (population)
population=GenerateRandomSolutions();
fitness=Evaluate(population)
initialize bestSolution = the solution with maximum fitness;
while *current iteration* < *MAXITERATIONS* **do**
    random1=GenerateRandom(0,1)
    if *random* <*crossover probability* **then**
        select two parents: p1,p2 = SelectOperator(population)
        $p1_{new}, p2_{new} = DoCrossover(p1, p2)$;
    end
    random2=GenerateRandom(0,1)
    if *(random2*<*mutation probability* **then**
        offspring=DoMutation($p1_{new}, p2_{new}$)
    end
    if *fitness* <*Evaluate (offspring)* **then**
        fitness =Evaluate (offspring)
        bestSolution=offspring;
    end
end
Output decision

3.2 Swarm intelligence-based algorithms

The basic procure of Swarm algorithm is:

1. Create initial set of solutions (population).
2. While stopping criteria not satisfied:
3. Evaluate each solution (particle)
4. Get the best solution based on the fitness value

5. Move each particle towards the best

6. Return the best solution.

Chapter 4

Automatic Speech Recognition ASR- Background

Automatic speech recognition (ASR) is converting the captured audio signal to underlying textual representation. ASR is very natural interface for human communication and you can obviously interact with machines without needs for mouse or keyboard to do basic tasks such as searching using speech, controlling simple devices and interacting with intelligent devices.

Fig4.1 show the basic components of ASR system. The first step is transforming the analog acoustic signal to its digital representation. This digital signal is moved to feature extraction phase to find set of parameters that keep the most relevant information. Acoustic model establishes the association between the acoustic information and phonetics through training process. The language model component holds the structural constraints in the language and it basically brings the probability of existence a certain word after a word sequence. Finally, pattern classification component is responsible for comparing the testing data with each class generated by the model and computing the similarities between them. Speech recognition is considered as machine learning problem, where some input of labeled data are provided to build the acoustic model. This model will be used to recognize new examples of unlabeled data.

The main challenges of ASR system are phonemes co-articulation, and the diversity in the pronunciation of some phonemes caused by existing of different dialects. Phonemes co-articulation occur because each sound in the word is affected by the sounds that come before and after. Also, the input data to ASR system is variable-length sequence and this is more difficult than static data such as images, so the unique features of speech signal that resides in its temporal dimension should be considered. Moreover, building an ASR system requires various resources to train the acoustic model from large data.

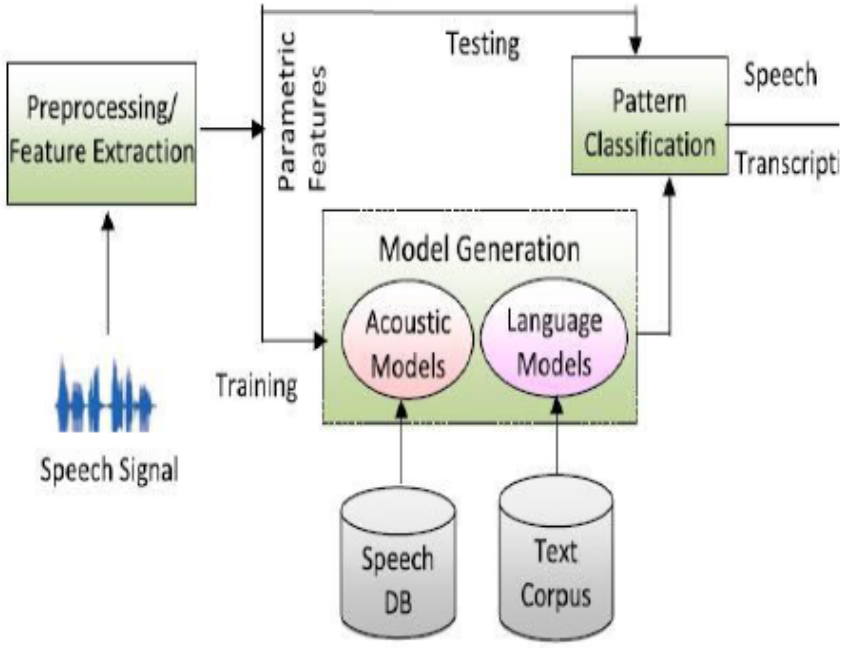

FIGURE 4.1: The basic components of ASR system [3].

4.1 Feature Extraction

This phase convert each speech frame into set of informative features that will be the input to the acoustic model. The most used techniques to extract the acoustic features are: MFCC and Fbank[45].

4.2 Phonetic Dictionary

The Phonetic Dictionary provides the phone level transcription for each word. It used in the training phase to train the acoustic model for different phones. Also, it is used in the recognition phase to find the possible sequence of phones that form the word.

4.3 Language Model

Language model is concerned with guessing the most probable word coming after another certain word. In other words, after the sequence of recognized phones are converted to a sequence of words by acoustic models and phonetic dictionary, the language model tests the probability of coming after the previously guessed word. Based on the highest probability of those nominated by the pronunciation dictionary, the best candidate is picked up as the next guessed word.

4.4 Acoustic Model

The acoustic features extracted from speech are then used to build an acoustic model for each phoneme (mono-phone) or each tri-phone. This model should be able to accommodate the variation in pronunciation of each mono-phone or tri-phone.

4.5 Hidden Markov Models (HMMs) as the Starting Point in Speech Recognition

HMM is a supervised machine learning algorithm used to discover unknown states by observing a sequence of frames. HMM is specified by the parameter $\lambda = (A, B, \pi)$ where A is the transition likelihood matrix, B is the observation likelihood matrix which provides probability of observation given the state and π is the matrix of initial state probability [4]. Fig 4.2 demonstrates a general hidden markov model where the X symbol denotes for the hidden states. The aij symbol represents the probability to move from state Xi to state Xj and bij represents the likelihood of observing a symbol Yj given state Xi and so on.

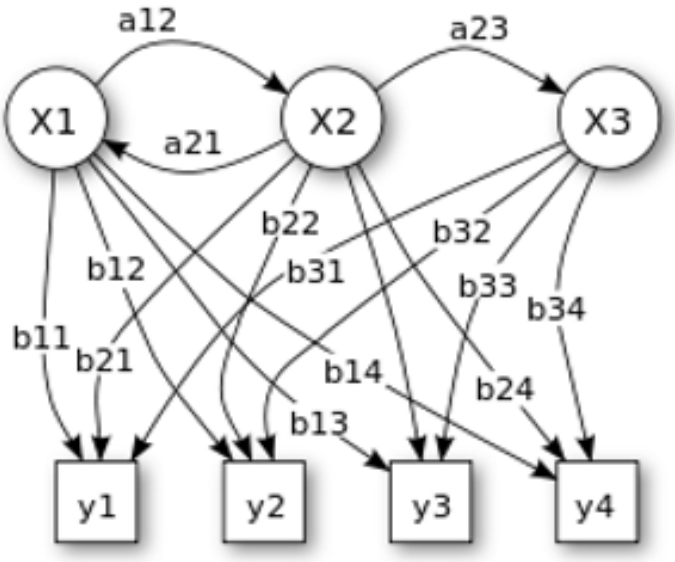

FIGURE 4.2: The general hidden markov model [4].

Hidden Markov Models usually used for solving three types of problems:

- Problem 1. With Known $\lambda = (A, B, \pi)$ and an order vector of observations Y, we can solve $P(Y|\lambda)$. This problem often used to test new vector of observation after

creating the model so that we can find how well it match with the created model λ.

- Problem 2. With Known $\lambda = (A, B, \pi)$ and an order vector of observations Y, we can find the optimal sequence of states to generate observations Y.
- Problem 3. With known vector of observation Y, number of states and number of observation symbols. We can find model $\lambda = (A, B, \pi)$. This problem often used to train the model to fit the sequence of observations Y.

Hidden Markov Models (HMMs) is considered the starting point in speech recognition. It uses mixture of Gaussians to model the speech signal. Many researchers in [46–49] have been built the ASR system based on HMM and completed a successful application using large vocabulary. Young [46] has discussed the main components of large-vocabulary recognition systems based on HMM model. Champion et al. [48] have discussed the problem of appropriately including dynamic information into the acoustic model using continuous state Hidden Markov Models. Srinivasan [49] has recorded the acoustic signal using wave surfer tool where the test data was compared with the trained data using Hidden Markov Model. However, she did not use a benchmark dataset to evaluate the performance of the algorithm. The HMM-based ASR systems estimate the likelihood of each phone and then recognize speech by converting each word in the vocabulary to a sequence of phonemes. Each phoneme is modeled as a series of HMM states. The emission probabilities are computed by using traditional Gaussian mixture models. To train HMM, the number of hidden states is constant and it is not necessary to equal the number of the states in the source signal[50].

HMM is proposed by many researches in context of Arabic speech recognition system. Al-Otaibi [51] use HMM method to build a new technique for labeling Arabic speech. The proposed technique accomplished a recognition rate of 93.78% for speaker dependent Arabic ASR system.

The major advantages for using HMM in speech recognition systems are that the mathematics of HMM are well expressed and its ability to model time-varying spectral vector sequences in effective way. The limitation of standard HMM is the lack of lawful correlation between the acoustic frames because each frame depends only on one state and all adjacent observation frames are independent to that frame. This make the problem of handling stationary strongly correlated frame harder[52].

4.6 Artificial Neural Networks in Speech Recognition Systems

Neural networks (NN) consists of interconnected components called neurons that represent very simple processors. NN contains many layer, the first layer is called Input Layer which is composed into neurons, and the last layer called output layer which is also composed into neurons. The other layers are lie between them and called hidden layers. Many researchers have recommended the use of neural networks (NN) for speech recognition due to its performance on complex real applications. Learning through neural network follow a hierarchical architecture to process the speech signal. This architecture includes many layers of non-linear transformation. The key feature of using neural network is the possibility of applying this algorithm without a previous knowledge of the speech process as it can be trained using the input data directly and generates the output words.

4.6.1 Static Neural Networks

Most of the recent researches show the significant improvement of the artificial neural networks in speech recognition systems. However, it needs long time to train especially with using many hidden layers. Evolutions in computing hardware is the main factor of employing these techniques to model the acoustic speech [53].

There are mainly two types of Neural Network: Shallow architecture and Deep architecture. Shallow architecture Neural Network (e.g., with one hidden layer) needs large amount of labeled data to train the model. On the other hand, increasing the number of hidden layers requires less amount of labeled data but the local optimization algorithms like back-propagation algorithm will have poor performance when used to train this type of network. This behavior due to the fact that the probability of stopping at local optimum points will increase. By contrast, shallow architectures usually employ a convex loss function which allows these architectures to optimize the initialization parameters efficiently[54].

