Jimma University

Jimma Institute of Technology

School of Computing

Hierarchical and Answer-to-Answer Attention Based Neural Network for Subjective Question Marking

Abebawu Eshetu

A Thesis Submitted to the School of Graduate Studies of Jimma University in Partial Fulfillment for the Degree of Master of Science in Information Technology

Jimma, Ethiopia

November (2017)

Jimma University

Jimma Institute of Technology

School of computing

Hierarchical and Answer-to-Answer Attention Based Neural Network for Subjective Question Marking

Abebawu Eshetu

Advisor: Dr. Fekade Getahun

This is to certify that the thesis prepared by *Abebawu Eshetu*, titled: *Hierarchical and Answer-to-Answer Attention Based Neural Network for Subjective Question Marking* submitted in partial fulfillment of the requirements for the Degree of Master of Science in Information Technology complies with the regulations of the University and meets the accepted standards with respect to originality and quality

Signed by Examining Committee:

Name	Signature	Date
Advisor: Fekade Getahun (PhD)		09/11/2017

Examiner: _____

Examiner: _____

Abstract

The Evaluation of students' capacity to construct a sustained argument with subjective questions allows mentors to assess implicit understanding ability of learners. However, manual evaluation of subjective question is challenging process and results grading inconsistency. From early 1960 several approaches are proposed to automate subjective question marking by giving due attention for essays. Recently, with advent of deep learning technique automatic essay assessment shown improved result that approaches to human raters without need of handcrafted features.

The aims of this study were to model that can able to evaluate both essay and short answer questions without handcrafted features using deep learning technique. Given essay or short answer word sequences, our model first embed word level context using FastText word vectors and sub-word embedding built by character based convolutional neural network. For essay, the model encodes embedded essay vectors hierarchically by applying two level bidirectional recurrent neural network. We applied hierarchical word and sentence level attention that extract most salient words encapsulated in a sentences and sentences encapsulated in an essay respectively. For short answer, we used the same encoder as essay for both model and student answer vectors. Then, we applied reference attention on encoded vectors using model answer vector as weight. Finally, answer-to-answer attention is applied to get the relatedness level of resulting vector and encoded model answer from model to student and student to model answer.

We evaluated our model on three datasets: Kaggle essay and short answer English dataset and Amharic short answer dataset prepared for this thesis work. Experimental results on Kaggle dataset show that our model achieves the state-of-the-art performance for both essay and short answer by improving weighted Kappa to +2 and +4 respectively. The experiment done on Amharic dataset shows promising result by achieving 66% and 62% correlation on Pearson and Kappa respectively on small sized dataset. This shows our model is capable of evaluating both short answer and essay questions from any domain in very human like way if trained on enough data. Our work not considered subjective questions with formulas and diagrams and we left open. We also recommend to include feedback that show how the model scored and rated missed points to student answer.

Key words: Deep Learning, Subjective Question Marking, Character CNN, FastText embedding, Reference attention, Answer-to-Answer attention, Hierarchical attention

Dedication

Fafiye you are always special!!!

Acknowledgments

Most of all, I would like to thank God, who makes everything possible, for helping me pass all those challenging times.

I owe my deepest gratitude to my advisor Dr. Fekade Getahun for the continuous support for his time, patience and undeniably helping comments all the way through this study. His knowledge and advice have helped me to keep on track and work at a smooth pace. I would like to thank Mr. Habtamu, Head Department of Jimma University Amharic Language and Literature, for facilitating all required resources including mentors who participated to score answers from department and his very helpful expert support.

Finally, and most importantly, I would like to thank my best Fafiye. Your support, encouragement, quiet patience and unwavering love were undeniably the bedrock upon which the past two years of my life have been built. Your tolerance of my occasional vulgar moods is a testament in itself of her unyielding devotion and love. I owe my deepest gratitude to Mahi for her encouragement and unreserved assistance. I will always cherish the time you spent with us. I also like to express my appreciation to my family members and friends who have helped me in so many ways. Finally, I want to thank all the people who have contributed in one way or another on this thesis work.

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Acronym

BLEU: Bi-Lingual Evaluation Understudy

BLSTM: Bidirectional Long Short Term memory

BRNN: Bidirectional Recurrent Neural network

CNN: Convolutional Neural Network

CPU: Central Processing Unit

DISCO: Distributional Similarity Co-occurrences

ESA: Explicit Semantic Analysis

GLSA: Generalized Latent Semantic Analysis

GMAT: General Management Aptitude Test

GPU: Graphics Processing Unit

GRU: Gated Recurrent Unit

IEA: Intelligent Essay Assessor

LDA: Latent Dirichlet Allocation

LSA: Latent Semantic Analysis

LSTM: Long Short Term Memory

MaxEnt: Maximum Entropy

NLP: Natural language Processing

OOV: Out of Vocabulary

PEG: Project Essay Grader

RNN: Recurrent Neural Network

SSWE: Score Specific Word Embedding

SQM: Subjective Question Marking

SVD: Single Value Decomposition

TOEFL: Test of English as a Foreign Language

CHAPTER ONE

1. INTRODUCTION

1.1 Background

E-assessment is the use of information technology for any assessment-related activity. This definition embraces a wide range of student activity ranging from the use of a word processor to on-screen testing. Due to its obvious similarity to e-learning, the term e-assessment is becoming widely used as a generic term to describe the use of computers within the assessment process [1]. Now a day, the most dominant assessment methods is paper based examinations. However, it is cumbersome, tedious and inefficient because it requires more time and resources in carrying out the checking and grading. In addition, when it is open questions it becomes much more difficult to evaluate than more restricted tests such as multiple choice tests or oral exams. Moreover, the time spent by teachers and moderators in E-learning courses is critical and costly in resources, so multiple choice questions (MCQ) seem to be a good option because they can automatically be evaluated even if such questions lack evaluating student reasoning skill and easy to guess. As compared to MCQ, subjective questions want students to write their own answer, it also permits students to put across and prop up their thoughts in response to the question. Because of this, students can exhibit their various capabilities and talents like describing his or her individual responses, producing their own assumptions, or explaining analysis. But on the other hand, the grading of such descriptive questions is costly and protracted. Furthermore, it includes probable measurement fault to check outcomes because of discrepancies in the grading process.

In addition, subjective question assessment is an inherently subjective process when carried out manually. Evaluators read, analyze and interpret the answer to be scored with different rate of errors and subjective differences. For instance, two teachers may not give the same grade to the same essay and also the same teacher may not give the same grade to the same essay on different occasions. Choosing representative and trained evaluators can circumvent this only to a limited amount. Empirical evidence for this can be found in the usually rather low inter rater correlation of two human assessors which typically floats around 0.6 to 0.8 [15]. So, applicability of computer based assessment has untold benefit from different point of view.

Supporting the human assessment process with automated scoring mechanisms is an excellent option to increase both effectiveness and efficiency in the assessment process and several attempts are proposed to automate assessment process by giving due consideration to essay assessment. With the initiation of innovative technology, for example, enhancement in the area of natural language processing, information extraction, and artificial intelligences, it is feasible to incorporate specific categories of subjective questions in automated tests because their trustworthy computerized scoring is now achievable. Some of the currently deployed automated essay scoring system including Electronic Essay Rater (E-rater) [2], Intelligent Metric System (IntelliMetric) [3], and the Intelligent Essay Assessor (IEA) [4] have shown to be successful and many standard international exams like General Management Aptitude Test (GMAT), Test of English as a Foreign Language (TOEFL) and others have started integrating them. It has also been developed in other language for example Japan Essay Scoring System [5], Automatic Chinese Essay Scoring [6] and so on. But, more focus is given to essay than short answer questions that are more common to assess students' implicit knowledge than essay questions.

State-of-art works used to automate subjective question marking use or combines machine learning community and knowledge based approaches [26]. However, both lexicalized machine learning approaches and Ontologies are relying on handcrafted quality features. Ontology based approaches usually better represent answer semantically, but are restrictive and domain dependent. One should build specific domain knowledge base to use such system. Similarly, machine learning approaches are also challenged to score answers in human manner as they need several features that represent input answer statement which are oversimplified and incomplete. Moreover, as Ontology the features are usually domain dependent and not transferable.

With the advent of artificial inelegancies, deep learning based models can represent feature for the given text as human beings do. They encode word vectors as knowledge to represent the given text. Recently, Dimitiros *et al.*, [47], proposed deep learning based approach to assess written essay and achieved promising accuracy in Kaggle essay dataset using domain trained word vectors as the only feature. The feature used with such deep learning models are transferable and easy to build as they can be encoded from unlabeled data that is readably exist today.

As compared to resourced languages such English, Amharic language gets limited attention despite the fact that about 18 government universities are launched in Ethiopian, Amharic Language and Literature department with mode of delivery of Amharic at undergraduate level and 10 universities at graduate level [7]. More than 12 core modules with different courses are delivering under Amharic Language and Literature department [8]. Amharic is also given as subject for primary and secondary school class in all region and all subjects are delivered with Amharic for primary school [9] on the other hand, number of student who join both higher and primary school education in Ethiopia increasing as intake capacity of universities and schools increase. As the result, especially in countries like Ethiopia that follow continuous assessment, load on teachers or lecturers also increase simultaneously.

The aim of this thesis is to develop the system that automatically evaluate and score Amharic and English subjective questions using neural attentive deep learning technique.

1.2 Statement of the Problem

In the context of Amharic language, despite the increasing number of students and schools offering courses in Amharic language, only one attempt has been proposed using Latent Semantic Analysis (LSA) by Abel [15] which limited to content of the text for Amharic factual essay. With advent of deep learning that encode meaning of words using neural word embeddings we can represent input answer in both syntactic and semantic way that is capable of assessing both essay and short answer without handcrafted features.

However, state-of-art works proposed by Dimitiros *et al.,* [47] represented essay in single vector by applying two-layer essay level bidirectional recurrent neural network. Unlike short answer, assessing essay is not restricted to few sentences. Essay are long and usually span into one or more paragraphs. Moreover, we need coherency in essay. That is how essay is organized than simple content analysis. Looking essays as hierarchically organized text can better represent essay structure than encoding all essay words to single vector. Furthermore, not all words in essay are equally important for meaning of essay. Because of the word vectors used to represent words in essay the model of Dimitiros *et al.,* [47] poorly treat rare words which is especially problematic for morphologically rich languages such as Amharic.

The works used to assess short answer used machine learning and or ontology to represent given model answer and student answer. Ontologies are domain dependent and restrictive for only those concepts which exist in taxonomy used. Moreover, to deal with structure of given answer such models require external sophisticated NLP tools such parser and pos-tagger. Machine learned models need features that are over simplified to represent answer. Furthermore, designing such features require intensive human power. However, short answer can be also assessed using deep learning models as essay.

Research questions

1. How to evaluate essay text by considering coherency as essay text is hierarchically or coherently organized using deep learning models?
2. How to assess short answer question using deep learning model?
3. How to consider rarely occurring and miss-spelled words in student answer with word vectors?
4. How to treat out-of-vocabulary words when encoding student answer?
5. How to score student answer by giving attention to only relevant concepts using neural models?

1.3 Objective of Study

General Objective

The general objective of this research is to investigate an automatic marking system model for subjective questions using deep learning model.

Specific Objective

The following specific objectives are identified in order to achieve the specified general objective:

- ✓ Conduct a literature search and literature review of existing subjective assessment techniques
- ✓ Collect corpus used for training word embedding model the system
- ✓ Collect corpus used to train and evaluate subjective question marking model
- ✓ Create word vectors for Amharic text
- ✓ Develop assessment and grading model for subjective question

- ✓ Evaluate the model in both English and Amharic dataset using appropriate statistical techniques

1.4 Methodology

The following methods are applied in order to achieve the above specified objectives.

1.4.1 Literature Review

A thorough literature review done on computer based assessment/E-assessment/Auto Marker in general and deep learning based approaches in particular for subjective question assessment. Moreover, techniques and tools used in each approach investigated and techniques or tools that can be used for assessment are adopted.

1.4.2 Data Collection

Two category of data is required to develop SQM; the data used to train and extract Amharic FastText word vector and data used to evaluate SQM system. The former collected from Amharic news, Amharic Wikipedia, educational sources, etc. The latter is answered pre-graded data collected from Jimma University Amharic Language and Literature department. For English we used publicly available standard Kaggle essay and short answer dataset[1].

1.4.3 Building Amharic Word Vectors

We trained our data collected for word vectors on Neural Network based FastText predictive model and generate meaningful word vectors to be used as external knowledge for our SQM system. In addition to word vectors, we also created character vector for each character in a word by averaging vectors of words that contain a character as characters are constituent of word.

1.4.4 Automatic Marking Model

We used attention based neural network model to assess both essay and short answer. For essay we represented input text hierarchically as essay are organized in coherent nature, we first encode sentences in essay to get more informative essay words and generate sentence level attended vectors. Then using sentence vectors we again apply same encoder to get essay level context. Since

[1] https://www.kaggle.com/c/asap-sas/data

short answer length may span from phrase to sentences, we encoded at answer level only. Then the encoded and attended essay or short answer context is provided as input to output layer Softmax classifier to predict score.