Convolution neural network (CNN) uses the technical operation of convolution to search for a particular pattern. It consists of many component layers: Conventional layer, Rectified Linear Unit (RELU) layer and pooling layer. The main goal of Conventional layer is to extract features from the input sequence of data. The output of convolution layer is connected to RELU layer which allows the network to be properly trained without decaying of the gradient through back-propagation. The pooling layer is mainly used for dimensionality reduction with keeping the most important information in the features.

Most of the researches that use neural network can be classified based on the architecture used into three main classes:

1. Generative deep architecture: the purpose of this architecture is to differentiate the observed data. In this architecture, deep learning works like other dimensionality reduction techniques such as Principle Component Analysis (PCA)[54]. There are many researchers [55, 56] used generative models in deep learning. Hinton et al. [55] use feedforward neural network to generate posterior probabilities. They use new technique consists of two-stages to train DNN with many hidden layers. The first stage includes initialization of feature detectors layers by fitting a stack of generative models that are trained without using any information about the HMM states. The second stage includes initialization the hidden layers, where each one of the hidden layer is initialized using a generative model. This proposed technique outperforms Gaussian mixture models on many benchmarks of speech recognition, but it is a computationally expensive approach compared with Gaussian mixture models and needs more hardware capabilities. Deng et el. [56] proposed technique for automatic discovery of good representations for speech for scalable speech recognition. They used a generative deep architecture where each layer is completely linked to the layer below and the weights are pre-trained by using contrastive divergence approximation. The result was proved using TIMIT database. However, the proposed encoding technique did not cover the overlapping propriety of the speech signal.

2. Discriminative deep architecture: in this architecture, posterior probabilities of the classes are characterized. Some variants of neural networks are dominant discriminative models and many researchers [53, 57] uses discriminative architecture with the back-propagation algorithms for speech recognition. Ossama et al. [53] modify the convolutional neural network (CNN) and improving the way of modelling the speech features by adding limited-weight-sharing scheme. They improve error rate by 6%-10% compared with other deep neural networks on the TIMIT phone recognition. CNNs have been used in speech recognition before [57], but the operation of convolution was applied on windows of frames where the windows are overlapped to learn more stable acoustic features. Even the error rate was improved, CNN is computationally expensive and needs large amount of data for training.

3. Hybrid deep architecture: in this architecture, the output of the generative model is provided to a discriminative model. Mitra et al.[58] apply deep learning for speech recognition under noisy and channel degraded conditions. They reduced

the word error rate (WERs) by using robust features and they compared the result with baseline mel-filterbank features.

Over-fitting is a serious problem in deep neural networks (DNNs) when it use large number of parameters to train the model for large vocabulary speech recognition [59]. Over-fitting occurs when the neural network fits the training data very well, but it cannot generalize the model to many new examples. There are several methods used to avoid over-fitting: Regularization, Cross Validation, Pruning and Early Stopping. Cross-validation is basically utilized when the objective of the experiment is prediction, and we need to approximate how accurately a created model will be reflected in practice. For example, 1-round of cross validation includes dividing a data set into two separated subsets, one of them used in training stage and it is called training data and the second used in testing phase and it is called testing data [60]. One of the regularization methods is Dropout[61]. During training the neural network (NN), dropping technique drops units from the NN to avoid the case of co-adapting. Srivastava et al.[61] show that this technique introduce enhancement on the performance of neural networks on supervised learning tasks.

4.6.2 Dynamic Neural Networks

There are main two types of dynamic neural networks commonly used in speech recognition tasks.

1. Recurrent Neural Networks: It provides an excellent modelling to sequence data such as speech and text. In this type of neural network, the output of an activation layer is provided as input to preceding layers. That means you can sometimes get back to where you started by following the feed-backs. It is a dynamic network because its depth is not fixed and depends on the size of the observed data sequence. Many researches [62–65] used Recurrent Neural Networks in the speech recognition context. However, RNNs suffer from well-known problem which is called vanishing gradient, where the gradient decay through back-propagation process and become smaller for early layers. It is a fundamental problem because the early layers are responsible for detecting the simple patterns and if they get wrong, the result built up by the network will be wrong. There are several ways to address this problem. Gating [66, 67] is the most popular way that helps the network to decide when to forget the current input, and when to remember it for future time steps.

2. Echo State Networks: it developed by Jaeger [68] based in the structure of recurrent neural networks. The key idea of this network is not to train the hidden to

hidden connection at all, but to just fix them randomly and hope that you can learn sequences by just training the effect the outputs. This has strong similarity with old ideas about perceptrons that fix input-hidden and hidden-hidden connections at random values and only learn the hidden-output connections. So, the learning is very simple and fast if the linear model at output units are used. It is essential to set the random connections wisely so the network does not explode or die. Many researchers [69, 70] used Echo State Networks in the speech recognition context. Hmad and Allen [69] introduced their work of Arabic phoneme recognition using an Echo State Network. They used two feature extraction methods: Cepstrum Coefficients (MFCCs) and Linear Predictive Code (LPC) and they compared between them through experimental results. The accuracy of the system was 72.3% when it was evaluated using 6 speakers from KAPD dataset and 38.2% when 34 speakers from CSLU2002 dataset were used in the evaluation process. However, the work was evaluated using single dialect dataset, but it is recommended that the database for speech recognition purpose should include multiple dialects. Triefenbach et al. [70] used a deep architecture of echo state network for acoustic modelling and they produced Phone Error Rate (PER) of 23.1% on TIMIT speech dataset. However, they did not test their work on a Large Vocabulary Speech Recognition datasets.

Table 7.5 compares between RNN,ESN and CNN with respect to the time needed to train the algorithm, difficulties of training process and the type of the algorithm.

Neural network	Time for training	Training(easy/hard)	Dynamic/static
RNN	computationally expensive	difficult to train(vanishing gradient)	dynamic
ESN	fast with less computation (only learn the hidden-output connections.)	Simple and linear but set the random connections wisely	dynamic
CNN	computationally expensive and needs several resources like GPU	Usually needs large amount of data for training	Static

TABLE 4.1: Brief comparison between RNN,ESN and CNN

4.7 Towards End-to-End ASR with Neural Networks

This architecture aims to convert a sequence of frames of acoustic signal $X = \{x1, x2, x3, ...\}$ into its corresponding text $Y = \{y1, y2, y3, ...\}$ by learning a probabilistic model p(y/x).

Fig4.3 shows that this architecture is end-to-end because it collapses the language, pronunciation and acoustic models into one big probabilistic model which directly transcribes the sequence of speech frames into corresponding text, without requiring lexicon or language model. There are many response for going towards End-to-End architecture. First, the process of creating pronunciation dictionary requires significant human effort. Also, this architecture use different optimization function (sequence-level transcription) to train the network. Moreover, the training of hybrid approach (DNNs with HMMs/GMMs) by training each module independently with dissimilar criteria may not be optimal way [71]. You can find other reasons in [72].

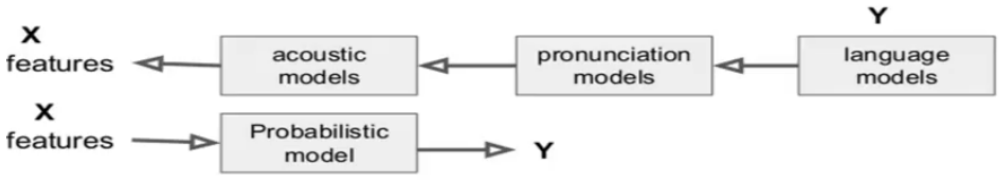

FIGURE 4.3: End-to-End ASR with Neural Networks.

End-to-End style has been effectively exploited for speech recognition [71, 72]. Graves and Jaitly [72] proposed End-to-End ASR system using a bidirectional LSTM architecture [73] . Connectionist Temporal Classification objective function was used but with some modification to minimize the loss function. Without using any prior linguistic information, a word error rate of 27.3% was achieved using Wall Street Journal corpus. Whereas the result is impressive, the training speed of RNNs/LSTMs can be very slow especially if the input sequence is very long and this architecture suffer from gradient vanishing problem. Zhang et al. [71] propose an end-to-end speech structure based on combination between CNNs and CTC. TIMIT dataset was used to test the system and the results are similar to those achieved by multiple layers of LSTMs. However End-to-End architecture usually provide superior performance only with existing very large amounts of training data.

Since recognition systems based on HMM/GMM model suffer from many limitations,Hidden Markov Model (HMM) and Artificial Neural Network (ANN) are combined to take the advantages from both algorithms to enhance the performance of ASR system. This hybrid architecture mainly uses ANNs to estimate observation probabilities that represent the basic parameters for HMMs. Mohamed [74] proposed his work based on this architecture where the features were extracted using MFCC technique. He has significantly enhanced acoustic models by 2% with using Deep Neural Networks (DNNs) and 2.4%with using Convolution Neural networks (CNNs). Morgan et al. [75] emphasis on the hybrid HMM/ANN method which has been applied to large vocabulary recognition system. Whereas this approach was used intensively, training of deep neural network still depends on Gaussian mixture models to get frame-level labels. Building these models

need multiple stages, and each stage includes many feature processing techniques. So, recent researches attempt to create the acoustic model without any middle components.

4.8 Adaptation tools used for AM

There are two main variation in the context of ASR: speaker variation and environment variation. Both type of variations causes a mismatch between the training data and testing data, especially if the AM is trained on a native speaker dataset or trained on data recorded in free noise environment. To deal with these variations, different adaptation tools can be used to adapt the features or the model parameters using specific amount of adaptation data.

Different adaptation techniques are available at the AM including:

- Maximum A Posterior (MAP) It takes advantage of prior knowledge to add limitation on the parameter deviation.
- Maximum Likelihood Linear Regression (MLLR): the main goal of this techniques is to make the ASR robust to speaker variability. It adapts the model parameters by estimating the linear transformations on these parameters to maximize the likelihood of the adaptation data.
- Feature-space Maximum Likelihood Linear Regression (fMLLR): It applies a set of linear transforms of an acoustic space to maximize the probability of test data given the speaker independent model.
- Linear Discriminant Analysis (LDA): it is feature selection method in speech recognition. It is used to reduce the dimensionality and maximize the separability among HMM states. LDA uses the information from each HMM state to create new dimensions so that it maximizes the separation between the mean of each HMM state, and then project the data to these dimensions.

4.9 Kaldi toolkit

Kaldi toolkit[13] is one of the most common tools in speech recognition and it is used in many published researches. It is an open source tool and written in C++ programming language.

4.10 Word Error Rate (WER)

WER measure is used to evaluate the performance of the speech recognition system. It can be computed by the equation4.1

$$WER = \frac{S + D + I}{S + D + C} \tag{4.1}$$

where

- S is the number of substitutions that required to convert the recognized text into reference text,
- D is the number of deletions that applied on recognized text to covert it to reference text, I
- I is the number of insertions that required to convert the recognized text into reference text,
- C is the number of the corrects

The SVM classifier is widely used in many applications due to its high performance

Chapter 5

Introduction to Machine Learning Methods

There are two main types of classification: Generative classification which follows probabilistic approach to determine labels for new points like Bayesian classification. The second type is discriminative classification which discriminates the classes from each other by a line or curve.

5.1 K Nearest Neighbors(KNN)

KNN classifier is a non-parametric classification technique, it's a Lazy classification. In the testing stage, the new data will be classified by the closest class, the closest is measured by taking the distance which could include hamming distance and Euclidean distance between each class and the sample data [76]. Even the simplicity and easiness of the implementation, this algorithm needs large memory because it compares with all of the training data.

5.2 Neural Networks (NN)

Neural networks (NN) consists of interconnected components called neurons that represent very simple processors. As shown in fig5.1 it consists of many layer, the first layer is called Input Layer which is composed into neurons, and the last layer called output layer which is also composed into neurons. The other layers are lie between them and called hidden layers. Many researchers have recommended the use of neural networks (NN) for many machine learning tasks due to its performance on complex real applications.

Learning through neural network follow a hierarchical architecture to process the input features. This architecture includes many layers of non-linear transformation. The key feature of using neural network is the possibility of applying this algorithm without a previous knowledge of the details of the application as it can be trained using the input data directly and generates the output class.

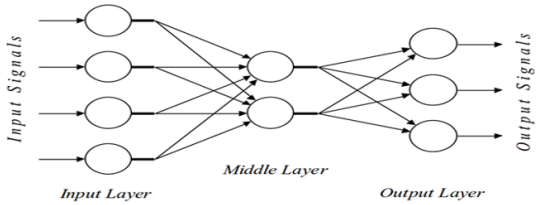

FIGURE 5.1: The structure of neural network

5.3 SVM

SVM is one of state-of-the-art classification techniques introduced in 1992 [29]. The SVM classifier is widely used in many applications due to its high performance.

5.3.1 SVM concept

SVM is a discriminative classification method which separate the data by maximizing the margin between two classes. Figure 5.2 shows a group of points distributed on two classes. Many linear discriminative classifiers can be used to separate the two sets of data. Figure 5.3 shows that three lines can be used to separate the red points from the yellow points. SVM takes the line that maximize the margin between it and nearest data points from each class. To show the idea, we can draw around each line a margin of some width, up to the nearest point .

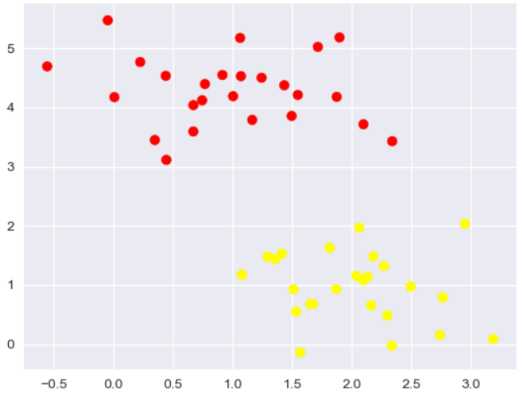

FIGURE 5.2: Example of data with two classe

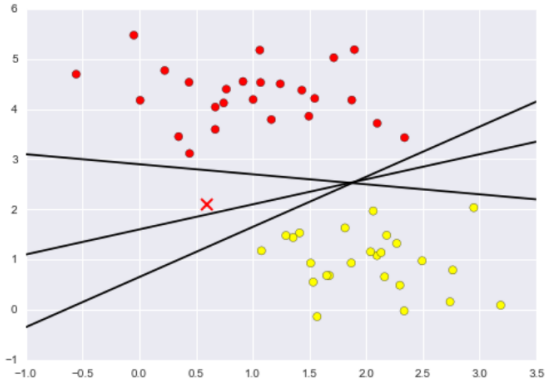

FIGURE 5.3: Three classifiers can be used to separate the two sets

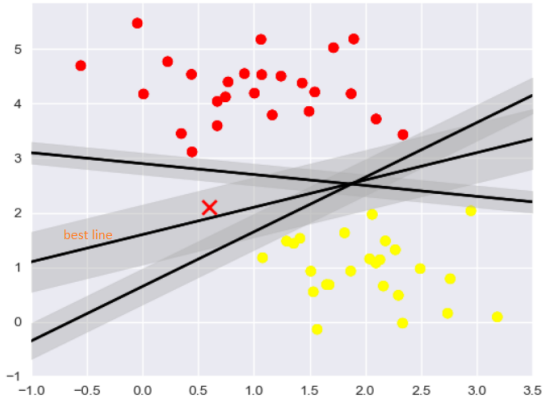

FIGURE 5.4: Three classifiers can be used to separate the two sets

5.3.2 The concept of support vectors

So, support vector machines choose the line that maximizes the margin and consider this line as the optimal model. Figure 5.5 show the best classifier for these training points. There are three training points just touch the margin and it is circled in figure 5.5. These three training examples are considered the pivotal elements of the model and known as the support vectors. These points make the SVM a successful classifier; because only the position of these points affect on the position of the model. The other points do not modify the position of the model! do not change the cost function used to fit the model. This make SVM works well with small number of training example [77]. The left part of figure 5.6 is a model trained on 60 examples and the right part shows that the model was not changed when trained on 120 examples because the position of the support vectors points did not change by increasing the number of training examples.

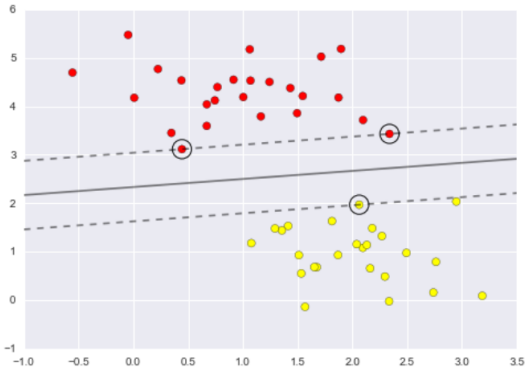

FIGURE 5.5: Best classifier that maximize the margin

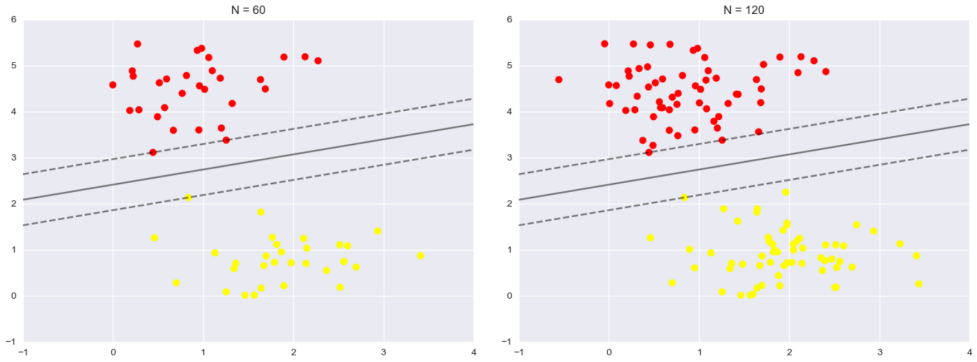

FIGURE 5.6: The effect of number of training examples

5.3.3 SVM with non linear kernals

In some cases, the linear kernel can not discriminate the training data into two separate sets. For example, the training data in figure5.7 can no be separated using a linear kernel. One way to solve this issue is projecting the data points into a higher dimension space such that a linear kernel can be separate the two sets. Fore example, we can compute a radial basis function of these data points. Figure 5.8 show the application of a radial basis function on the training points in figure 5.7. After projection of the data into three dimensional space, we can apply a linear kernel to separate these data.

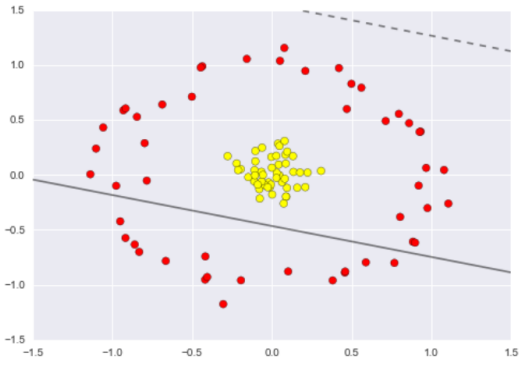

FIGURE 5.7: Linear classifier can not be used to separate the two sets

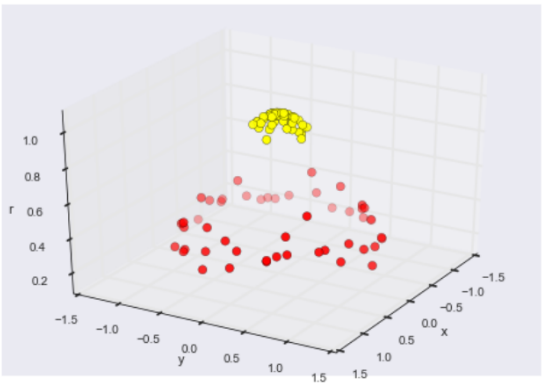

FIGURE 5.8: Radial basis function on the data points

5.3.4 SVM with overlapped data

In come case the data set is not clean and no perfect decision boundary may exist because the existence of . overlapping between two classes as show in the figure 5.9. To handle this case, the fudge-factor in SVM implementation can be used to softening the decision boundaries and allows to include some points inside the margin. This factor is tuned by C parameter. When increasing the value of C parameter, the margin is hard, and no points can be included inside the margin and vise verse. Figure 5.10 shows the effect of C parameter on the trained model. When the When the value of C parameter is 10, the margin is hard. But when the value of C parameter is 0.1 , some points were included inside the margin.