1.4.5 Testing and Evaluation

The SQM model evaluated using Amharic short answer sets collected from Amharic Language and Literature department. We also evaluated our model on Kaggle standard dataset for both essay and short answer. Using human rated score as gold standard, we evaluated correlation between gold standard and predicted scores using standard metrics such Pearson, Spearman, and Kappa. All tests in Kaggle dataset will be evaluated in Quadratic Kappa as Kappa is taken as standard evaluation metric for Kaggle dataset.

1.4.6 Scope and Limitation

The scope of our work is limited to score two subjective type questions such short answer and essay questions and model evaluation is done for both independently. We evaluate SQM essay model only on Kaggle standardized English written essay dataset and experiment for short answer will be done to both languages. For short answer, English dataset is used from Kaggle short answer and for Amharic we collect and prepare Amharic short answer data for experimentation purpose. We develop Amharic FastText word vectors as the only feature that our neural model use. Finally, we experiment and analyze result of our SQM.

Because of time constraints we will not consider the following subjective questions that require figures (diagrams), formulas, etc., proof type questions experimental questions. Our work also will not include feedback which is specific to missed points and instructional for specific student.

1.5 Application of Results

This research work is believed to produce an effective approach for assessments subjective question. The main application of this thesis result is on finding an efficient method of automatic evaluation system for subjective question. Therefore, it will have a significant usage for easing of a teaching and learning process in education.

More specifically, the proposed work will have applied on educational institutes to bring the following significance for instructors, students, and institutions.

Students

- ✓ Improves impartiality (machine marking does not 'know' the students so does not favor nor make allowances for minor errors)
- ✓ Improves reliability (machine marking is much more reliable than human marking)

Instructor

- ✓ It enables the assessment of a wide range of topics very quickly, with an associated reduction in the time that lecturers dedicate to marking.
- ✓ The need for double marking is totally eliminated. This time and resource saving allows more regular assessment than might otherwise have been possible, consequently enabling more detailed knowledge of students' progress and quicker identification of problems.
- ✓ Tests can be tailored to match students' abilities and, with formative assessments, it may be possible for content to be varied automatically as the test itself progresses, matching students' weaknesses as they emerge during the test.
- ✓ Being able to regularly evaluate student progress.

Institution

- ✓ The saving of time in marking, and a reduction in subjectivity and human error in the marking process itself. When dealing with large groups of students, the time and resource saving can be of a significant order.
- ✓ Given the computer-based nature of the exercise, substantial reductions in printing costs should be achieved when assessments are updated or altered.

Therefore, it will have a significant aid for the development of education, since examinations determine the extent to which educational objectives have been achieved as well as the extent to which educational institutions have served the needs of community and society.

1.6 Organization of the Thesis

The rest of the thesis is organized as follows. Chapter two discusses about educational assessment and different types of assessment then it will direct to automating this task. The chapter explains background information about Automatic Subjective Question Marking, criteria on assessing subjective questions and various approaches to Automatic Subjective Question Marking system. Chapter three critically reviews related work on Automatic Subjective Question Marking system. The review focuses on approach and performance of the system. It also indicates relevant method to that of Amharic Automatic Subjective Question Marking. Chapter four presents our proposed approaches to Automatic Amharic Subjective Question Marking System and describes the architecture of the proposed system along with the implementation issues. Chapter five presents the empirical results of the proposed system along with their interpretations. Finally, Chapter six concludes the thesis with the research findings, conclusions and future works.

CHAPTER TWO

2. LITERATURE REVIEW

2.1 Overview

In this chapter, a brief overview of the field of subjective question marking is explained. It begins by introducing the broader topic educational assessment, and among the various types of assessment which are considered to be relevant to the research. Assessment, types of subjective question, and method of evaluation are among the topics described in this chapter. Moreover, tools and techniques used to evaluate subjective question are described.

A number of research papers are reviewed to discuss background information related to subjective question assessment is addressed in this chapter. Furthermore, the research investigates state of the art techniques used in the area of subjective question assessment. Automatic subjective question marking is a hot topic of research and hence, there are a lot of works available, but this thesis only present those works whose contribution made a great progress to the automatic assessment for short answer questions.

2.2 Assessment

Assessment is the systematic collection of information about student learning, using the time, knowledge, expertise, and resources available, in order to inform decisions that affect student learning. The purpose of assessment is informed decision-making, including the use of information about student learning [2]. According to Farrell [11], assessing student is used to determine that the intended learning outcomes of the course are being achieved, to provide feedback to students on their learning, enabling them to improve their performance, to motivate students to undertake appropriate work, to support and guide learning, to describe student attainment, informing decisions on progression and awards, to demonstrate that appropriate standards are being maintained, and to evaluate the effectiveness of teaching. Assessment strongly influences students' learning, including what they study, when they study, how much work they do and the approach they take to their learning.

Meaningful and constructive assessments need to make students to think critically and should encourage students' interest in learning. As it is widely acknowledged, assessment drives student

learning and directs student effort, assessment design must be planned accordingly and must be an integral part of course design. Furthermore, assessment tasks influence the direction and quality of student learning. Therefore, to move forward students need to be given more responsibility for assessment processes and must be encouraged to participate in this task.

2.3 Types of Assessment

Though the notion of assessment is generally more complicated and those classifications which are relevant to this thesis work are expressed below. The first classification is based on the purpose of assessment. Accordingly, there are two types of assessment namely, formative and summative [12].

Summative Assessment: - is the process of summing up or checking what has been learned at the end of a particular stage of learning, whether it is a module or a course. The goal of summative assessment is to evaluate student learning at the end of an instructional unit by comparing it against some standard or benchmark. It is used towards and at the end of the instruction period. Teachers document the conclusion of students' learning achievements through tasks that invite students to demonstrate their mastery and knowledge of the course content. As stated by [13], summative assessment data provides teachers with information about how effective teaching strategies have been, time needed for instruction and how to improve teaching for future students. In an educational setting, summative assessments are evaluative and typically used to assign students a course grade.

Formative Assessment: - is part of teaching and learning and is generally carried out throughout a course or project. It is used at the beginning of an instructional period and during the process of instruction as teachers check for student understanding [12]. Diagnostic tools determine what students already know and where there are gaps and misconceptions. Formative assessment also includes assessment ***as*** learning, where students reflect on and monitor their own progress. The information gained guides teachers' decisions in how to enhance teaching and learning. Formative assessment enables students to learn through the process of feedback and opportunities to practice and improve. More specifically, formative assessments help students identify their strengths and weaknesses and target areas that need work and help faculty recognize where students are struggling and address problems immediately. It is also referred to as educative assessment which

is used to aid learning. In an educational setting, formative assessment might be a teacher or peer or the learner, providing feedback on a student's work, and would not necessarily be used for grading purposes rather it is diagnostic.

Summative and formative assessments are often referred to in a learning context as assessment of learning and assessment for learning respectively. Assessment of learning is summative in nature and intended to measure learning outcomes and reports those outcomes to students, parents, and administrators. In addition, Assessment of learning generally occurs at the end of a class, course, semester, or academic year. Assessment for learning is formative in nature and is used by teachers to consider approaches to teaching and next steps for individual learners and the class [14]. As indicated in [13] and others, most of existing assessment procedures, for example, tests, exams, mark and grades have evolved in relation to the needs of summative assessment. Although formative assessment has always been part of the teaching and learning process, as in the case of teachers comment in the paper, it only very recently that it has become an explicit focus for attention. The educational community is much more confused about what constitutes formative assessment and how it may conduct than it is in relation to more familiar forms of assessment practice. So the research has noticed this gap and believes that a lot has to be done in supporting formative assessment through various techniques, considering the benefits to the students' improvement as well as to the educational community at large.

The second classification is based on the type of question included in the exam. Assessment (either summative or formative) is often categorized as either objective or subjective based on type of question. The student's performance is evaluated with the help of Objective and Subjective examinations as per the need of the course. Subjective Examinations include short-answer, long-answer and essay-length answer questions. The answers are evaluated on the basis of a number of parameters like correctness, presence of keywords and style of writing.

Objective Assessment: - is a type of assessment which requires a form of questioning which has a single correct answer. Objective question types include እዉነት/ሐሰት (true/false answers), ምርጫ (multiple choice), and አዛምድ (matching) questions. Objective question can be described as a closed ended question that expects a yes or no, true or false or a choice among several options. For

example, the question “ሰዋሰው ስም ሲሆን፣ አንደኛው ትርጉሙ፤ መሰላል፤ መረማመጃ፤ መወጣጫ፤ መውረጃ ነው።እዉነት/ሐሰት? “is objective type of question [15].

Subjective Assessment: - is a type of assessment which requires a form of questioning which may have more than one correct answer or more than one way of expressing the correct answer. Subjective questions include extended-response questions and essays [16]. Subjective question can be described as open ended question having many right answers. Essays and short answer question are in this category. For example, the question “ሰዋሰው ማለት ምን ማለት ነዉ? አብራራ/ሪ”, “ሰዋሰዉ በቋንቋ ዉስጥ ያለውን ጠቀሜታ በምሳሌ አስረዳ” is short answer subjective type of question. Subjective assessment means assessing answers which have Descriptive, Define or Explain types of question; such examinations are to evaluate the conceptual grasping level of a candidate to how much the concepts are understood in a particular subject.

Assessment through objective questions like multiple choice, fill in the blanks, matching, and true/false is common in educational systems, but this type question format is widely criticized, because it allows students to blindly guess the correct answer and lacks deeper assessment. Moreover, students may also reduce the writing skills. Subjective types of assessment on the other hand can reveal the depth and breadth of student’s knowledge but are much more difficult to grade because of the perceived subjectivity and more effort needed to do the task [16]. One can more effectively assess the learner’s knowledge using descriptive type questions.

Furthermore, objective assessment is well suited to the increasingly popular computerized or online assessment format. Whereas automated technology for analysis and scoring of subjective assessment is still open problem. A lot of work has to be done in subjective assessment considering its importance and the need for improving the assessment process. This thesis focuses on assessing subjective answer type questions. Consequently, the following sub section devoted to discuss on subjective assessment.

2.4 Subjective Question Assessment

Subjective examination has been a major way of evaluating a candidate’s knowledge & understanding about on course or subject in traditional education system for centuries [17]. Every university has its own examination pattern based on subjective examination. According to Amharic Language and Literature department, the questions may be considered in the following forms.

- ምን(what), እንዴት(how), ለምን(why)
- አስረዳ/ግለጽ/Define: explain the meaning and (often) provide an appropriate example
- በምሳለ አስረዳ/Describe / illustrate: present the main points with clear examples that enhance the discussion
- ልዩነቱን ግለጽ/Differentiate / distinguish: present the differences between two things
- በምሳሌ አብራራ/Discuss / explain: present the main points, facts, and details of a topic; give reasons
- ዘርዝር/Enumerate / List / Identify / Outline: write a list of the main points with brief explanations
- የራስህን እይታ ስጥ/Interpret: present your analysis of the topic using facts and reasoning
- አረጋግጥ/Justify / Prove: present evidence and reasons that support the topic
- በአጭር ገለጽ/Summarize: briefly state the main ideas in an organized manner

With subjective assessment, the scores assigned by human raters are intrinsically subjective. Human raters have different characteristics like age, training, mood, prejudices, social, ethnic backgrounds, and reaction to the handwritten style that may influence the way they assign scores. That is, there are always intra-rater and inter-rater variations. For example, the same person scoring the same question at different times may assign different scores (intra-rater variation) depending on their mood or health. Different raters scoring the same question may assign different scores (inter-rater variation). The teachers may be influenced by personal knowledge of different students (positive or negative bias) and the general pressure of the schools to have higher scores (as a competition factor).

2.4.1 Criteria for Subjective Question Assessment

Defining criteria to assess subjection question is usually personal. It depends on purpose of test and type of test assessment. As defined in different literatures assessing subjective questions criteria is set depending upon the purpose of the question, subjective question may be evaluated in one or most of features like (1) mechanics, (2) structure, (3) content and (4) style (5) Vocabulary and Language use (5) Grammar and the scores must reflect these areas [18]. The following sub section define each of criteria and their relevance in assessment.

Content: refers to knowledge of subject and semantic similarity and substantive development of idea which is relevant to assigned topic. According to Abel [15], content is the most important features and focuses on what is said rather that how it is said. Student answer may be related to a specific subject and it must fulfill some content criteria. For example, answer may be related to some area of cell structure in biology and the scores must show that the corresponding contents are covered.

Style: refers to the way in which sentences or group of sentences put to together. It is very subjective and the focus is on how it is structured rather that what is included.

Structure: deals with fluent expression, ideas clearly supported flow of ideas and have logical sequencing of statements.

Vocabulary and Language use: in this case the focus is on knowledge of vocabulary or idiom choice.

Grammar usage: deals with complex sentences, errors of agreement, tense, number, word order, articles, pronouns and prepositions.