FIGURE 5.9: Overlapping between classes

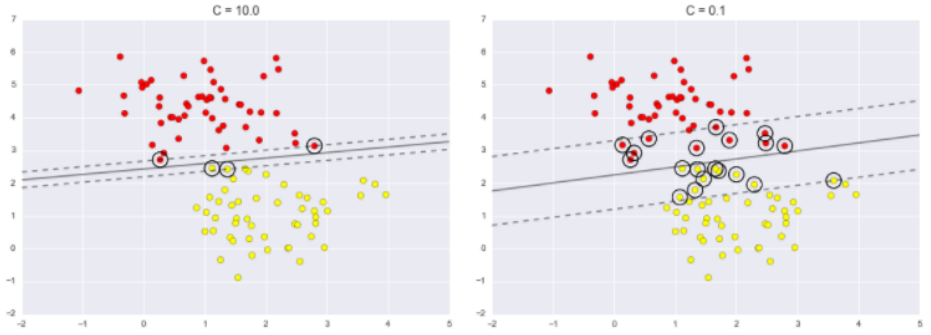

FIGURE 5.10: The effect of C paramter

Chapter 6

Methodology

In order to achieve our research objectives, We propose three approaches to solve the spoken CALL Shared task. The first approach is rule-based, which takes a final decision, reject or accept, about the given response by passing the audio transcription given by ASR through a sequence of pipelined stages and rules. Each rule checks if the response has a language error or not. If a rule can not detect any errors, it passes the response to the next rule. In the second approach, the genetic algorithm was combined with first approach to tune the parameters and thresholds used in each rule. The third approach is a machine learning model which predicts the final decision (reject or accept). Each proposed system computes the similarity between the user response and a set of reference responses in different way. The set of reference responses are available with the dataset. For each prompt, human experts defined a set of possible correct responses in different forms. Those are used as reference responses in our systems.

6.1 Pre-processing

In all of our systems, and before extracting features and matching with the given reference responses, the output of ASR is first cleaned for further processing. In this stage, abbreviations in the transcript text are expanded (e.g 'I'd' to 'I would) and some duplicated words are removed.

6.2 Cosine similarity

Cosine Similarity (CS) measure is used to compute the similarity between the user response and each reference response in the grammar file. Formally, given a user response

UR and its possible references $PR = [PR_1, PR_2, \ldots, PR_N]$ where N is the number of all possible responses for the corresponding prompt. Cosine similarity is calculated as:

$$CS(UR, PR_i) = \frac{\sum_{n=1}^{m} URw_i.PRw_i}{\sqrt{\sum_{n=1}^{m} URw_i^2}.\sqrt{\sum_{n=1}^{m} PRw_i^2}} \tag{6.1}$$

Where, UR=$[URw_1, \ldots, URw_m]$ represents the m-dimensional vector for the user response UR and PR_i=$[PRw_1, \ldots, PRw_m]$ represents the vector for the ith possible response PR. Note that, all of these vectors are computed using bag-of-words model of all distinct terms occurred in the set of all possible responses PR. Then each vector is multiplied by a weighting vector $Tw = [Tw_1, Tw_2, \ldots, Tw_m]$, where m is the number of distinct terms and Tw_i is the weight of corresponding term T_i calculated as:

$$Tw_i = TF_i * \frac{n_i}{N} \tag{6.2}$$

where TF_i is the frequency of a term T_i in response $r_i \in \{\text{UR,PR}_i\}$, N is the number of all reference responses in PR and n_i is the number of possible responses containing term T_i. Weighting vector in equation6.2 puts more weight on the term that occurs more frequently in the corresponding possible responses.

6.3 Part-of-Speech (POS) level similarity

In this type of similarity, a sequence is created for each user response by converting each possible response for that user response into its corresponding POS (Part of Speech) level. Formally, let $r_i \in \{\text{UR,PR}_i\} = [t_1, t_2, \ldots, t_m]$ represents all the terms occurred in a certain response. Thus, the POS-level list for r_i can be represented by POSTAG = $[p_1, p_2, \ldots, p_m]$, where the p_i is the part of speech for the term t_i in the response r_i. Also, let $PR = [PR_1, PR_2, \ldots, PR_N]$ represents a collection of all possible responses for a certain user response, POS_{PRi} is the POS-level list for possible response PR_i and POS_{UR} is the POS-level list for user response. Therefore, the similarity between POS_{UR} and POS_{PRi} is estimated by Jaccard score (JS), as shown in the following equation:

$$JS(POS_{UR}, POS_{PRi}) = \frac{POS_{UR} \bigcap POS_{PRi}}{POS_{UR} \bigcup POS_{PRi}} \tag{6.3}$$

The cosine similarity measure can be applied to measure the distance between two vectors of real values. So, we compute the similarity between two POS-level lists using jaccard score.

However, Jaccard score does not take into consideration the order of the elements in the two sets. So, Ratcliff/Obershelp pattern matching algorithm [78] is employed to measure the similarity between POS_{UR} and POS_{PRi} sets.

6.4 Proposed systems

6.4.1 Basic rule-based judgment approach

The 2018 spoken CALL shared task basic system [12] was used as a baseline system for all of our proposed systems. The proposed systems take a final decision about the given response (correct or incorrect), by passing audio transcription given by ASR through a sequence of stages and rules:

- **Rule1**: Extract the grammar errors using Python checker tool [1]. Therefore, if a grammar errors is found, the system rejects the response at this stage. One example in the test set is "I going in the holiday". This is the student response for 'Sag: Ich gehe in die Ferien' prompt. In this case the Python checker tool detected the grammatical error in this response. Another example is 'these is my password' which is the student response on the 'Sag: Dies ist mein Pass' prompt.

- **Rule2**: If the response has no grammatical errors, the system converts each possible response in Grammar XML file (i.e. reference responses) into its corresponding POS_{PRi} set. Similarly, it converts the user response into POS_{UR} set. Then, it computes the Jaccard coefficient and the Ratcliff distance (RD) between POS_{PRi} and POS_{UR} sets. Therefore, If the maximum value of the Jaccard measure is less than an experimentally predefined threshold (JAC_{TH}) and the maximum value of the RD is less than an experimentally threshold (RD_{TH}) , the system rejects the response (i.e. the response is incorrect). For example, the student response 'i would like to sit in the front ' is converted to POS level as follows: ('i', 'NN'), ('would', 'MD'), ('like', 'VB'), ('to', 'TO'), ('sit', 'VB'), ('in', 'IN'), ('the', 'DT') and ('front', 'NN'). One of the references for this prompt is 'i want to sit in the front ' and can be converted to POS level list to: ('i', 'NN'), ('want', 'VBP'), ('to', 'TO'), ('sit', 'VB'), ('in', 'IN'), ('the', 'DT'), and ('front', 'NN'). Then, the Jaccard score is computed using equation 6.3 which is equals to 0.71. To take the

[1] https://pypi.python.org/pypi/grammar-check/1.3.1

rest of references into consideration, we compute the Jaccard score for between the student response and each possible reference. The maximum value is compared with experimentally tuned threshold to accept or reject that response.

- **Rule3**: If the conditions in step 2 and step 3 above are not satisfied, each response (user response and all its references) is represented using real-values vector using bag of words model. Then, the system computes the cosine similarity between the student vector and each reference vector. The system takes the maximum value and compares it with an experimentally threshold (COS_{TH}) to decide if the response is correct or not. This threshold value is practically tuned on the enrollment data.

Each response is processed using these three rules as shown in the figure 6.1. Each rule tries to find some kinds of errors in the response. Finally if the response passes through all the rules, it will be accepted.

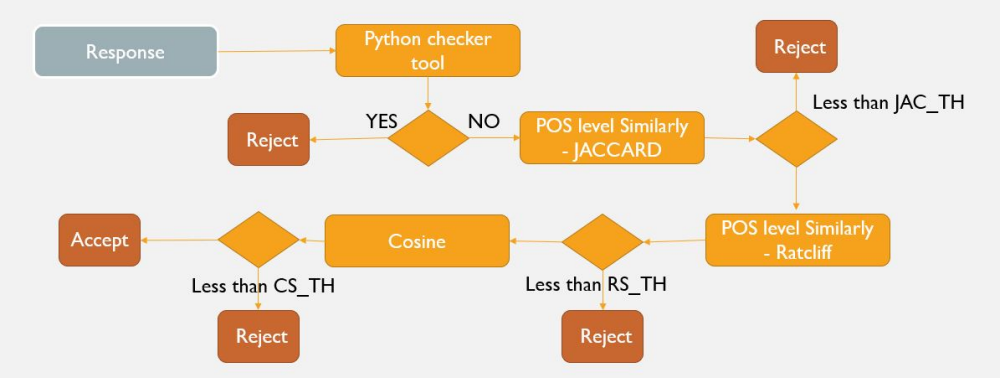

FIGURE 6.1: Rule based approach to process the spoken response

6.4.2 Rule-based judgment with optimization approach

This section shows how the rule based method can be combined with an optimization technique to enhance the overall performance. First, it shows how all thresholds (JAC_{TH}, RD_{TH} and COS_{TH}) that used in rule based method can be optimized using the genetic algorithm. Also, the optimization technique will be used to extract a list of all incorrect bi-gram tokens which plays an essential rule in the final decision.

6.4.2.1 Fusion of multiple systems

To handle some of the errors caused by ASR system, an additional two well-known ASRs were used to process the user response including; Google ASR and Microsoft Bing

ASR. Therefore, each user response is converted into text using these ASRs in addition to the SLaTE2018 ASR to get three transcriptions, $TEXT_{GOOGLE}$,$TEXT_{BING}$ and $TEXT_{SLaTE2018}$. Table 6.1 shows the transcriptions of three examples recognized by the three mentioned ASRs. It is clear that GOOGLE and BING ASRs are more accurate than the baseline ASR in the first example. On the other hand, the baseline ASR performs better in the second and third examples.

TABLE 6.1: Examples for different recognized texts.

Recognized Text	**ASR**
from italy	True Transcription
i'm from italy	SLaTE2018
from italy	GOOGLE
from italy	BING
i would like to buy some boot	True Transcription
i would like to buy some boots	SLaTE2018
i would like to buy some food	GOOGLE
i would like to play some food	BING
I want to leave at Tuesday	True Transcription
I want to leave at Tuesday	SLaTE2018
I want to leave on Tuesday	GOOGLE
I want to leave at two today	BING

Algorithm 2 describes how these three recognized texts are combined to make the final decision. The algorithm starts by computing CS, JS, and RD scores for each ASR transcript. It computes each similarity measure between $TEXT_{SLaTE2018}$ and each reference response and selects the maximum scores (three scores as described in 6.4.1). The same process is applied for $TEXT_{GOOGLE}$ and $TEXT_{BING}$ to produce CS, JS, and RD scores for each ASR.