Mechanics: refers to the correctness of a paper: complete sentences, correct punctuation, accurate word choice. The mechanics represent the grammar and spelling requirements. Correct spelling and grammar are usually basic requirements in all educational subjective question assessment.

Plagiarism: deals with similarity between student answers. The aim is to detect whether student 'A' answer is copy of student 'B' or not.

Usually, based on the above features a specific criterion is prepared to perform the evaluation process in any language.

2.4.2 Automatic Subjective Question Marking (SQM)

The manual system for evaluation of subjective answers for technical subjects involves a lot of time and effort of the evaluator. Assessing through computerized intelligent techniques ensures uniformity in marking as the same inference mechanism is used for all the students. Subjective answers are evaluated on the basis of content and style of writing. For technical subjects, emphasis is more on content. If standard keywords are found in students' answer, then answer is correct.

However, we cannot mark the answers by just counting the number of keywords. A more wholesome approach is required, which can evaluate on the basis of not only keyword presence but the semantic relationship between words and concepts. Starting from early work of PEG [19], different works has been researched to deal with the aforementioned problem. The following sub section discuss some of approaches.

2.4.3 Approaches to Automatic Subjective Question Marking

Literature of Automatic Subjective Question Marking Systems is vast and, there have been many publications in the last decade in particular. Besides, there have been a considerable amount of different classifications of techniques to automatically assess subjective question or free text answers [18]. Table 2-1, summarizes list of most common Automatic Subjective Question Marking Systems and their respective approaches.

Table 2-1*: Survey of Subjective Evaluation techniques: Correlation metrics and evaluation dataset used is may vary from approach to approach*

Year	**Author**	**Tool**	**Technique**	**Results (Correlation with human)**
1998	Burstein	E-rater	Hybrid of features	84-94%
2001	Callear	Automated Text Marker	Conceptual Dependency	None
2002	Rudner	Betsy	Bays Theorem	80%
2003	Landauer	Inteliigent Essay Assessor	Latent Semantic Processing	59-88%
2005	Perez	Atenea	BiLingual Evaluation Understudy, LSA	50%
2008	Kakkonen	Automatic essay Assessor	LSA, Probabilistic LSA, Latent Dirichlet Allocation	LSA better than rest (not defined)
2008	Li bin		K-Nearest Neighbor	76%
2010	Islam		Generalized Latent Semantic Analysis	86-96%
2012	Sukkarieh	C-rater	Maximum Entropy	80%
2016	Shourya	An Iterative Transfer Learning Based Ensemble Technique	Ensemble of two classifiers (First classifier use TFIDF, then second classifier predict correlation of texts	1.04 (MAE)

Year	Author	Tool	Technique	Results (Correlation with human)
		for Automatic Short Answer Grading	using output of first and other features)	
2016	M. Syamala Devi and Himani Mittal	Machine Learning Techniques With Ontology For Subjective Answer Evaluation (Both Essay and short answer)	MaxEnt with domain ontology (hybrid approach)	90%
2016	Dimitrios *et al.*,	Automatic Text Scoring Using Neural Networks (Essay)	score-specific word embeddings (SSWEs) + 2 Layer Bi-directional LSTM	96% (2.4 MAE)

It can be seen from the above table that various approaches are used in the development of automated essay and short answer assessment system. When the computer technology advances the approaches used to develop the system also advances, as a result there are now various types of approaches. It is not the aim of this thesis to review all the approaches rather, we give due emphasis to those approaches whose contribution made a great progress to the Automatic Subjective Question Marking field. For the simplicity, the thesis would like to classify the various approaches as in to four general categories as Machine Learning, Text Similarity, Deep Learning, and Ontology based. Deep Learning based approach is given due consideration and a detail description is given as thesis body of knowledge depends on deep learning approach.

1. Machine Learning Approach for SQM

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can change when exposed to new data. Machine learning systems typically utilize some number of measurements extracted from natural language processing techniques and similar, which are then combined into a single grade or score using a classification or regression

model. This can be supported by a machine learning toolkit such as Weka, LIBSVM, etc. Features involving bag-of-words and n-grams are typical of this category, as are decision trees and support vector machines as representative learning algorithms. Implicitly or explicitly, previous work has primarily treated text scoring as a supervised text classification task, and has utilized a large selection of techniques, ranging from the use of syntactic parsers, via vector semantics combined with dimensionality reduction, to generative and discriminative machine learning.

Vast research done using this approach to deal with subjective question evaluation problem [21]. All works relay on hand crafted lexical, syntactic and semantic features. As multiple factors influence the quality of texts, Machine Learning based systems typically exploit a large range of textual features that correspond to different properties of text, such as grammar, vocabulary, style, topic relevance, and discourse coherence and cohesion. In addition to lexical and part-of-speech (POS) *n*-grams, linguistically deeper features such as types of syntactic constructions, grammatical relations and measures of sentence complexity are among some of the properties that form an SQM system's internal marking criteria. The final representation of a text typically consists of a vector of features that have been manually selected and tuned to predict a score on a marking scale. Popular machine learning techniques such as SVM, MaxEnt, MLP, RF, Decision Tree, etc. are used to score specific student grade based on labeled answer provided by instructor.

2. Text Similarity Approach for SQM

This approach addresses the grading problem from a text similarity perspective and examine the usefulness of various text to-text semantic similarity measures for automatically grading student answers. Text similarity measures play an increasingly important role in text related research and applications in tasks such as information retrieval, text classification, document clustering, topic detection, topic tracking, questions generation, question answering, essay scoring, short answer scoring, machine translation, text summarization and others [22].

Finding similarity between words is a fundamental part of text similarity which is then used as a primary stage for sentence, paragraph and document similarities. Words can be similar in two ways lexically and semantically [23]. Words are similar lexically if they have a similar character sequence. Words are similar semantically if they have the same thing, are opposite of each other, used in the same way, used in the same context and one is a type of another. Lexical similarity is introduced in this approach though different String-Based algorithms, Semantic similarity is

introduced through Corpus-Based and Knowledge-Based algorithms. String-Based measures operate on string sequences and character composition. A string metric is a metric that measures similarity or dissimilarity (distance) between two text strings for approximate string matching or comparison. Corpus-Based similarity is a semantic similarity measure that determines the similarity between words according to information gained from large corpora using LSA, ESA, Distributional Similarity Co-occurrences (DISCO), etc. [24]. Knowledge-Based similarity is a semantic similarity measure that determines the degree of similarity between words using information derived from semantic networks such as WordNet, Wikipedia, etc. [23]. To score student answer, this approach uses some heuristics that combine different similarity results obtained from measuring string, corpus-based and knowledge based similarity approaches.

3. Ontology Based Approach for SQM

Ontologies are applied in different approach for question marking process. One is using ontology as knowledge base and other is ontology mapping. The following sub-section discuss the two approaches in detail.

A. Knowledge Representation

As a branch of symbolic Artificial Intelligence, knowledge representation and reasoning aims at designing computer systems that reason about a machine-interpretable representation of the world, similar to human reasoning. A knowledge-based system maintains a knowledge base which stores the symbols of the computational model in form of statements about the domain, and it performs reasoning by manipulating these symbols. Domain ontology is one of knowledge representation technique and used in different domain [25]. It specifies the concepts, and the relationships between concepts, in a particular subject area rather than specifying only generic concepts, as found in an upper ontology. A domain ontology models the information known about a particular subject and therefore should closely match the level of information found in a textbook on that subject.

In this approach concepts extracted from student answer is mapped to concepts of model answer. Ontology construction for student and model answer is not required. The ontology is extracted from domain course ontology for the concept that has relation to model answer or question.

Similarity between student answer concept and extracted model answer is calculated using text semantic similarity technique or given to classifier to predict correlation between two texts [49].

B. Ontology Mapping

This approach requires two ontologies to map or align concepts. Ontology mapping seeks to find semantic correspondences between similar elements of different ontologies [26]. We first model ontology for both model or correct answer and student answer using manual or automatic ontology learning techniques. Then we try to align each concept in ontology to other. Given two ontologies O1 and O2, mapping one ontology onto another means that for each entity (concept C, relation R, or instance I) in ontology O1, we try to find a corresponding entity, which has the same or similar semantics, in ontology O2 and vice versa. Works done with this approach follow first extract machine understandable format such as RDF, RDFS, OWL, etc. from two text then map two created ontologies.

4. Deep Learning Approach for SQM

Although current approaches to scoring, such as regression and ranking, have been shown to achieve performance that is indistinguishable from that of human examiners, there is substantial manual effort involved in reaching these results on different domains, genres, prompts and so forth. Linguistic features intended to capture the aspects of writing to be assessed are hand-selected and tuned for specific domains. In order to perform well on different data, separate models with distinct feature sets are typically tuned.

Recent advances in deep learning reveal another promising direction to solve this problem. Instead of discrete features and logics, continuous representation of the sentence is more robust to unseen features without sacrificing performance [27]. Success in unsupervised approaches for learning embedding's for textual entities from large text corpora altered the way NLP problems are studied today. This embedding's have been shown to capture syntactic and semantic information as well as higher level analogical structure. These methods have been adopted to learn vector representations of sentences, paragraphs and entire documents. Embedding based approaches allow models to be trained end-to-end from scratch with no handcrafting.

Deep neural networks are known for automatically learning useful features from data, with lower layers learning basic feature detectors and upper levels learning more high-level abstract features

[28]. Recurrent neural networks and convolutional neural networks are well-suited for modeling the compositionality of language and have been shown to perform very well on the task of language modeling.

Deep learning approaches use word vectors as knowledge to encode sentence. Main components that every deep learning models utilize is used for subjective question assessment also. General architectural model for deep learning neural network based subjective question assessment system is depicted in Figure 2.1 below.

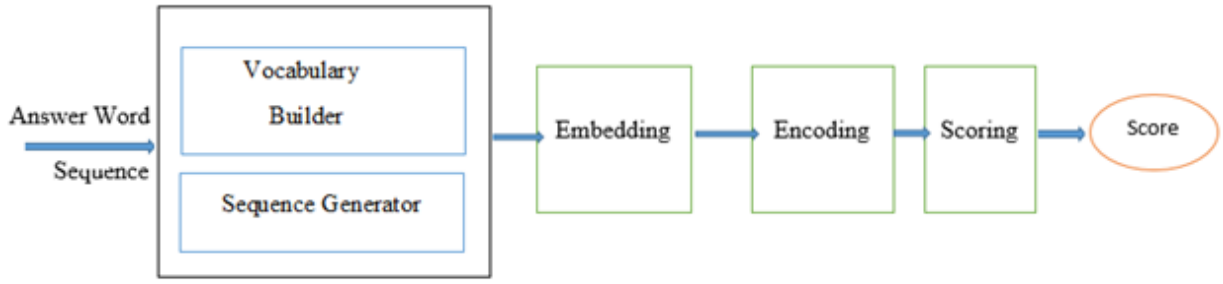


***Figure 2-1**: General Architectural Model for Deep Learning based subjective question marking*

Neural Network work with continuous values than discrete input. But, the task input is natural language so we have discrete values (sequence of words). We should first change such sequence in the way applicable for neural network. To make the data sequence compatible with the network, the first task is transposing this discrete sequence to continuous value (integers). To do so, vocabulary of words to their indices is created from training data. Using created vocabulary, we build integer sequence by replacing each words in answer to their respective indices. Next we represent our data sequence in to vector what we call it word representation or embedding. This module is responsible to represent input into meaningful feature representations and are integral part of any neural network based models. In the following section we will brief remaining components in detail.

Feature Representation

For NLP tasks, we know that all the information required to successfully perform the task is encoded in the data (i.e., sequence of words or characters). To work with neural network, we need to represent our input into *d* dimensional vector. When dealing with natural language, the input

encodes features such words, part-of-speech tags or other linguistic information. The biggest jump when moving from sparse input with linear models to neural network model is to stop representing each feature as a unique dimension (one-hot representation) and representing them instead as a dense vector.

One-hot Representation

NLP systems traditionally treat words as discrete atomic symbols as one-hot representation of word index. One-hot sparse representation is a technique that treat words as atomic units, there is no notion of similarity between words as this are represented as indices in a vocabulary. The method represents only one element as 1 and the other elements are 0 in the vector. These encodings are arbitrary, and provide no useful information to the system regarding the relationships that may exist between the individual symbols. If we represent 'ድመት (cat)' and 'ዉሻ (dog)' in one-hot representation, the occurrence of *cat* does not tell us anything about the occurrence of *dog*. However, in the dense vector representation the learned vector for *cat* may be similar to the learned vector from *dog* allowing the model to share statistical strength between the two events.

Vector Representation

Vector representation also called embedding is used to extract meaning from text to understand natural language. Word embedding is a learned dense representation for words where words with similar meaning have similar representation. So instead of using one-to-one mapping between an element in the vector (one-hot vector) and a word, the representation of a word is spread across all of the elements in the vector, and each element in the vector contributes to the definition of many words. These distributed representations encode shades of meaning across their dimensions, allowing for two words to have multiple, real-valued relationships encoded in a single representation. We can use these word vectors as meaning bearer features for various supervised NLP tasks [33]. The models do not need labels in order to create meaningful representations. This is useful, since most data in the real world is unlabeled. If the model is given enough training data, it produces word vectors with intriguing characteristics. Words with similar meanings appear in clusters, and clusters are spaced such that some word relationships can be easily inferred.

Many different types of models were proposed for estimating continuous representations of words, including the well-known Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA). The different approaches that leverage this principle can be categorized into two categories: count-based methods (LSA, GloVe [29]), and predictive methods (e.g., neural probabilistic language models [29, 32]).

Count-based methods compute the statistics of how often some word co-occurs with its neighbor words in a large text corpus, and then map these count-statistics down to a small, dense vector for each word. From count based models GloVe recently gain better attention as predictive models. GloVe (Global Vectors) is an unsupervised learning algorithm for obtaining vector representations for words [29]. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. Glove is similar with Word2Vec model except vector representation used. With word2vec you stream through n-grams of words, attempting to train a neural network to predict the n-th word given words [1… n-1] or the other way round. The end result is a matrix of word vectors or context vectors respectively. With Glove, you build a co-occurrence matrix for the entire corpus first, then factorize it to yield matrices for word vectors and context vectors.

Predictive models such as Word2vec [31] and FastText [32] directly try to predict a word from its neighbors in terms of learned small, dense *embedding vectors* (considered parameters of the model). Predictive model embed word in a continuous vector space where semantically similar words are mapped to nearby points significantly changed the way NLP does. With predictive models, each word is represented by a distribution of weights across those elements. The weights associated with each word becomes that word's dense vector embedding. This predictive ability of predictive models over count based models and memory consumption of count based model is usually taken as criticizing point for both predictive and count based models.

Inspired by their power on representing words, we use predictive models to create word vectors we will use for both English and Amharic word vectors that are used as only feature that we provide to our neural model to score subjective question. In the following sub section, we give detail review on predictive word embedding models.

Neural Word Embedding

Word embeddings are one of the few currently successful applications of unsupervised learning. Their main benefit arguably is that they don't require expensive annotation, but can be derived from large unannotated corpora that are readily available. The term word embeddings is coined in 2003, but the eventual popularization of word embedding can be attributed to Mikolov *et al.*, [31] in 2013 who created word2vec, a toolkit that allows the seamless training and use of English pre-trained embedding. In 2016, Facebook released another predictive model called FastText that represent word information through sub-words or character n-grams. FastText is extension of word2vec model by extending character n-gram feature.

Word2Vec

Word2Vec is the name given to a class of neural network models with two layer that, given an unlabeled training corpus, produce a vector for each word in the corpus that encodes its semantic information. Word2vec can utilize either of two model architectures to produce a distributed representation of words: continuous bag-of-words (CBOW) or continuous skip-gram. In the continuous bag-of-words architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction (bag-of-words assumption). In the continuous skip-gram architecture, the model uses the current word to predict the surrounding window of context words. The skip-gram architecture weighs nearby context words more heavily than more distant context words. CBOW is faster while skip-gram is slower but does a better job for infrequent words [31]. Word2vec uses a single hidden layer for both architecture, fully connected neural network as shown below in Figure 2.2 and 2.3. The neurons in the hidden layer are all linear neurons. The input layer is set to have as many neurons as there are words in the vocabulary for training. The hidden layer size is set to the dimensionality of the resulting word vectors. The size of the output layer is same as the input layer. Thus, assuming that the vocabulary for learning word vectors consists of V words and N to be the dimension of word vectors, the input to hidden layer connections can be represented by matrix W of size $V \times N$ with each row representing a vocabulary word. In same way, the connections from hidden layer to output layer can be described by matrix W' of size $N \times V$. In this case, each column of W'output matrix represents a word from the given vocabulary.

The one-hot encoded input vectors are connected to the hidden layer via a weight matrix and the hidden layer is connected to the output layer via a weight matrix. The weights between the input layer and the output layer can be represented by a $V \times N$ matrix W. Each row of W is the N-dimension vector representation V_w of the associated word of the input layer. That is hidden layer of the network. The word vectors W and W' are learned via backpropagation and stochastic gradient descent. Finally, the output layer is output word in the training example which is also one-hot encoded.

Continuous Bag-of-Word Model (CBOW)

While a language model is only able to look at the past words for its predictions, as it is evaluated on its ability to predict each next word in the corpus, a model that just aims to generate accurate word embeddings does not suffer from this restriction. Mikolov et al., [31] thus use both the n words before and after the target word w_t to predict it as depicted in Figure 2.2 below.

In Word2Vec framework, every word W in the dictionary V is mapped to a vector $w(x)$, which is a column in the matrix W (matrix W is randomly initialized). The CBOW model predicts a word $w(x)$ using its context $w(x - n),\cdots, w(x - 1), w(x + 1),\cdots, w(x + n)$. CBOW described in Figure 2.2 below is implemented in the following steps.

Step 1: Generate one hot vectors for the input context of size C.

For each alphabetically sorted unique vocabulary terms as target word, we create one hot vector of size C. i.e., for a given context word, only one out of V units,$\{x_1 \cdots x_v\}$ will be 1, and all other units are 0.

Step 2: Compute the output of the hidden layer.

Hidden layer is based one hot encoded input layer. When computing the hidden layer output, instead of directly copying the input vector of the input context word, the CBOW model takes the average of the vectors of the input context words, and use the product of the input→hidden weight matrix W and the average vector as the output.

$$h = \frac{1}{C} W^T (x_1 + x_2 + \cdots + x_c) \qquad (2.1)$$

$$= \frac{1}{C}(v_{w1} + v_{w2} + \cdots + v_{wc})^{\mathrm{T}} \quad (2.2)$$

, where *C* is the number of words in context, $w_1, \cdots, w_c$ are the words in context and v_w is the input vector of word *w* (is product of its weight vector to input one hot vector *x*).

Step 3: Compute the inputs to each node in the output layer

Next we compute score of each input vectors of output layer as

$$u_j = {v'_{w_j}}^T h \quad (2.3)$$

, where v'_{w_j} is the j[th] column of the output matrix W'. And finally we compute the output of the output layer.

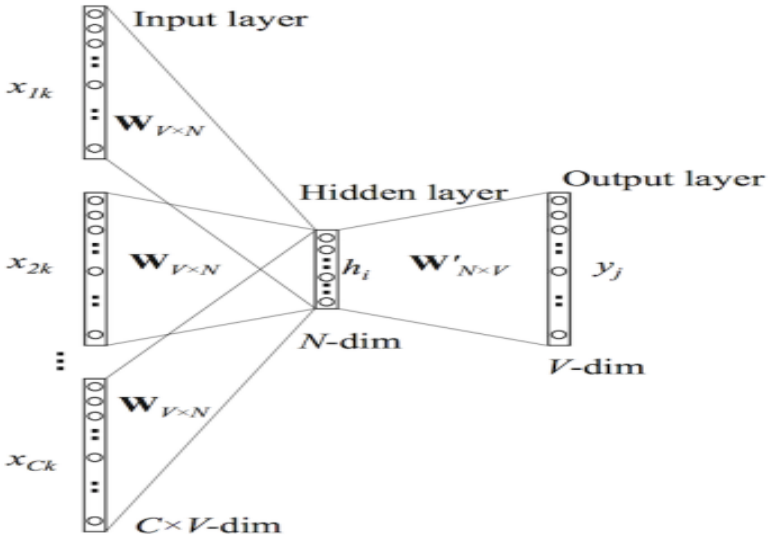


***Figure 2-2**:Word2Vec [31] CBOW Model*

Step 4: Compute probability distribution of target word.

Finally, the output y_j i.e., the j[th] unit in output layer, is obtained by passing the input u_j through the soft-max function. The Softmax log-linear classification model used to calculate the probability distribution of the target word given a specific context is:

$$p(w_y|w_1, \cdots, w_c) = \frac{exp(u_j)}{\sum_{j=1}^{v} exp(u'_j)} \quad (2.4)$$

Skip-gram Model

Instead of using the surrounding words to predict the center word as with CBOW, skip-gram uses the center word to predict the surrounding words as can be seen as opposite of CBOW model as shown in Figure 2-3 below. The input of the skip-gram model is a single target word and the output is the words in w_I's context $\{w_o, 1, \cdots w_o, C\}$ defined by a word window of size. We still use v_{WI} to denote the input vector of the only word on the input layer, and thus we have the same definition of the hidden→layer outputs h as in CBOW, which means h is simply copying (and transposing) a row of the input→hidden weight matrix, W, associated with the input word .

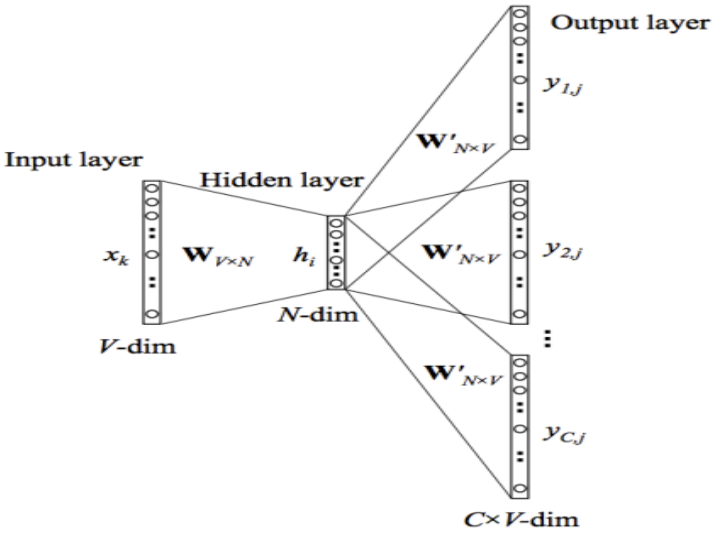


Figure 2-3*:Word2Vec [26] Skip-gram model.*

In the above model ***x*** represents the one-hot encoded vector corresponding to the input word in the training instance and $\{y_1, \cdots y_C\}$ are the one-hot encoded vectors corresponding to the output words in the training instance. The *V x N* matrix *W* is the weight matrix between the input layer and hidden layer whose i^{th} row represents the weights corresponding to the i^{th} word in the vocabulary. This weight matrix *W* is what we are interested in learning because it contains the vector encodings of all of the words in our vocabulary (as its rows). Each output word vector also has an associated *N x V* output matrix W'. There is also a hidden layer consisting of *N* nodes (the exact size of *N* is a training parameter). We know that the input to a unit in the hidden layer $h_i$is simply the weighted sum of its inputs. Since the input vector *X* is one-hot encoded, the weights coming from the nonzero

element will be the only ones contributing to the hidden layer. Therefore, for the input *X* with $X_k = 1$ and $X_{k'} = 0$ for all $k' \neq k$ the outputs of the hidden layer will be equivalent to the k^{th} row of *W*.

$$h = x^T W = W_{(k,.)} := V_{w_I} \tag{2.5}$$

In the same way, the inputs to each *C* x *V* of the output nodes is computed by the weighted sum of its inputs. Therefore, the input to the j[th] node of the c[th] output word is

$$u_{c_j} = {v'_{w_j}}^T h \tag{2.6}$$

However we can observe that the output layers for each output word share the same weights therefore $u_{c_j} = u_j$. We can finally compute the output of the j[th] node of the c[th] output word via the Softmax function which produces a multinomial distribution.

$$p(w_{c,j} = w_{0,c} | w_I) = y_{c,j} = \frac{exp(u_{c,j})}{\sum_{j=1}^{v} exp(u_j)} \tag{2.7}$$

In simple term, this value is the probability that the output of the j[th] node of the c[th] output word is equal to the actual value of the j[th] index of the c[th] output vector (which is one-hot encoded).

FastText

Motivated by, Google's word2vec embeddings, in 2016 Facebook released an embedding model that recently attracted a great deal from the machine learning community especially for morphologically rich languages called FastText [32]. The main goal of the FastText embeddings is to take into account the internal structure of words while learning word representations; this is especially useful for morphologically rich languages like Amharic, where otherwise the representations for different morphological forms of words would be learnt independently. The limitation becomes even more important when these words occur rarely unless we use external language dependent tools such as morphological analyzers. The semantic and syntactic information of words that is contained in these vectors make them powerful features for NLP tasks.

One issue FastText criticized is its memory consumption to construct character n-gram level, it takes longer to generate FastText embeddings compared to word2vec model. However, word2vec and GloVe treats each word in corpus like an atomic entity and generates a vector for each word. For example, the word ዘገባ, ለዘገባ, በዘገባ, ስለዘገባ, etc are treated as atomic unless we apply

morphology analysis before providing dataset to model. For morphologically rich languages such as Turkic, Arabic, Chinese, Amharic, etc. treating each varieties of words as atomic unit not effective approach. In contrast, FastText treats each word as composed of character n-grams. So the vector for a word is made of the sum of this character n grams. For example, the word vector "ዘገባ" is a sum of the vectors of the n-grams "<ዘገ", "ዘገባ"," ገባ">. With this manifest it benefits to generate better word embeddings for rare words. Moreover, FastText predict vector for out of vocabulary words from its character n-grams even if word doesn't appear in training corpus. In contrast both Word2vec and Glove leave unseen words as out-of-vocabulary words. So, with this intuition, we proposed to use FastText as word vector generating model for SQM.