We adopt weighted linear sum of each ASR score, such that $Score_i = W_{i1} * Score_{iASR1} + W_{i2} * Score_{iASR2} + W_{i3} * Score_{iASR3}$. All wights ($W_1$,$W_2$,...,$W_9$), and thresholds ($JAC_{TH}$, RD_{TH} and COS_{TH}) in algorithm 2 were optimized using the genetic algorithm.

The function "pythonGrammarCheck" in algorithm 2 returns 1 if grammar checker tool detects an error in the response.

The configurations of the genetic algorithm is as follow: the chromosomes are represented by a list of length 12 (all elements are real numbers), where the first 9 positions 1_{th} to 9_{th} hold the weights W_1 to W_9, while the remaining positions (10_{th},11_{th} and 12_{th}) were used to hold JAC_{TH}, RD_{TH} and COS_{TH}, respectively. The chromosome was initialized by 12 random numbers between 0 and 1 taking into account three constrains:

1. $W_1 + W_2 + W_3 = 1$
2. $W_4 + W_5 + W_6 = 1$
3. $W_7 + W_8 + W_9 = 1$

For mutation operator, we choose a randomly position in the chromosome and set its value to new random number between 0 and 1. Two point crossover operator were used. The crossover probability and mutation probability were set to 0.7 and 0.3, respectively.

The D score of 6698 training samples was used to decide how chromosome fitness will be evaluated during the iterations. In each training example, algorithm 2 was used to accept/reject the response. However D score was set to 0 when the system rejects less than 25% of all incorrect responses.

Figure 6.2 illustrate the idea of the optimization in more details. For simplicity, we will consider the optimization for three thresholds, but the same procedure can be applied to all variables. The genetic algorithm starts with set of random solutions (called chromosomes). To simplify the idea, we consider two chromosomes in this example. Each chromosome is initialized randomly. After that, the algorithm evaluates each chromosome (Compute the D score in our case) and then do the crossover and mutation operators on the parents to determine a better chromosome. In this example, the crossover operator was employed by selecting the JAC_{TH} and Rd_{TH} thresholds from the first chromosome and COS_{TH} from the second one. The resulting chromosome is evaluated again and the algorithm keeps the thresholds if it is better from the two parents. The process will repeated again for 1000 iterations to enhance the chromosome more and more.

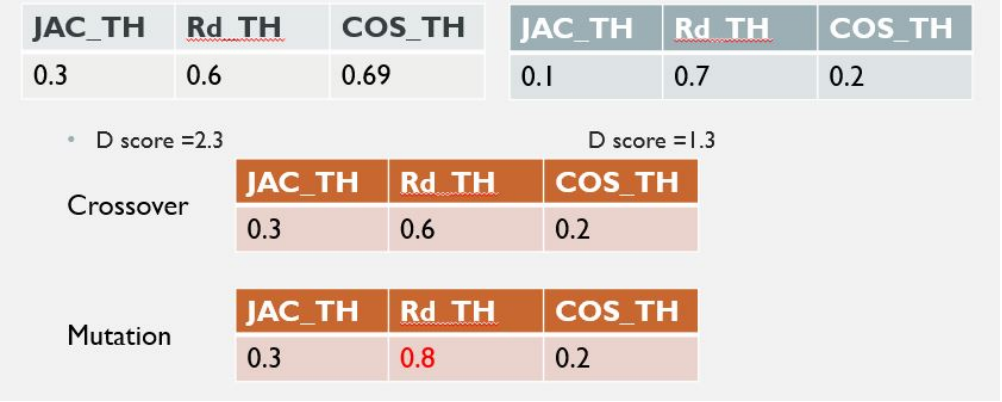

FIGURE 6.2: Tuning the thresholds using the genetic algorithm.

Algorithm 2 provide a different decision when the genes in a specified chromosome fluctuate through iterations, because the wights W_1 to W_9 and all thresholds are inputs

to this algorithm. Finally, a tournament selection was used, and the algorithm was executed for 1000 generation with population size equal 100.

Algorithm 2: Classification of the response based on the weights and thresholds

Input RecognizedTranscripts= [$TEXT_{GOOGLE}$,$TEXT_{BING}$,
$TEXT_{SLaTE2018}$], W_1 to W_9, JAC_{TH}, RD_{TH},COS_{TH};
Initialize JacScoreList=[], RDScoreList=[],
CosScoreList=[], pythonCheckList=[];
while *Text= RecognizedTranscripts.getElement* **do**
    while *Possible Response= getPossibleResponse* **do**
        CosineValues.add(CS(UR, PR_i);
        Jaccards.add(JS(POS_{UR},POS_{PRi});
        NormalizedEDs.add(RD(POS_{UR},POS_{PRi});
    end
    JacScoreList.add(MAX(Jaccards));
    RDScoreList.add(MAX(NormalizedEDs));
    CosScoreList.add(MAX(CosineValues));
    pythonCheckList.add(pythonGrammarCheck(Text))
end
CosScore = W_1 * CosScoreList[0] + W_2 * CosScoreList[1] + W_3 * CosScoreList[2] ;
JacScore=W_4 * jacScoreList[0] + W_5 * jacScoreList[1] + W_6 * jacScoreList[2];
RDScore = W_7 * RDScoreList[0] + W_8 * RDScoreList[1] + W_9 * RDScoreList[2];
if *sum(pythonCheckList)* $\geqslant 2$ **then**
    decision=reject;
else if *JacScore*$< JAC_{TH}$ *and RDScore*$< RD_{TH}$ **then**
    decision=reject;
else if *CosScore* $< COS_{TH}$ **then**
    decision=reject;
else
    decision=accept;
end
Output decision

6.4.2.2 Extract a list of incorrect bi-grams

In this approach, a list of all incorrect bi-gram tokens are extracted from the user response that has one or more language errors (Language="incorrect and meaning = "correct"). Similarly, the bi-gram tokens of all of the corresponding reference responses are extracted. As figure 6.3 shows, if there is a bi-gram in the grammatically incorrect

response which does not exists in any reference response bi-grams, we add it to a new list called "list A".

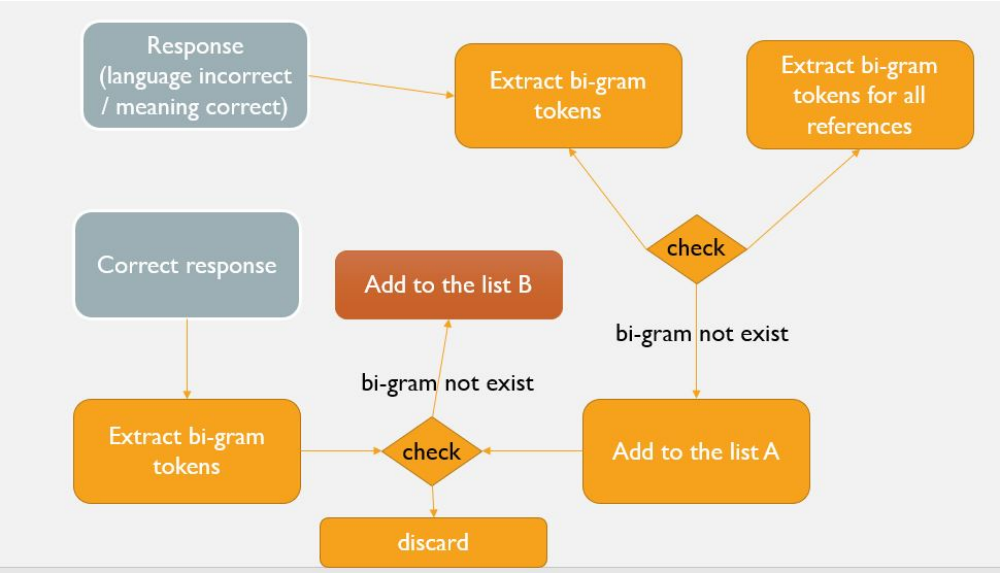

FIGURE 6.3: Procedure of extracting a list of incorrect bi-grams.

On the other hand, if a correct user response includes a bi-gram that is found in "list A", we remove it from the list. Otherwise, the bi-gram will be added to the 'list B'.

However, not all bi-gram tokens in the "list B" cause a linguistic errors because the student may correct his response. For example, the student response "can I pay with credit card ah post card sorry" is classified as correct and it has "card sorry" bi-gram which does not exist in any reference response.

Genetic algorithm was used as an optimization technique to refine the "list B". Formally, let Bi=[BI_1, . . . ,BI_m] represents all bi-grams in "list B", where m is the number of bi-gram tokens in the list. The chromosome is modeled as binary list [C_1, . . . ,C_m]. In this representation, the gene C_i equals zero if removing the bi-gram BI_i leads to increase the D score of training examples. Otherwise, C_i equals one.

To compute the D score of 6698 training examples, the weights W_1 to W_9, and thresholds (JAC_{TH}, RD_{TH} and COS_{TH}) were fixed and computed as described in section 6.4.2.1. Two point crossover operators were used. For mutation, we flip the value of a randomly chosen gene in the chromosome. The crossover probability and mutation probability were set to 0.7 and 0.3 respectively. The algorithm was executed for 1000 generation with population size equal 100.

To add the refined list in the judgment procedure, the student response is rejected when the recognized text contains a bi-gram in the refined list.

6.4.3 Machine learning Approach

This section shows how to build a machine learning model to predict the final decision. This model mainly depends on the extracted features from the student response and its references (set of possible responses in the grammar file). In this section, we investigate the effect of many extracted features on the overall system performance. First, many features were extracted from the output of two ASRs: Google ASR and SLaTE2018 ASR which was released by the shared task organizers. Then, we use the universal sentence encoder [79] to encode each sentence (student response and each possible reference) into 512-dimensional vector, and we extract the features based on embedding vectors.

6.4.3.1 Part-of-Speech (POS) level features

We use the POS taggger in NLTK toolkit [2] to generate POS set for the student response(the transcription given by the ASR) and each possible reference.

Formally, let the set [t_1, t_2,t_3 . . . , t_m] represents all terms that the transcription consists of. The set [pos_1, pos_2, pos_3. . . , pos_m] is produced by NTLK POS tagger, where pos_i is the part of speech for the term t_i. Also, POS level set was generated for each reference. After that, we compute the similarity between the student response and each reference by equation 6.4.

$$\frac{POSs \bigcap POSr}{POS_s \bigcup POS_r} \tag{6.4}$$

where $POSs$ represents the POS level set for the student response, and $POSr$ is the POS level set for a reference response. This equation was computed for each possible reference in 'grammar.xml' file. Two features were extracted from this approach:

- ***F*1** is the maximum similarity value between transcription given by SLaTE2018 ASR and each possible reference.
- ***F*2** is the maximum similarity value between transcription given by Google ASR and each possible reference.

Also another two features were extracted using Ratcliff distance similarity measure.

beginitemize***F*3** is the maximum similarity value between transcription given by SLaTE2018 ASR and each possible reference. ***F*4** is the maximum similarity value between transcription given by Google ASR and each possible reference.

[2] https://www.nltk.org/

6.4.3.2 Features using cosine similarity

- F5: this feature represents the maximum cosine similarity value between transcription given by SLaTE2018 ASR and each possible reference.

  F6: this feature represents the maximum cosine similarity value between transcription given by Google ASR and each possible reference.

6.4.3.3 Features using Python checker tool

We use a Python checker tool [3] to extract grammar errors from audio transcription given by the ASR. Two features were extracted using this tool:

- ***F*7**: The number of grammar errors produced when this tool was applied on the transcription given by SLaTE2018 ASR.
- ***F*8**: The number of grammar errors produced when this tool was applied on the transcription given by Google ASR.

6.4.3.4 Features produced by universal sentence encoder

This Encoder converts any sentence into a high dimensional vector which can be used for natural language processing tasks such as text classification and semantic similarity. The encoder is provider with a variable length English sentence as an input to construct 512-dimensional vector. We use this encoder to process the text from the student response and its all references. Formally, let V_i=$[F_1, F_2, ..., F_{512}]$ represents the feature vector for the student response. Also, let the set PR=$[R_1, R_2, R_3, ..., R_N]$ represents all possible references in 'grammar.xml' file. Each element in PR is a possible reference and consists of 512 features R_i=$[F_1, F_2, ..., F_{512}]$. The universal sentence encoder was used to generate the feature set. Then, the cosine similarity measure was computed for each (V_i,R_i) pair. Finally, we take the R_i which has maximum cosine value (maximum similarity) to compute the difference between it and V_i. The difference vector was provided to a machine learning algorithm to predict the final decision.

6.4.3.5 Response embedding to binary features

In this section, each student response and all its references were embedded into 438 binary features vectors. Let the set D = [t_1, t_2,t_3 . . . , t_m] represents all terms in

[3] https://pypi.python.org/pypi/grammar-check/1.3.1

grammar file. Every term t_i was normalized by removing punctuation and stemming using Porter Stemmer. Then, the normalized term is added to the list B if it is not added before. The terms in the list B were used to extract the 438-dimensional vector for all responses, where the number 438 is the size of the list B.

Each student response is tokenized to find its terms. Then, each term is normalized and added to the list S_T. The same process is applied on each possible reference to find the list P_T. Let SF=[F_1, F_2 . . . ,F_{438}] represents the 438-dimensional features vector for the student response. $F_i = 1$ if the i_{th} term in the list B exists in the list S_T. Otherwise $F_i = 0$. In the same way, we can find the 438-dimensional features vector RF_i for each possible reference.

Let the set $PF = [RF_1, RF_2, \ldots, RF_N]$ includes group of 438-dimensional vectors where each vector RF_i for a possible reference and N is the number of all possible references for a student response. The similarity between SF and RF_i can be computed by equation 6.5.

$$similarity = \sum_{i=1}^{438} SF_i * F_i \tag{6.5}$$

Where SF_i and F_i represent one item in SF and RF_i vectors respectively.

After computing the similarity measure between each (SF,RF_i) pair. We take the RF_i which has maximum similarity value to compute the difference between it and SF. The difference vector was provided to a machine learning algorithm to predict the final decision.

6.5 Implementation for mobile application

In this section, we present the components that used in our mobile application, the control flow between its activities, and all libraries used to simplify the implementation.

6.5.1 The main application flow

In this section, we present the basic flow of our application. First, the Login screen allows the user to sign in with different options including: FaceBook, Gmail, and any email the user like to register with. Figure 6.4 shows these three options.

If the user does not have a Google account or FaceBook account, the user will be allowed to register in our application using any account he prefers.

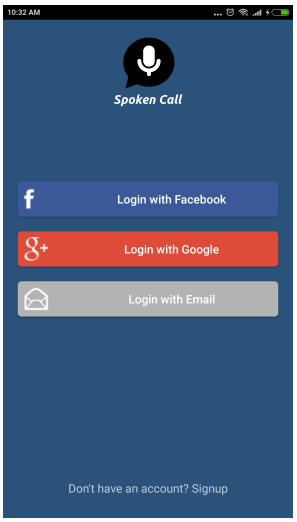

FIGURE 6.4: Login activity

The main menu for our application is shown in figure6.5. It includes the following buttons:

- START button: it moves the user speech challenge activity.
- CATEGORIES button: it shows all categories used in speech challenge.
- PAID VERSION button : it moves the user to a web page and ask him if he wants to buy a complete version which includes more challenges and more categories with various topics.
- BEST RESULTS: it shows the logs for the user including the best result for each category.
- CONTACT US button: it moves the user to activity to allow him to contact us with our emails.
- ABOUT US button: it moves the user to new activity to show some information about the developers.

Figure 6.6 shows the most important part of our application. The android application requests the server to get a prompt and its image, it displays the image in ImageView component and prompt text in the TextView component. The role of the user here is to answer on the provided prompt and the app records the his/her answer. The Google ASR is used to convert the recorded audio file to corresponding transcript which will be submitted to the server. Finally, the server processes the user response by using our developed techniques and displays the result for the user.

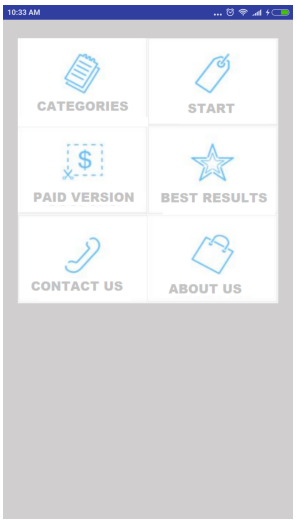

FIGURE 6.5: About Us Activity

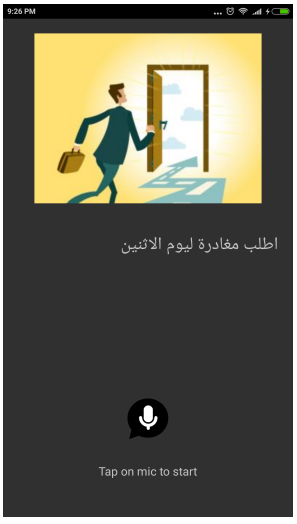

FIGURE 6.6: Speech challenge activity

Chapter 7

Experiments and Results

7.1 Dataset

All of our experiments and results demonstrated in section6.4 are conducted using a free available dataset which was collected by the organizers of a spoken CALL shared task [80] designed for German-English languages. The details of shared task dataset in section2.1.

Each response consists of 5 items: Prompt in German, text transcription output of English ASR, human transcript of audio file, language annotation and meaning annotation. The annotation can be either "correct" or "incorrect". When the language field is correct, the response is correct in terms of meaning and grammars.

7.2 Evaluation metrics (D score)

To evaluate the overall system and to easily compare its performance with similar systems, D score is used as a performance measure for the overall system. As the metric for evaluating the quality of systems competing in this task. So, we will compare between different versions of our system based on this metric. D metric can be evaluated by equation2.1. The D score for the base-line system of the spoken shared task is equal 1.0.

7.3 Experiments for rule-based approach

This section reports the results for rule-based approach which takes a final decision about the given response by passing audio transcription given by ASR through a sequence of stages and rules. The results for combinations of the rules in section6.4.1 is investigated.

System1:

it is a simple one, where it uses baseline ASR and applies rule1 and rule3 that descried in section 6.4.1.

System2: It has the same structure of System1. But we expand the grammar XML file by adding the correct answers in training set to generate more possible responses for a certain prompt.

System3: It adds another level of enhancement on system2. It adds new phase to check the grammar errors(rule2 in section 6.4.1), where POS tag for the user response and each response in the grammar XML file were collected to compute the jaccard coefficient between them. If the maximum value is lower than a certain threshold, the system will reject the response.

System4 uses baseline ASR and applies three basic rules that descried in section 6.4.1. Also, it disables the spell checker and excludes the following rules in Python wrapper for language check:

- $SENTENCEFRAGMENT$ Rule like 'where is the pool'
- $MUCHCOUNTABLE$ Rule
- $AINFINITVE$ Rule

System	1st edition			2nd edition		
	IRej	CRej	Dscore	IRej	CRej	Dscore
System1	0.389	0.183	2.13	0.63	0.21	3.03
System2	0.503	0.141	3.57	0.57	0.14	4.01
System3	0.839	0.18	4.66	0.57	0.10	5.51
System4	0.783	0.063	12.4	0.40	0.06	6.50

TABLE 7.1: Evaluation for rule-based approach, where IRej is the rejection rate on incorrect responses and CRej is the rejection rate on correct responses.

Referring to the shared task instructions, the system should correctly reject at least 25% of total of incorrect responses to guarantee that it is not biased to accept most of the responses. The percentage of correctly reject responses in System1 is 39%. This

guarantee that this system is not biased to correct decision and we insured that we prevent "gaming" of the D metric and the system will not be biased to one decision.

Adding correct responses from the training data to the grammar file increases the opportunity to give the right decision for the user response. The D score is improved from 2.13 to 3.57 and from 3.03 to 4.01 when the system2 was evaluated using the test data of CALL shared task first and second editions respectively.

The rule1 and rule3 are not sufficient for giving the right decision. For example, the cosine similarity value for student response "can i a ticket for piccadilly circus" is one, and Python checker tool does not check any grammatical errors in that response. So, System3 adds rule2 (using jaccard coefficient metric only without using Ratcliff distance) to improve the results by rejecting some responses that have grammatical errors. There are many examples that are correctly classified by System3: "the capital from french is paris", "i go in the holidays" , "i would like leave at thursday afternoon","can i have a ticket to the green park", and "i would like to go at thursday evening".