As it is extension to Word2Vec [31] model, FastText also has two architectures for computing word representations called Skip-gram and CBOW (**c**ontinuous-bag-of-words). The Skip-gram model learns to predict a target word given a nearby word. On the other hand, the CBOW model predicts the target word according to its context. For instance, given the sentence "የህዝብ ግንኙነት ለህዝብ መረጃ በመስጠት የህዝብን አዝማሚያዎችና ተግባራት ለማስተካከል የሚከሄድ የማሳመን ሥራ ነዉ" and the target word "ተግባራት". The Skip-gram model predicts the target using a random close-by word, like "አዝማሚያዎችና" *or* "የማሳመን". Whereas the CBOW model takes all the words in a surrounding window, like {የህዝብን: አዝማሚያዎችና, ለማስተካከል: የሚከሄድ}, and uses the sum of their vectors to predict the target word "ተግባራት". At the time of training, FastText trains by sliding a window over the input text and either learning the target word from the remaining context (CBOW), or all the context words from the target word ("Skip-gram"), and learning can be viewed as a series of updates to a neural network with two layers of weights and three tiers of neurons, in which the outer layer has one neuron for each word in the vocabulary and the hidden layer has as many neurons as there are dimensions in the embedding space. In this way, it is similar to Word2Vec. Unlike word2vec, FastText also learn vectors for sub-parts of words called character n-grams ensuring that e.g., the words "የህዝብን", "የህዝብ"," ህዝብ" and "ህዝብን" all fall into same dimension in vector space, even if they tend to appear in different contexts. This feature enhances learning on heavily inflected languages. Despite use of sub-word information, training FastText is same as Word2Vec for both CBOW and Skip-gram models.

The key difference between FastText and Word2Vec is the use of n-grams. Word2Vec learns vectors only for complete words found in the training corpus. FastText, on the other hand, learns vectors for the n-grams that are found within each word, as well as each complete word. At each training step in FastText, the mean of the target word vector and its component n-gram vectors are used for training. The adjustment that is calculated from the error is then used uniformly to update each of the vectors that were combined to form the target. This adds a lot of additional computation to the training step. At each point, a word needs to sum and average its n-gram component parts. The trade-off is a set of word-vectors that contain embedded sub-word information. These vectors have been shown to be more accurate than Word2Vec vectors by a number of different measures. Treating character n-gram manifests FastText the following advantage over Word2Vec:

- ✓ Generate better word embeddings for rare words (even if words are rare their character n grams are still shared with other words - hence the embeddings can still be good). This is simply because, in word2vec a rare word (e.g., 10 occurrences) has fewer neighbors to be pulled by, in comparison to a word that occurs 100 times whereas FastText has more neighbor context words and hence is pulled more often resulting in better word vectors.

- ✓ Out of vocabulary words - they can construct the vector for a word from its character n grams even if word doesn't appear in training corpus. Both Word2vec and Glove can't.

It is not clear to use which embedding in what situation, but based on comparative study done by [32], FastText built on the top of Word2Vec do significantly better on morphology dependent tasks. In contrast, word2vec model seems to perform better on semantic tasks, since words in semantic analogies are unrelated to their char n-grams, and the added information from irrelevant char n-grams worsens the embeddings. But, in all task FastText significantly outperformed Word2vec models for morphologically rich languages. In this thesis as Amharic is one of morphologically rich language we will be using FastText model for Amharic word vector. Moreover, FastText allows us to cluster words with spelling errors to their semantically similar words which are common in student written exams. Using those word vectors, embedding module represent word index sequence into sequence of vectors.

Other than words character level language model also performs comparable result to represent text meaning. Character model is used with NLP in two approach. One is encoding entire text as sequence of character and the other is enhancing word vectors by concatenation sub-word information of each words to their vector. In practice the later outperformed the former approach [48]. In languages such as Amharic, a word is usually composed of several characters and contains rich internal information since semantic meaning of a word is also related to the meanings of its composing characters. Enhancing word embedding with character embedding may improve the embedding capacity of word embeddings in morphologically rich language such as Amharic [48]. It allows us to tackle mechanics problem (i.e., spelling errors and heterogeneity in word formation) happen in writing. Character encoding can be performed either RNN or CNN. As stated by Zhang et al., [61] using CNN model for encoding character has advantage on treating morphemes due to the property of CNN in extracting informative feature. Moreover, it is recommended technique to represent out-of-vocabulary words with their character level information than treating them as zero encoded or with dummy randomized vectors [48, 61].

Answer Encoding

Encoding text is semantic representation of the word in the text sequence that hold global contextual features of the whole text. Embedding module represents words independently, but we need how each words contributed to meaning of sentence or whole answer to score entire answer. Several techniques are used ranging from simple vector averaging to recurrent neural networks. The input to this step is vector representation i.e., whether one-hot encoded or word embedding vectors[2]. Averaging vectors tries to get cumulative context of a sentence vectors by summing all word vectors of words in a sentence or use word frequencies (can be also inverse document frequency) as weight and multiply to their vectors to minimize effect of frequent words (i.e., stop words) [55]. Finally, to get sentence level context summation of vectors is divided into number of words in a sentence. One benefit of averaging vectors is its fastness to represent sentence. However, averaging does not consider word order in a sentence. Such model looks existence of words than their location. If word exist in both model and student answer statement, the approach reward highest value. In practice, word order changes the meaning because of subject and object

[2] Practical approach is representing words to their meaning distribution using pre-trained word vectors such as FastText or word2vec. For tasks with high probability to spelling error and out-of-vocabulary words FastText is preferable.

change. The possible encoder to use for tasks such as assessment that require word sequence is neural sequence encoder.

Popular deep learning models such as convolutional neural networks (CNN), recurrent neural networks (RNN), and recursive neural networks (RecursiveNN) are used to represent contextual representation of input answer to fixed-length high-level context dense vectors usually called sentence matrix [52]. The job encoder network is to read the input word sequence to sequence encoder model and generate a fixed-dimensional context vector for the entire sequence.

Convolutional Neural Networks (CNN)

For some NLP task such as sentiment analysis we need to predict on availability of some salient information than sequential representation by scarifying order of words. Convolutional neural networks (CNNs) [63] architecture is an elegant and robust solution to model such problem [60]. A convolutional neural network is designed to identify indicative local predictors in a large structure, and combine them to produce a fixed size vector representation of the structure, capturing these local aspects that are most informative for the prediction task at hand. The main idea behind a convolution and pooling architecture for language tasks is to apply a non-linear (learned) function over each instantiation of a *k*-word sliding window over the sentence. This function (also called "filter") transforms a window of *k* words into a *d* dimensional vector that captures important properties of the words in the window (each dimension is sometimes referred to in the literature as a "channel"). Then, a "pooling" operation is used to combine the vectors resulting from the different windows into a single *d*-dimensional vector, by taking the max (also known as MaxPooling) or the average (also called AveragePooling) value observed in each of the *d* channels over the different windows. The intention is to focus on the most important "features" in the sentence, regardless of their location. The *d*-dimensional vector is then fed further into a network that is used for prediction. The gradients that are propagated back from the network's loss during the training process are used to tune the parameters of the filter function to highlight the aspects of the data that are important for the task the network is trained for. Intuitively, when the sliding window is run over a sequence, the filter function learns to identify informative k-grams.

Recursive Neural Networks

The recursive neural network (RecursiveNN) is a generalization of the RNN from sequences to (binary) trees popularized in late 2014 [30]. Much like the RNN encodes each sentence prefix as a state vector, the RecursiveNN encodes each tree-node as a state vector in $\mathbb{R}^d$. We can then use these state vectors either to predict values of the corresponding nodes, assign quality values to each node, or as a semantic representation of the spans rooted at the nodes. The main intuition behind the recursive neural networks is that each subtree is represented as a *d* dimensional vector, and the representation of a node *p* with children *c*1 and *c*2 is a function of the representation of the nodes: $vec(p) = f(vec(c_1), vec(c_2))$, where f is a composition function taking two *d*-dimensional vectors and returning a single *d*-dimensional vector. Context h_i is used to encode the entire sequence$x_1 : i$, the RecursiveNN state associated with a tree node *p* encodes the entire subtree rooted at *p*. It is on debate whether sentence structure is recursive or not, but if we have parsed data RecursiveNN can do well for structure dependent NLP tasks.

From survey [60], for sequence dependent tasks CNNs are not preferable as they skip order of sequence and considered good at extracting local and position-invariant features. However, for assessment we need to give attention for text structure in addition to content level contexts. Recursive NN can do well with sequence dependent tasks, but such model require external tools such as syntactic or dependency parser to create parsed sentences [30]. Recurrent Neural Networks are deep learning model that are suitable to represent sequence dependent tasks that require context dependencies and sequence order [60]. As subjective question marking is one of sequence dependent task the thesis use RNN to build abstraction of input answer by analyzing each words sequentially. In the following subsection we will discuss the detail how RNN works and its variants.

Recurrent Neural Network (RNN)

When dealing with language data, it is very common to work with sequences, such as words (sequences of characters), sentences (sequences of words) and documents (sequence of sentences or paragraphs). Recurrent Neural Networks is initially proposed by Elman in 1990 [65] and explored for use in language modeling by Mikolov in 2012 [66] are a family of neural networks designed specifically for sequential data processing and allow representing arbitrarily sized

structured inputs in a fixed-size vector, while paying attention to the structured properties of the input. RNNs are called recurrent because they perform the same task for every words of a sequence, with the output being depended on the previous operations. Recurrent Neural Networks have become the common approach to sequence learning and mapping problems in recent times [34]. The Sequence to sequence mapping [34], as well as several of its variants have fueled RNN based approaches to a wide variety of problems including language modeling, language generation, machine translation, question answering, automated essay scoring and many others. The intuition behind is to predict next word given previous word information for sentence level task and predicting next sentence given previous sentence vector in a document for document level task. To formalize this chain assumption let we want to compute the likelihood of the sentence "በአማርኛ ከሚታወቁት የዜማ ቅኝት አይነቶች አንዱ አምባሰል ነዉ", we need to estimate the following probabilities:

$$p(\text{በአማርኛ }), p(\text{ከሚታወቁት}|\text{በአማርኛ }), p(\text{የዜማ}|\ \text{በአማርኛ ከሚታወቁት}), \cdots,$$

$$p(\text{ነዉ}|\text{በአማርኛ ከሚታወቁት የዜማ ቅኝት አይነቶች አንዱ አምባሰል})$$

We know that we have word vectors that we discussed on previous section. Each words are represented to their word vectors that give how the word is related to entire vocabulary word in a vector space; with RNN we first, initialize the memory vector *h* to zero. In the first time step (denoted by zero) the input to the RNN unit is special token <\s> which symbolizes the beginning of a sentence. As an output, we get the probability of every possible word in the vocabulary given the start of sentence token. The memory vector gets updated in this same operation and sent to the next time step. Now we repeat the procedure for time step 1 in which በአማርኛ is the input of the cell, h_1 is the memory state which contains information about the past and $p(w_2| <\s > \text{በአማርኛ })$is the output. The following Figure 2-4 illustrate how Vanilla RNN compute score of the entire sentence.