However, Jaccard score does not take into consideration the order of the elements in the two sets. So, System4 employs the Ratcliff distance (RD) metric to measure the similarity between transcription given by SLaTE2018ASR and each possible reference. For example, the jaccard coefficient for the response " i would like pay with the credit card" is one but RD value of that response is 0.9.

There are some rules in Python checker tool should not be checked in such systems. For example, the response "where is the pool" does not end with a question mark. So, System3 rejects this response as it contains a grammar error. However, CALL systems should not check if the output of the ASR includes a question mark or not. System4 handle this case by excluding three rules in Python checker tool.

7.4 Experiments for optimization-based approach

Table 7.2 shows the results for **system5** which applies the fusion technique described in section 6.4.2.1 and for **system6** which adds the "incorrect bigrams" list described in section 6.4.2.2. In the first columns of Table 7.2 (1st edition), **System5** uses the training data of first edition of CALL shared task to optimize all wights and thresholds ($JAC_{TH}, RD_{TH} and COS_{TH}$). In the second column of Table 7.2, the training data of second edition of CALL shared task was used to optimize all wights and thresholds. As the same manner, **system6** extracts the "incorrect bigrams" list from training data of first edition and then we report the results in the "1st edition" column when the system was evaluated using testing data. Then the training data of second edition was used to

extract that list and the results were reported in "2nd edition" column when the system was evaluated using the testing data for second edition.

These wights and thresholds were used at testing phase using the testing data of CALL shared task.

System	1st edition			2nd edition		
	IRej	CRej	Dscore	IRej	CRej	Dscore
System5	0.53	0.01	37.61	0.26	0.02	12.12
System6	0.59	0.01	42.23	0.33	0.02	14.41

TABLE 7.2: Evaluation for optimization approach, where IRej is the rejection rate on incorrect responses and CRej is the rejection rate on correct responses.

As shown in the table 7.2, the performance of our system is improved to 37.61 and 12.12 (D score) when was evaluated on testing data of first and second edition of CALL shared task respectively. This is because of the fusion weights (W_1 to W_9) and thresholds (JAC_{TH}, RD_{TH} and COS_{TH}) optimization with the genetic algorithm.

Also, the results show the improvement (D score of 42.32 and 14.41) when adding the refined list of incorrect bi-grams to system5. As expected, this list did not affect the rejections on correct responses (CRej), but it improved the D score by increasing IRej (rejections on incorrect responses). For example, the student response 'i would like to pay be visa' has an incorrect bi-gram 'be visa' which was extracted from training data. Moreover, this list helps us to find the correct usage of 'the' before a noun. For example the response 'i want a ticket to the piccadilly circus' has an incorrect bi-gram 'the piccadilly'. So, it was classified as incorrect response (note that the word 'the' should be removed to accept that response) .

7.5 Experiments for machine learning-based approach on eight features (F1 to F8)

Different types of machine learning algorithms have been used to train the model based on features(F1 to F8). The features (F1 to F4) is described in section 6.4.3.1, F5 and F6 is described in section 6.4.3.2 and finally (F7 and F8) is described in section 6.4.3.3.

Table 7.3 shows the results for four machine learning algorithms that used to trains the 7 models.

In table 7.3, we use four kernels to train the SVM model: linear, sigmoid, polynomial($degree = 3$), and Gaussian. The result of each kernel is reported in one row. Regarding to DNN

Model	1st edition			2nd edition		
	IRej	CRej	Dscore	IRej	CRej	Dscore
KNN	0.86	0.04	20.48	0.50	0.07	6.99
SVM-linear	0.82	0.03	24.37	0.47	0.06	7.99
SVM-sigmoid	0.57	0.03	19.50	0.29	0.05	6.50
SVM-polynomial	0.83	0.03	24.65	0.49	0.07	6.63
SVM-Gaussian	0.84	0.03	25.15	0.51	0.07	7.31
XGBoost	0.90	0.04	20.8	0.59	0.07	7.88
DNN	0.69	0.03	**26.13**	0.60	0.07	**8.69**

TABLE 7.3: Evaluation for machine learning approach, where IRej is the rejection rate on incorrect responses and CRej is the rejection rate on correct responses.

model the number of hidden layers was set to three and the number of epochs was set to 20. The results after training each model on training data of first edition were reported in first column "1st edition" and the results of testing the models that were trained on training data of second edition were reported in the second column.

The D-score of KNN method was increased to (26.13 in 1st edition , 7.67 in 2nd edition) after using training data of first and second editions together as one training set. However, there is no improvement in case of SVM on that training set. This is because SVM model is represented by the position of support vectors that may not be affected by increasing the size of training data.

The effect of regularization parameter of SVM is shown in Fig7.1. D score is decreasing as regularization parameter is increased. This indicates that the two classes are overlapped and no perfect decision boundary exists in training data. To handle this case, SVM implementation can be used to softening the decision boundaries and allows to include some points inside the margin.

7.6 Experiments for the two embedding methods

In this section, we present the results of four DNN models, were Feed-forward neural networks [81] was used to train all models. Also, four different feature sets were investigated:

- Feature Set1 and includes: 512 features that described in section 6.4.3.4.
- Feature Set2 and includes: Two features were described in section 6.4.3.3, two features were described in section 6.4.3.1, and 512 features that described in section 6.4.3.4.

FIGURE 7.1: The effect of regularization on D score

- Feature Set3 and includes: 438 features that described in section 6.4.3.5.
- Feature Set4 and includes: Two features were described in section 6.4.3.3, two features were described in section 6.4.3.1,and 438 features that described in section 6.4.3.5.

According to the above four categories, the performance of the overall system is calculated by the Differential (D) score. The performance of each trained model is reported in table 7.4 with a comparison between them based on rejection rate on incorrect responses (IRej), rejection rate on correct responses (CRej), D-score. All results were evaluated using test data of the 2018 CALL shared task, as shown in table 2.1.

TABLE 7.4: Results of the four proposed models, where IRej is the rejection rate on incorrect responses and CRej is the rejection rate on correct responses.

Model	**IRej**	**CRej**	**D-score**
Model-1	0.50	0.05	9.09
Model-2	0.55	0.05	10.0
Model-3	0.41	0.05	8.89
Model-4	0.58	0.06	10.2

Table 7.4 show the results of the four trained model. Feature Set1, Feature Set2, Feature Set3, and Feature Set4 were used to train Model-1, Model-2, Model-3, and Model-4 respectively. We can note that the grammar features played an essential rule to increase the rejection rate (and to enhance the D-score) when were added to Feature Set1 and Feature Set3 because of its ability to detect some grammar errors in the incorrect utterances. Also, the results show that The Model-1 and Model3 are comparable in term

of D-score. This proves that the two ways of embedding that were described in section 6.4.3.4 and section 6.4.3.5 are capable to represent the student response in this task.

In the this task, the cost of accepting a grammatically incorrect response is less than the cost of rejecting a correct response. Model-2 and Model-4 were fused together so that the final decision is 'reject' if the output of the two models is 'reject'. Otherwise, the final decision is 'accept'. The fused system was evaluated on the same test set, and achieved D-score of 13.87.

7.7 Comparison and discussion

In this section, we compare our results with other four systems [36–39] as all systems were evaluated on the same test set and using the same measures. As shown in [36, 38], the improvement on the ASR component has a key factor of increasing the D-score. For this reason, we use an additional ASR (Google ASR) in section 6.4.3.1 and section 6.4.3.3 to handle some of the errors caused by ASR system.

Further tuning for the model parameters to increase the D-score was proposed in [36]. In section 6.4.2.1, the model parameters where optimized using genatic algorithm. The fused system achieved D score of 14.4 [7] in the 2018 spoken call shared task. Also, the fusion between two DNN models (Model-2 and Model-4) leads to increase the D-score of best DNN model (Model-4) from 10.2 to 13.87.

In general, both rule-based approaches [7, 39] and machine learning based approaches [36–38] achieved good results. Table7.5 reports the D-score of each system published in INTERSPEECH2018 in addition to our best results. Our systems are [7] and FFF which represents the fusion of Model-2 and Model-4.

TABLE 7.5: Comparison between results

System	[36]	[7]	**FFF**	[38]	[39]	[37]
D-score	19.088	**14.4**	**13.87**	10.764	10.08	7.397

Chapter 8

Conclusion and future work

The main objective of this research is developing a system which helps English learners to exercise and improve speaking skills in English conservation. This system prompts the student in his/her L1 language (German in our case) indicating in an indirect way what he/she is supposed to say in the L2 language (English in our case). Then, the systems automatically assess the spoken response, based the grammar and linguistic, and provides a feedback. University of Geneva released a Computer Assisted Language Learning (CALL) Shared Task System. This system is very simple and it is shared as base line. Many researchers introduced many enhancements to the baseline system and different approaches were applied to enhance the system performance. In this research , we develop and investigate different methods in natural language processing context to increase the performance. First, we proposed a rule-based model which mainly depends on the similarity between the user response and the set of reference responses, where the similarity is calculated in two ways: using the Cosine similarity between plain texts and using the Jaccard similarity based on Part Of Speech (POS) tagging. After that, the genetic optimization method was employed in two directions. It was used to tune the weights and parameters of the rule-based approach. Also, a list of bi-gram tokens was extracted from grammatically incorrect responses and then refined using the genetic algorithm. Finally, a DNN model was trained to assess the student responses in 2018 CALL shared task. Different types of features were investigated to improve the system performance. Finally, Two DNN models were fused together to improve the D-score. For further research, we recommend to build a DNN model with D-score as the cost function of the optimizer.

Bibliography

[1] Yoo Rhee Oh, Hyung-Bae Jeon, Hwa Jeon Song, Byung Ok Kang, Yun-Kyung Lee, Jeon-Gue Park, and Yun-Keun Lee. Deep-learning based automatic spontaneous speech assessment in a data-driven approach for the 2017 slate call shared challenge. In *Proc. 7th ISCA Workshop on Speech and Language Technology in Education*, pages 103–108.

[2] Reda Siblini and Leila Kosseim. Clac: Semantic relatedness of words and phrases. *arXiv preprint arXiv:1708.05801*, 2017.

[3] Rajesh Kumar Aggarwal and Mayank Dave. Integration of multiple acoustic and language models for improved hindi speech recognition system. *International Journal of Speech Technology*, 15(2):165–180, 2012.

[4] Mark Stamp. A revealing introduction to hidden markov models. *Department of Computer Science San Jose State University*, 2004.

[5] Horacio Franco, Harry Bratt, Romain Rossier, Venkata Rao Gadde, Elizabeth Shriberg, Victor Abrash, and Kristin Precoda. Eduspeak®: A speech recognition and pronunciation scoring toolkit for computer-aided language learning applications. *Language Testing*, 27(3):401–418, 2010.

[6] Emmanuel Rayner, Pierrette Bouillon, and Johanna Gerlach. Evaluating appropriateness of system responses in a spoken call game. 2012.

[7] Mohammad Ateeq, Abualsoud Hanani, and Aziz Qaroush. An optimization based approach for solving spoken call shared task. *Proc. Interspeech 2018*, pages 2369–2373, 2018.

[8] Claudia Baur, Cathy Chua, Johanna Gerlach, Emmanuel Rayner, Martin Russel, Helmer Strik, and Xizi Wei. Overview of the 2017 spoken call shared task. 2017.

[9] Emmanuel Rayner, Claudia Baur, Cathy Chua, and Nikolaos Tsourakis. Supervised learning of response grammars in a spoken call system. 2015.

[10] Claudia Baur, Johanna Gerlach, Emmanuel Rayner, Martin Russell, and Helmer Strik. A shared task for spoken call? 2016.

[11] Mengjie Qian, Xizi Wei, Peter Jancovic, and Martin Russell. The university of birmingham 2017 slate call shared task systems. In *Proceedings of the Seventh SLaTE Workshop, Stockholm, Sweden*, 2017.

[12] Claudia Baur, Cathy Chua, Johanna Gerlach, Emmanuel Rayner, Martin Russel, Helmer Strik, and Xizi Wei. Overview of the 2017 spoken call shared task. *Proc. Interspeech 2018*, 2018.

[13] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[14] T Robinson, J Fransen, D Pye, J Foote, and S Renals. Wsj-cam0: A british english corpus for large vocabulary continuous speech recognition. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 1994.

[15] A Robinson. The british english example pronunciation (beep) dictionary. *Retrieved from World Wide Web: ftp://svrftp. eng. cam. ac. uk/pub/comp. speech/dictionaries/beep. tar. gz*, 1996.

[16] Nico Axtmann, Carolina Mehret, and Kay Berkling. The csu-k rule-based pipeline system for spoken call shared task.

[17] Ahmed Magooda and Diane Litman. Syntactic and semantic features for human like judgement in spoken call. In *Proc. 7th ISCA Workshop on Speech and Language Technology in Education*, pages 109–114.

[18] Keelan Evanini, Matthew Mulholland, Eugene Tsuprun, and Yao Qian. Using an automated content scoring system for spoken call responses: The ets submission for the spoken call challenge.

[19] Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, et al. The ami meeting corpus: A pre-announcement. In *International Workshop on Machine Learning for Multimodal Interaction*, pages 28–39. Springer, 2005.

[20] Anton Batliner, Mats Blomberg, Shona D'Arcy, Daniel Elenius, Diego Giuliani, Matteo Gerosa, Christian Hacker, Martin Russell, Stefan Steidl, and Michael Wong.

The pf_star children's speech corpus. In *Ninth European Conference on Speech Communication and Technology*, 2005.

[21] J-L Gauvain and Chin-Hui Lee. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE transactions on speech and audio processing*, 2(2):291–298, 1994.

[22] Shakti P Rath, Daniel Povey, Karel Veselỳ, and Jan Cernockỳ. Improved feature processing for deep neural networks. In *Interspeech*, pages 109–113, 2013.

[23] ES Atwell, PA Howarth, and DC Souter. The isle corpus: Italian and german spoken learner's english. *ICAME Journal: International Computer Archive of Modern and Medieval English Journal*, 27:5–18, 2003.

[24] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[25] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[27] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[28] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[29] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

[30] Yue Zhang and Stephen Clark. Syntactic processing using the generalized perceptron and beam search. *Computational linguistics*, 37(1):105–151, 2011.

[31] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[32] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(4):694–707, 2016.

[33] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[34] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

[35] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(4):694–707, 2016.

[36] Huy Nguyen, Lei Chen, Ramon Prieto, Chuan Wang, and Yang Liu. Liulishuo's system for the spoken call shared task 2018. *Proc. Interspeech 2018*, pages 2364–2368, 2018.

[37] Keelan Evanini, Matthew Mulholland, Rutuja Ubale, Yao Qian, Robert Pugh, Vikram Ramanarayanan, and Aoife Cahill. Improvements to an automated content scoring system for spoken call responses: The ets submission to the second spoken call shared task. *Proc. Interspeech 2018*, pages 2379–2383, 2018.

[38] Mengjie Qian, Xizi Wei, Peter Jančovič, and Martin Russell. The university of birmingham 2018 spoken call shared task systems. *Proc. Interspeech 2018*, pages 2374–2378, 2018.

[39] Dominik Jülg, Mario Kunstek, Cem Freimoser, Kay Berkling, and Mengjie Qian. The csu-k rule-based system for the 2nd edition spoken call shared task. *Proc. Interspeech 2018*, pages 2359–2363, 2018.

[40] Chuan Wang, RuoBing Li, and Hui Lin. Deep context model for grammatical error correction. In *Proc. 7th ISCA Workshop on Speech and Language Technology in Education*, pages 167–171, 2017.

[41] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966, 2015.

[42] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[43] Holger H Hoos and Thomas Stützle. *Stochastic local search: Foundations and applications*. Elsevier, 2004.

[44] Thomas A Feo and Mauricio GC Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995.

[45] Maryam Najafian. *Acoustic model selection for recognition of regional accented speech*. PhD thesis, University of Birmingham, 2016.

[46] Steve Young. A review of large-vocabulary continuous-speech. *IEEE signal processing magazine*, 13(5):45, 1996.

[47] Mark Gales and Steve Young. The application of hidden markov models in speech recognition. *Foundations and trends in signal processing*, 1(3):195–304, 2008.

[48] Colin Champion and SM Houghton. Application of continuous state hidden markov models to a classical problem in speech recognition. *Computer Speech & Language*, 36:347–364, 2016.

[49] A Srinivasan. Speech recognition using hidden markov model. *Applied Mathematical Sciences*, 5(79):3943–3948, 2011.

[50] Rajan Mehla, RK Aggarwal, et al. Automatic speech recognition: a survey. *International Journal of Advanced Research in Computer Science and Electronics Engineering (IJARCSEE)*, 3(1):pp–45, 2014.

[51] Fahad Al-Otaibi. Speaker-dependant continuous arabic speech recognition. *Prince Salman Library*, 2001.

[52] Rajesh Kumar Aggarwal and Mayank Dave. Acoustic modeling problem for automatic speech recognition system: advances and refinements (part ii). *International Journal of Speech Technology*, 14(4):309–320, 2011.

[53] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

[54] Ruslan Salakhutdinov. Learning deep generative models. *Annual Review of Statistics and Its Application*, 2:361–385, 2015.

[55] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6): 82–97, 2012.

[56] Li Deng, Michael L Seltzer, Dong Yu, Alex Acero, Abdel-rahman Mohamed, and Geoffrey E Hinton. Binary coding of speech spectrograms using a deep auto-encoder. In *Interspeech*, pages 1692–1695. Citeseer, 2010.

[57] Darren Hau and Ke Chen. Exploring hierarchical speech representations with a deep convolutional neural network. In *11th UK workshop on computational intelligence (UKCI 11)*, page 37, 2011.

[58] Vikramjit Mitra, Wen Wang, Horacio Franco, Yun Lei, Chris Bartels, and Martin Graciarena. Evaluating robust features on deep neural networks for speech recognition in noisy and channel mismatched conditions. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[59] Adam P Piotrowski and Jarosław J Napiorkowski. A comparison of methods to avoid overfitting in neural networks training in the case of catchment runoff modelling. *Journal of Hydrology*, 476:97–111, 2013.

[60] Chao Tong, Jun Li, and Fumin Zhu. A convolutional neural network based method for event classification in event-driven multi-sensor network. *Computers & Electrical Engineering*, 2017.

[61] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[62] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[63] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *ICML*, volume 14, pages 1764–1772, 2014.

[64] Haşim Sak, Andrew Senior, Kanishka Rao, and Françoise Beaufays. Fast and accurate recurrent neural network acoustic models for speech recognition. *arXiv preprint arXiv:1507.06947*, 2015.

[65] Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech*, pages 3771–3775, 2013.

[66] Kazuki Irie, Zoltán Tüske, Tamer Alkhouli, Ralf Schlüter, and Hermann Ney. Lstm, gru, highway and a bit of attention: an empirical overview for language modeling in speech recognition. *Interspeech, San Francisco, CA, USA*, 2016.

[67] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014.

[68] H Jaeger. A tutorial on training recurrent n. networks, covering bppt, rtrl, and the echo state network approach. Technical report, Tech. report, Fraunhofer Inst. for Aut. Intelligent Systems, 2013.

[69] Nadia Hmad and Tony Allen. Echo state networks for arabic phoneme recognition. In *Proceedings of World Academy of Science, Engineering and Technology*, number 79, page 424. World Academy of Science, Engineering and Technology (WASET), 2013.

[70] Fabian Triefenbach, Azarakhsh Jalalvand, Kris Demuynck, and Jean-Pierre Martens. Acoustic modeling with hierarchical reservoirs. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(11):2439–2450, 2013.

[71] Ying Zhang, Mohammad Pezeshki, Philémon Brakel, Saizheng Zhang, Cesar Laurent Yoshua Bengio, and Aaron Courville. Towards end-to-end speech recognition with deep convolutional neural networks. *arXiv preprint arXiv:1701.02720*, 2017.

[72] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *ICML*, volume 14, pages 1764–1772, 2014.

[73] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5): 602–610, 2005.

[74] Abdel-rahman Mohamed. *Deep Neural Network acoustic models for ASR*. PhD thesis, University of Toronto, 2014.

[75] Nelson Morgan and Herve Bourlard. Continuous speech recognition. *IEEE signal processing magazine*, 12(3):24–42, 1995.

[76] Oliver Kramer. K-nearest neighbors. In *Dimensionality reduction with unsupervised nearest neighbors*, pages 13–23. Springer, 2013.

[77] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.

[78] John W Ratcliff and David E Metzener. Pattern-matching-the gestalt approach. *Dr Dobbs Journal*, 13(7):46, 1988.

[79] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[80] Andrew Caines. Spoken call shared task system description. In *Proc. 7th ISCA Workshop on Speech and Language Technology in Education*, pages 79–84.

[81] George Bebis and Michael Georgiopoulos. Feed-forward neural networks. *IEEE Potentials*, 13(4):27–31, 1994.