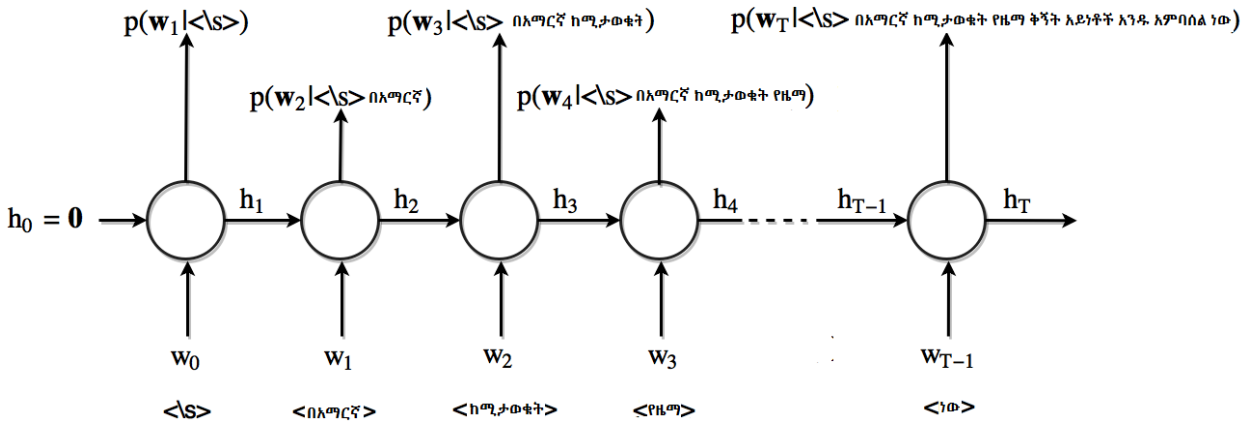


***Figure 2-4**: Figure that depicts how RNN works to get context of sentence*

In general, at each time step, we seek to estimate a probability distribution over all the possible next words in the vocabulary *V* given the previous words. The output layer of the RNN is then a Softmax layer which returns a vector of size |*V*| whose *i-th* element indicates the predicted probability of the word V_i being the next word to appear in the sentence. More precisely, the recurrent neural network computes the following function, with a *Softmax* output layer predicting the conditional probability of input x_i given the sequence of length k $[x_1, x_2, \cdots, x_k]$, which guarantees positive probabilities summing to 1:

$$e_t = \frac{exp(h_t)}{\sum_1 \exp(h_k)} \; for \; t = 1, \cdots, k \tag{2.8}$$

, whereas e_t is the resulting vector of non-negative real numbers, h_t is the memory hidden state is calculated as:

$$h_t = f(Ux_t + Wh_{t-1} + b) \tag{2.9}$$

, where *U* and W are learnable weights, $h_{t-1}$is previous hidden state output vector, *b* is bias also learned, x_t is current input vector in a sequence, h_t is the output at step t i.e., used to predict the next word in a sentence it would be a vector of probabilities across our vocabulary, and .f is non-linearity function. There is no good theory as to which non-linearity to apply in which conditions, and choosing the correct non-linearity for a given task is for the most part an empirical question. But, the common nonlinearities from the literature used in NLP applications are sigmoid, tanh, hard-tanh and the rectified linear unit (ReLu) [52]. However, because of its easiness to implement and cheaper computation in back-propagation as ReLu not susceptible for vanishing gradient

problem as compared to tanh and sigmoid, to efficiently train more deep neural network ReLu is commonly used in recent NLP applications [64].

The weight matrices *U* and *W* are filters that determine how much importance to accord to both the present input and the past hidden state. The error they generate will return via backpropagation and be used to adjust their weights until error can't go any lower. To update learnable weights *U* and *W* we apply gradient update called backpropagation through time (BPT). The goal of the BPT is to modify the weights of a RNN in order to minimize the error (cross entropy error or loss) of the network outputs compared to expected output in response to corresponding inputs. BPT can be directly applied to Figure 2-4, the computational graph of the unfolded network, to compute the derivative of a total error (for example, the log-probability of generating the right sequence of outputs) with respect to all the states h_t and all the parameters. The intuition is we compare predicted output $e_t$with actual word in a vocabulary and calculate error. Then if actual is different from predicted we adjust weights and repeat the same process. The loss function *L* for a given sequence is the negative log probability that the model assigns to the correct output is given by:

$$L(x) = -\sum_{t} \log p_{model}(w_t = x_{t+1}) = -\sum_{t} \log o_t[x_{t+1}] \qquad (2.10)$$

, where $o_t[x_{t+1}]$is the element of the output Softmax corresponding to the real wordx_{t+1}.
With the loss defined and given that the whole system is differentiable, we can back propagate the loss through all the previous RNN units and embedding matrices and update its weights accordingly.
In theory, RNNs are absolutely capable of handling such long-term dependencies. But, in practice it's not usually true especially when sequence is very long [35]. During the gradient back-propagation phase, the gradient signal can end up being multiplied a large number of times (as many as the number of time steps) by the weight matrix associated with the connections between the neurons of the recurrent hidden layer. This means that, the magnitude of weights in the transition matrix can have a strong impact on the learning process. If the weights (eigenvalue) in this matrix are less than 1, it can lead to a situation called vanishing gradients where the gradient signal gets so small that learning either becomes very slow or stops working altogether. It can also make more difficult the task of learning long-term dependencies in the data. Conversely, if the weights (eigenvalue) in this

matrix are greater than 1, it can lead to a situation where the gradient signal is so large that it can cause learning to diverge. This is often referred to as exploding gradients. Long short term memory (LSTM) and Gated Recurrent Unit (GRU) are variants of recurrent neural network designed to deal with such problem [35].

Long Short Term Memory (LSTM)

In gradient problem that happen because of long dependency, Hochreiter & Schmidhuber [35] introduce Long-short term memory (LSTM). The LSTM uses self-connected unbounded internal memory cells that ensure a constant error flow. A memory cell is composed of four main elements: an input gate, a neuron with a self-recurrent connection (a connection to itself), a forget gate and an output gate. The self-recurrent connection has a weight of 1.0 and ensures that, barring any outside interference, the state of a memory cell can remain constant from one-time step to another. The gates serve to modulate the interactions between the memory cell itself and its environment. The input gate can allow incoming signal to alter the state of the memory cell or block it. On the other hand, the output gate can allow the state of the memory cell to have an effect on other neurons or prevent it. Finally, the forget gate can modulate the memory cell's self-recurrent connection, allowing the cell to remember or forget its previous state, as needed. This allows the model to capture information across a wide range of timescales. Since then LSTMs have been implemented effectively across many natural language processing tasks [35] all tasks that place importance on the sequence of events. Figure 2-5 depicts how LSTM RNN works to calculate hidden state weights.

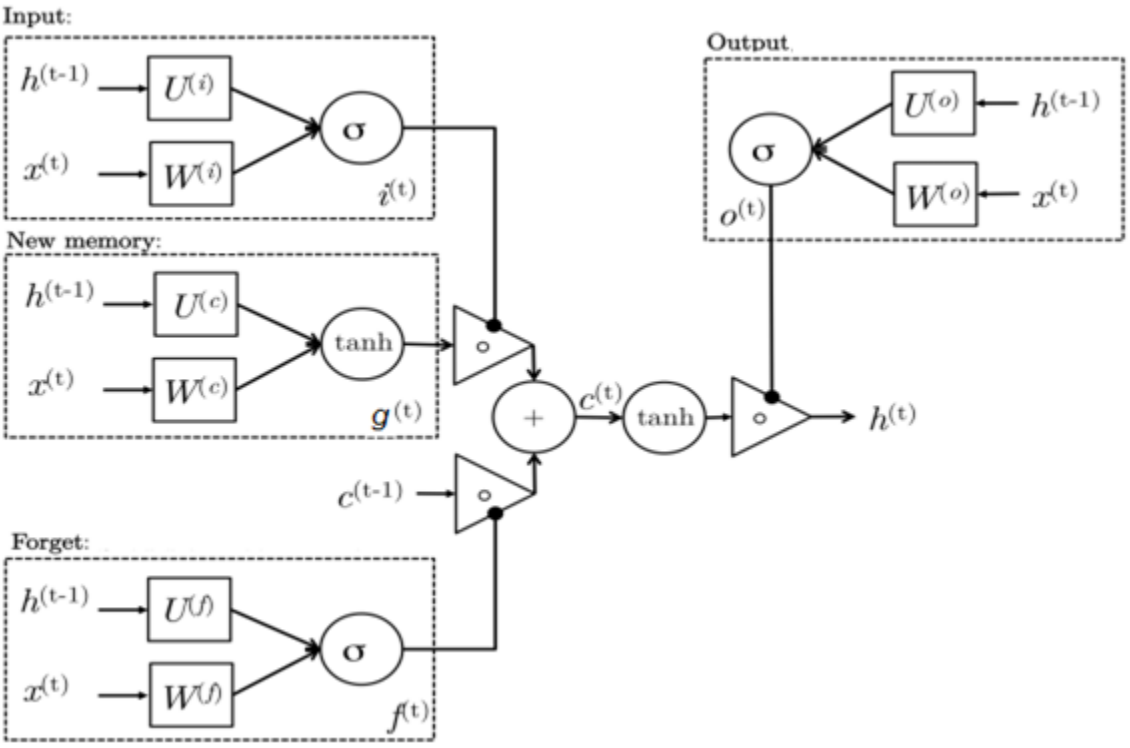


Figure 2-5*: How LSTM RNN [35] works*

In the above Figure 2-5:

- ✓ Gates *i, f,* and *o* are called the input, forget and output *gates*, respectively. They have the exact same equations as defined below, with different parameter weight matrices. They called *gates* because the sigmoid function (σ) squashes the values of these vectors between 0 and 1, and by multiplying them elementwise with another vector you define how much of that other vector you want to "let through". The input gate defines how much of the newly computed state for the current input you want to let through. The forget gate defines how much of the previous state you want to let through. Finally, the output gate defines how much of the internal state you want to expose to the external network (higher layers and the next time step). All the gates have the same dimensions$d^{(t)}$, the size of the hidden state.
- ✓ g is a new hidden state that is computed based on the $X^{(t)}$ current input and $h^{(t-1)}$ previous hidden state
- ✓ $C^{(t)}$ is called the internal memory of the unit is a combination of the previous memory $C^{(t-1)}$ multiplied by the forget gate *f*, and the newly computed hidden state g, multiplied by

the input gate. Thus, intuitively it is a combination of how we want to combine previous memory and the new input. We could choose to ignore the old memory completely (forget gate all 0's) or ignore the newly computed state completely (input gate all 0's), but most likely we want something in between these two extremes.

- ✓ Given the memory$C^{(t)}$, we finally compute the output hidden state $h^{(t)}$ by multiplying the memory with the output gate. Not all of the internal memory may be relevant to the hidden state used by other units in the network.
- ✓ $U\ and\ W$ are learnable weights

To formalize how LSTM hidden state $h^{(t)}$ is computed:

$$h^{(t)} = o^{(t)} \odot tanh(C^{(t)}) \tag{2.11}$$

$$C^{(t)} = f^{(t)} \odot C^{(t-1)} + i^{(t)} \odot g^{(t)} \tag{2.12}$$

$$g^{(t)} = tanh\left(W^{(c)}\left[x^{(t)}\right] + U^{(c)}h^{(t-1)} + b^{(c)}\right) \tag{2.13}$$

$$o^{(t)} = \sigma\left(W^{(o)}\left[x^{(t)}\right] + U^{(o)}h^{(t-1)} + b^{(o)}\right) \tag{2.14}$$

$$i^{(t)} = \sigma\left(W^{(i)}\left[x^{(t)}\right] + U^{(i)}h^{(t-1)} + b^{(i)}\right) \tag{2.15}$$

$$f_t = \sigma\left(W^{(f)}\left[x^{(t)}\right] + U^{(f)}h^{(t-1)} + b^{(f)}\right) \tag{2.16}$$

, where σ is sigmoid, $\odot$ is element-wise operation, non-linearity function, that decides which values will be updated. To update weights U and W it is same procedure as we did for Vanilla RNN above.

Gated Recurrent Unit (GRU)

A gated recurrent unit (GRU) was proposed by Cho et al., [37] in 2014 to make each recurrent unit to adaptively capture dependencies of different time scales. Similarly, to the LSTM unit, the GRU has gating units that modulate the flow of information inside the unit, however, without having a separate memory cells. Unlike LSTM, GRU has two gates, a reset gate *r*, and an update

gate z. Intuitively, the reset gate determines how to combine the new input with the previous memory, and the update gate defines how much of the previous memory to keep around. If we set the reset to all 1's and update gate to all 0's its function is same as Simple RNN model. The basic idea of using a gating mechanism to learn long-term dependencies is the same as in a LSTM, but there are a few key differences as depicted in Figure 2-6 below:

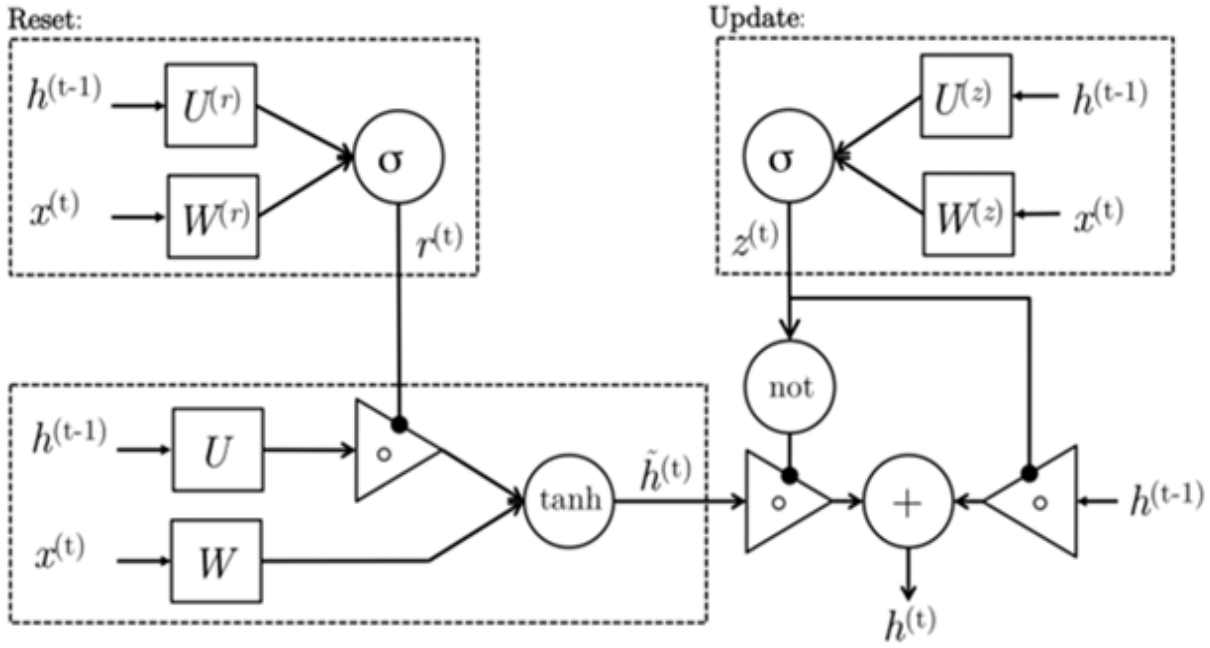


***Figure 2-6**: How GRU RNN [35] works*

To formalize Figure 2-6: the equation used to compute GRU hidden state $h^{(t)}$ is given by:

$$h^{(t)} = z^{(t)} \odot \tilde{h}^{(t)} + \left(1 - u^{(t)}\right) \odot h^{(t-1)} \tag{2.17}$$

$$\tilde{h}^{(t)} = tanh\left(W\left[x^{(t)}\right] + U\left(r^{(t)} \odot h^{(t-1)}\right) + b\right) \tag{2.18}$$

$$z^{(t)} = \sigma\left(W^{(z)}\left[x^{(t)}\right] + U^{(z)} x^{(t-1)} + b^{(z)}\right) \tag{2.19}$$

$$r^{(t)} = \sigma\left(W^{(r)}\left[x^{(t)}\right] + U^{(r)} x^{(t-1)} + b^{(r)}\right) \tag{2.20}$$

GRU has the following difference when compared to LSTM cells:

- ✓ GRU has two gates, an LSTM has three gates.
- ✓ GRUs don't possess and internal memory that is different from the exposed hidden state. They don't have the output gate that is present in LSTMs.
- ✓ The input and forget gates are coupled by an update gate z and the reset gate r is applied directly to the previous hidden state. Thus, the responsibility of the reset gate in a LSTM is split up into both r and z.
- ✓ We don't apply a second nonlinearity when computing the output.

According to empirical evaluations in RNN variants [36], there isn't a clear point of reference to select. In many tasks both architectures yield comparable performance and tuning hyper-parameters like layer size is probably more important than picking the ideal architecture. GRUs have fewer parameters (U and W are smaller) and thus may train a bit faster or need less data to generalize. On the other hand, if one has enough data, the greater expressive power of LSTMs may lead to better results.

Bidirectional-RNN

So far, we have focused on RNNs that look into the past words to predict the next word in the sequence. It is possible to make predictions based on future words by having the RNN model read through the corpus backwards. Dependencies in sentences don't just work in one direction; a word can have a dependency on another word before *or* after it. For natural language, we need to be able to effectively encode any input, regardless of dependency directions within that input, so this won't cut it. Bidirectional RNNs fix this problem by traversing a sequence in both directions and concatenating the resulting outputs (both cell outputs and final hidden states). For every RNN cell, we simply add another cell but feed inputs to it in the opposite direction; the output o^t corresponding to the *t*-th word is the concatenated vector $[o_t^{(f)} o_t^{(b)}]$, where $o_t^{(f)}$ is the output of the forward-direction RNN on word t and $o_t^{(b)}$ is the corresponding output from the reverse direction RNN. Similarly, the final hidden state is $h = [h^{(f)} h^{(b)}]$, where$h^{(f)}$ is the final hidden state of the forward RNN and $h^{(b)}$ is the final hidden state of the reverse RNN.

To sum up, one shortcoming of Vanilla RNNs is that they are only able to make use of previous context. In assessment task, the decision is made after the whole answer is processed and syntactic and semantic information behind provided answer should be summarized. Therefore, we need an encoder that analyze relevance of concepts in student answer with context of model answer by exploring both previous and future context in sequence representation. According to survey done on state of art deep learning networks recurrent neural network (RNN) and its variant has good performance on semantic feature learning [60], they declared evidence that both LSTM and GRU can work well with NLP task by computing a weighted combination of all words in the sentence for sentence level and weighted sum of all sentence for document level encoder. This variant of RNN, LSTM and GRU, can do well on long-dependency in sentence. With this inspiration, in this thesis we will explore the power of both bidirectional LSTM and or GRU to score essay by encoding essay level context and short answer at sentence level context.

Attention in Neural Network

Naturally word sequences are represented as meaningful information using last time-step of encoder. However, since not all vector in a vector sequence is relevant, it is hard to encode all the relevant input information needed in a fixed-length vector. This problem is addressed by introducing an attention mechanism at each level that estimates the importance of each time-step vector to the representation of the sentence or document meaning. The idea behind attention mechanisms is certainly motivated by observing the visual attention of humans. Despite processing the visual input all at the same time, humans rather pay attention to part of it sequentially one after the other. This allows to keep the amount of information to be manageable. Then we grasp only import representation of a text to infer meaning of what we are reading about. With an attention mechanism, we no longer try encode the full source text into a fixed-length vector. Importantly, we let the model learn what to attend based on the input sequence and what it has produced so far. As the result, attention mechanisms have become fundamental part of sequence modeling in various tasks. In subjective question assessment the goal of using attention is to derive a context vector that captures relevant answer information to help scoring module by clueing which answer words are more relevant. Several attention mechanisms are used in NLP. The most common way is applying similarity between provided or learned vector and attending vector then providing the

result to Softmax to get relevancy score distribution. Finally, getting maximum or average of attended vectors as relevant information.

Scoring

In subjective question assessment, this module works as score predictor by aiming to minimize cross entropy error or mean absolute error by treating the task as classification or regression respectively. It takes trained model and answers represented in the same format to training data, then predict score between specified ranges. Usually this layer is output layer in neural network using Softmax linear regression to predict score of provided answer.

2.5 Tools for Deep Learning

With advent of deep learning, several tools are designed to minimize programming load. The following are popular deep learning tools used for word representation and neural networks such as RNNs and CNNs [59].

TensorFlow

TensorFlow is open source python library for deep learning experimentation that is created by Google Brain team. It works on Linux, Mac OS X, Windows platform and it has C++, Python implementation. Its libraries are quite similar to Theano. It has pre-trained models for Recurrent Neutral Network (RNN) and Convolutional Neural Network (CNN).

Keras

Keras is an open source software for deep learning created by François Cholet. It is written by python and works on Linux as well as on window when there is Theano at back end. It has pre-trained models for Recurrent Neutral Network (RNN) and Convolutional Neural Network (CNN). It is deep learning library for Theano and TensorFlow that was developed with the intention of fast experimentation. It was developed with a focus on enabling fast experimentation and runs seamlessly on CPU and GPU. This make it preferable for research work.

Torch

Torch is also open source that work on Linux, MacOS, windows and Android. It is a computational framework with an API written in Lua that supports machine-learning algorithms. It is powerful

but, was not designed to be widely accessible to the Python-based community it has also pre-trained models for RNN and CNN.

Theano

It is a platform for deep learning library that allows to create the neural network models. Theano is a library that handles multidimensional arrays, like Numpy. Numerous open-source deep learning libraries have been built on top of Theano, including Keras.

Gensim

Gensim is free Python wrapper designed to process raw, unstructured texts to create word representation. It has efficient implementations for several popular word representation learning such as FastText, Word2Vec, and LSA.

Scikit-Learn

Scikit-Learn is simple and efficient tool for data mining and data analysis. It also automatically evaluates inter rater correlation between two rater values provided. Popular metrics included under scikit-learn are Pearson, Spearman, and Cohen's Kappa.

2.6 Performance Evaluation Measurements for SQM

The assumption in most of the SQM systems is that grades given by human assessors describe the true quality of an answer. Thus the aim of the systems is to simulate the grading process of human raters. Therefore, SQM systems is said to perform well if it's able to grade subjective question answers as accurately as human raters. According to [38] there are basically three critical elements of an assessment system: these are validity which deals with worth of measuring whereas the reliability question focused on the acceptable range of score consistency from one rating to another. Finally, the accountability question deals with how testing results are to be reported to the public. These issues should be considered when evaluating the performance of automated systems through various evaluation metrics [41].

Currently, there are a number of evaluation metrics available to measure the performance of SQM. However, common benchmarks and evaluation measures for this application do not currently exist. It is yet impossible to perform a comparative evaluation or progress tracking of this application across similar systems. Moreover, there is no common measure used to make

scoring results comparable. Scoring agreement has been reported in terms of exact or adjacent percentages, Pearson or Spearman's correlation, and kappa statistics [21]. Since correlation is most commonly used measurement in automated subjective text scoring, this thesis will use correlation of manual and system result, exact or adjacent agreement as a performance measurement which measures the percentage of agreement between system score and manual score.

Pearson Correlation or inter-rater reliability: It measures the standard correlation how much the actual scores (X) are related with the predicted scores (Y) [42] and calculated by applying the following Equation:

$$\text{Correlation}(X, Y) = \frac{Covariance(X, Y)}{SatndardDev(X) * StandardDev(Y)} \quad (2.21)$$

Spearman rank correlation is a non-parametric test that is used to measure the degree of association between the two ordinal variables reduced to ordinal scale. It uses ranks as opposed to actual values unlike that of Pearson correlation. Equation 2.22 is used to calculate the spearman rank correlation.

$$\rho = 1 - \frac{6x \sum d^2}{nx(n^2 - 1)} \quad (2.22)$$

Where 6 is a constant, *n* the number of paired ranks and *d* is the difference between the paired ranks

Root Mean Square Error (RMSE): is the standard deviation of the prediction errors. Prediction errors are a measure of how far from the regression line data points are; RMSE[3] is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Root mean square error is commonly used in and regression analysis to verify experimental results. The formula is:

$$RMSE = \sqrt{(p - a)^2} \quad (2.23)$$

, where *p* is expected values (predicted) and *a* is observed values (actual).

[3] http://www.statisticshowto.com/rmse/

Cohen's Kappa

Cohen's kappa[4] takes into account disagreement between the two raters, but not the degree of disagreement. It is a measure of the agreement between two raters who determine which category a finite number of subjects belong to whereby agreement due to chance is factored out. The two raters either agree in their rating (i.e., the category that a subject is assigned to) or they disagree; there are no degrees of disagreement (i.e., no weightings).

Score predictions are evaluated based on objective criteria, and specifically using the quadratic weighted kappa error metric, which measures the agreement between two raters.

Kappa does not take into account the degree of disagreement between observers and all disagreement is treated equally as total disagreement. Therefore, when the categories are ordered, it is preferable to use Weighted Kappa, and assign different weights $w_i$to subjects for whom the raters differ by *i* categories, so that different levels of agreement can contribute to the value of Kappa. This metric typically varies from 0 (only random agreement between raters) to 1 (complete agreement between raters). In the event that there is less agreement between the raters than expected by chance, this metric may go below 0. The quadratic weighted kappa is calculated between the automated scores for the responses and the resolved score for human raters on each set of responses. The mean of the quadratic weighted kappa is then taken across all sets of responses. For linear metric, if there are *k* categories, the weight *w* is calculated as follows:

$$w_i = 1 - \frac{i}{k-1} \tag{2.24}$$

And quadratic weighted kappa is calculated as:

$$w_i = 1 - \frac{i^2}{(k-1)^2} \tag{2.25}$$

[4] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.cohen_kappa_score.html

Summary

With this chapter reviewed educational assessment and its major classification based on its relevance to our work point of view. The thesis focused on subjective type of question and significant to measure the learning outcomes. Assessment as whole and subjective question assessment in particular and approaches to automatically assess subjective question answer is explained. Further literatures related to historical overview, benefits of automated subjective question scoring and approaches used to develop the system are reviewed. A number of approaches or techniques are available in order to develop the automatic subjective question marking, but among the most common approaches, thesis only deep learning approaches in detail and machine learning, ontology based and text similarity approach in highlight, considering their significance to our proposed method of development. Regarding performance measurement of the system, there are a number of metrics available which are used to measure the performance of the system. The research discussed some of the most common metric systems which are relevant to our work.

CHAPTER THREE

3. RELATED WORK

3.1 Subjective Question Assessment

Research in grading subjective questions has a history dating back to the early 1960's the development of Project Essay Grader (PEG) [19]. Since then, automatic grading of natural language responses has become a large field and several methodologies have been proposed to solve the problems in automatic evaluation of open questions. The key *focus* of the grading technique in subjective question grading systems tend to focus more on content and style [44]. Based on techniques used to understand implicit knowledge hidden in student answer through either or both content and style analysis, we have classified previous approaches used in short answer grading into five categories. The following sub sections discuss earlier works introduced by different authors to deal with subjective question assessment problem.

3.1.1 Statistical and Probabilistic Based Approach

In 2001 Lemaire et *al.,* [39] developed another essay scoring system, Intelligent Essay Assessor (IEA), analyzes and scores an essay using a semantic text-analysis method called *Latent Semantic Analysis* (LSA The underlying idea of LSA is that the meaning of a text is very much dependent on its words and changing even only one word can result in meaning differences in the passage. On the other hand, two texts with different words might have a very similar meaning [40]. IEA main focus is more on the content related features rather than the form related ones; however, this does not mean that IEA provides no feedback on formal in an essay. In other words, even though the system uses an LSA-based approach to evaluate mainly the quality of the content of an essay, it also includes scoring and provide feedback on spelling, grammar and redundancy. The system needs to be trained on a set of domain-representative texts in order to measure the overall quality of an essay. As stated by [18], IEA uses three sources to analyze an essay: (1) pre-scored essays of other students, (2) expert model essays and knowledge source materials, (3) internal comparison of an unscored set of essays. This approach allows IEA to compare each essay with similar texts in terms of the content quality. First, IEA compares content similarity between a student's essay and other essays on the same topic scored by human raters to determine how closely they match.

It then predicts the overall score by adding a "corpus-statistical writing-style" and. It also spots plagiarism and provides feedback. IEA requires only 100 domains representative pre-graded training essay to predict score for new essay, which is less than PEG training set requirement. Weakness of IEA is, it is limited to assess the content of an essay and fails to provide information regarding word order.

In 2009, Sukkarieh *et al.,* [21] developed Conceptual Rater (C-rater) that is a Natural language based prototype aimed at the evaluation of short answers related to content-based questions. There are four main steps in c-rater. The first one is Model Building, where a set of model answers are generated. Second, c-rater automatically processes model answers and students' answers using a set of natural language processing (NLP) tools and extracts the linguistic features. Third, the matching algorithm Gold map uses the linguistic features culminated from both first step and NLP to automatically determine whether a student's response entails the expected concepts. Finally, C-Rater applies the scoring rules to produce a score and feedback that justifies the score to the student. It used gold standard model patterns to score student answers according to their syntactical structure. These patterns are built semi-automatically by converting each answer into a set of one or more predicate-argument tuples. C-Rater reported having an accuracy of between 81% and 90% when used by The National Assessment of Education Progress agency. Modern work on C-Rater treats the grading task more similar to a textual entailment task. It analyzed 100-150 graded student answers to create a set of concepts for which each is represented by a set of sentences supplemented by a lexicon. Scoring is based on the presence or absence of these concepts. For more development of C-Rater, the student answers are parsed, to extract a predicate argument structure that is then categorized as absent, present, or negated for each concept, using a maximum entropy-based matching algorithm. The reported agreement (per concept-math) was 84.8% compared to an annotator agreement of 90.3%. Primary drawback of this approach is dependency on linguistic feature in addition to annotated matching corpora used create concepts.

3.1.2 Ontology Based Approach

In 2012, Fernando *et al.,* [49] proposed Ontology-based Information Extraction (OBIE) for short answer grading that support both marking and feedback. OBIE has mainly ontology, preprocessing and rule extractor modules to deal with marking problem. Manually created ecosystem domain ontology is used as knowledge source to extract concepts. The preprocessing stage considers

completing sentences, eliminating non-informative words, and correcting misspellings. The third module deal with extracting information from text. Based on extraction rule information extraction technique that use regular expression, authors categorized student summary into three as correct statements, incorrect statements, and incomplete statements. To identify correct, incorrect, and incomplete statements from student answer, authors constructed first order logic rules manually depending on level of importance, presence or absence of concepts on constructed ontology. Grading metrics used in this approach are existence of main concepts or ideas presented in the student answer, length of the text, and amount of relevant information.

One strength of this approach is it can generate constructive and individualistic feedback for students. However, generating rules for unstructured text is usually vulnerable to error prediction. Moreover, metrics used in grading are content based and style analysis is not considered. Because of length of text is included as metric the system is susceptible for cheating. Also assessment is dependent on domain knowledge that is not feasible to create quality knowledge for all domain. It needs domain knowledge with all concept coverage for all domain. Any unknown concept is treated as out-of-vocabulary. With this condition it is not feasible solution as compared to recent transfer learning approaches that follows train in one domain score other domain.

In 2015, V Senthil and A Sankar [26], proposed Ontology mapping for assessing short answer subjective questions. The system has four main modules. The first module is Sentence Extractor which read text (both model answer and student answer) and return sentence. The second module is part of NLP linguistic preprocessing feature that take sentence and apply NLP parse using Stanford Dependency Parser to build typed dependency representation of the sentence. The third module take dependency relation of each sentence and construct Ontology that is used for mapping in next step. The fourth module is Ontology mapping that perform similarity between two Ontology concepts (i.e., model answer ontology and student answer ontology). This module returns mark for student answer based on the weightage and similarity score. Strength of this approach is it overcome the problems related to syntax variation (order of sentence elements) and semantic understanding. As indicated by experimentation part of the article, using NLP based preprocessing with Ontology outperformed other conventional approaches. It has above 0.79 (79 %) Pearson correlation with human grader. However, according to [50], mapping two ontology concepts has uncertainty issue. Concept A exist in domain *x* is not usually same with concept A in

domain *y*. Moreover, learning ontology needs sophisticated NLP tools and model is not transferable.

3.1.3 Text Similarity Based Approach

In 2009, Michael M. and Rada M. [51] proposed unsupervised techniques for the task of automatic short answer grading by considering the problem of marking as text similarity. Experimentation of this paper focus on identifying semantic similarity measure suitable for short answer grading and determining extent to which domain and size of data used to train corpus based similarity approaches that influence accuracy of grading. To achieve the goal, several corpora based and knowledge based similarity measures are experimented. In addition, set of experiments which vary the size and domain of the corpus used to train corpus based semantic similarity measure metrics are done to show effect on accuracy of short answer grading. Latent Semantic Analysis (LSA) and the Explicit Semantic Analysis (ESA) are two corpus-based measures selected on this paper. All the word-to-word similarity scores obtained in this way are summed up and normalized with the length of the two input texts (model answer and student answer). The results indicate that when used in their original form, the results obtained with the best knowledge-based i.e., WordNet shortest path and corpus-based measures i.e., LSA and ESA have comparable performance. Finally, authors introduced a technique for integrating feedback from the student answers using a method similar to the pseudo-relevance feedback technique used in information retrieval to grading system and improved accuracy.

Strength of this approach is authors compared effect of several knowledge based and corpus based semantic similarity approaches in different corpus size and domain and introduced integrating relevance feedback from student answer to grading system. However, like other supervised approaches, only content is analyzed in marking process. Moreover, knowledge source used (WordNet) is not suitable to detect domain implicit knowledge and taxonomic databases like WordNet may not include some domain dependent terms.

In 2012, Hassan and Aly [23] presented string similarity and corpus based similarity technique for short-answer scoring. The presented system aims to measure the similarity between the student's answer and the model answer to produce the final score for the student response. Thirteen string-based similarity algorithms seven character-based distance measures and six term-based distance

measures are used. In addition to string similarity, corpus based semantic similarity algorithm called Distributional Similarity Co-occurrences (DISCO) that computes distributional similarity between words by using a simple context window of size ±3 words for counting co-occurrences is used. When two words are subjected for exact similarity DISCO simply retrieves their word vectors from the indexed data, and computes the similarity according to Lin measure [45].

To grade score of student answer, the system passes through three stages. The First stage is measuring the similarity between model answer and student answer using String-Based algorithms using. Secondly, measuring the similarity using DICSO corpus-based similarity is performed. In this stage removing the stop words, getting distinct words and constructing the similarity matrix is performed. The similarity matrix represents the similarity between each distinct word in the model answer and each distinct word in the student's answer. Each row represents one word in the model answer, and each column represents one word in the student's answer. The last two columns represent the maximum and the average similarity of each word in the model answer. Finally, overall similarity is determined by computing the average of the last two columns (Max, Average). This final overall similarity is taken as student mark. Strength of this approach is it cannot require any linguistically annotated corpus for training and requires only low level linguistic preprocessing such as tokenization, stop word removal and stemming. However, according to experimentation it achieved maximum correlation value of 0.504 which is comparatively less than other supervised approaches discussed above. Moreover, style grading is doesn't taken into account.

3.1.4 Supervised Machine Learning Based Approach

In 1998, Burstein et al., [2], developed and later enhanced in 2006. E-rater employs a corpus-based approach to model building, in which actual essay data are used to examine sample essays. The features of e-rater include a syntactic module, a discourse module, and a topical-analysis module. These modules provide outputs for model building and scoring. E-rater has been trained on a set of essays scored by at least two human raters on a 6-point holistic scale to build models. The origin of the syntactic module is parsing. The discourse module uses a conceptual framework of conjunctive relations including cue words (e.g., using words like "perhaps" or "possibly" to express a belief), terms (e.g., using conjuncts such as "in summary" and "in conclusion" for

summarizing), and syntactic structures to identify discourse-based relationship and organization in essays. Finally, the topical analysis module identifies vocabulary usage and topical content.

To summarize, e-rater uses NLP linguistic feature extraction techniques to identify the features of scored essays in its sample collection and store them-with their associated weights-in a database. E-rater can evaluate both style and content of essay. When e-rater evaluates a new essay, it compares its features to those in the database in order to assign a score. Because e-rater is not doing any actual reading, the validity of its scoring depends on the scoring of the sample essays from which e-rater's database is created. E-rater needs 465 expert scored essays as training set. It is successfully used in GMAT with agreement rates between human expert and system consistently between 84%. However, it is not suitable for technical answers and is like an extension of PEG [19].

In 2016 M. Syamala [58], compared four machine learning techniques (Latent Semantic Analysis (LSA), Generalized Latent Semantic Analysis (GLSA), Maximum Entropy (MaxEnt) and BiLingual Evaluation Understudy (BLEU)) in both with and without Ontology approach for subjective English answer evaluation. Author justified that use of Ontology looks not just for keywords but the keywords appearing in right context and thus models human mind more accurately as human evaluation is by and large influenced by answer length, keyword presence and context of keywords. From analysis done Ontology with Maximum Entropy (MaxEnt) shows that high correlation (up to 90 percent) with Human Performance. From training data word context is detected by analyzing word that follow and precede the given word. The entropy is calculated for the current word to appear in a given context. Using word context and similarity between each concept in Ontology calculated using path length between each concept in knowledge base ontology and given as weight for concepts appear and concepts not included in model answer are used to enhance unseen model answer. Finally, mapped concepts are passed to MaxEnt for context analysis and score is predicted based on output of MaxEnt classifier. This is state-of-art result achieved for short answers, but their dataset is not released. The drawback of this technique is it relies on domain knowledge with all concepts. If concept is not avail in knowledge base it is treated as out-of-vocabulary. Moreover, it requires external vocabularies such as WordNet for synonymy search. Gives more credit for concept presence. Can be vulnerable for cheating if student

repeatedly use keywords in answer. Moreover, because of Ontology concept relation, it may include not related concepts to reference answer.

In 2016, Shourya et al., [46] proposed iterative technique on an ensemble of text classifier of student answers and classifier using numeric features derived from various similarity measures with respect to model answers. The aim of this paper is to overcome couple of problems in previous supervised approach for short answer grading. The article criticized previous approaches for their great reliance on instructor provided model answers and need for labeled training data in the form of graded student answers for every assessment task. According to the author, variedness of nature of model answers across questions and difference on student answers and corresponding model answers matters the score. To address the above shortcomings, authors introduced automatic short answer grading as a supervised learning task where they employ an ensemble of two classifiers to predict student scores. In the ensemble, the first classifier is a text classifier trained using the classical TF-IDF representation of bag of word (BoW) model of student answers. It is independent of model answers and learns textual features (words and n-grams) from graded student answers to discriminate between student answers belonging to different scores. The second classifier has features expressed as real numbers indicating similarity of student answers with the corresponding model answer (analogous to model answer based classifiers). This reduce continuous labeling effort needed for the task.

Authors employ five generic short-text similarity measures to compute similarity between the model and student answers covering lexical, semantic and vector-space measures. Evaluating Responses with BLEU (a lexical measure comparing student answers against model answers using a modified version of the n-gram co-occurrence scoring algorithm), WordNet based similarities, Latent Semantic Analysis (LSA) trained on a Wikipedia dump and Word2Vec trained on 100 billion words of Google news dataset are five similarity measures used to compute similarity between student answer and model answer. Word-to-word similarity measures obtained using Euclidean distance between word vectors are used. Additionally, the model of the first classifier is question specific (i.e., a word which is a good feature for a question is not necessarily a good feature for another question), whereas features for the second classifier are more question agnostic (i.e., high similarity with respective model answer is indicative of high scores irrespective of question). The two classifiers thus capture complementary information useful for grading student

answers. It is done in two steps - (i) obtaining the second classifier through a feature based transfer of the model from the source to the target question, followed by (ii) iteratively building the first classifier and the ensemble using pseudo labeled data from the target question. Finally, these two classifiers are combined in a weighted manner to form an ensemble which is used to predict the final score. The authors experimented their approach with dataset released by for the joint task of student response analysis in SemEval 2013 Task 7 and achieved promising result.

Strength of this approach is its transferable feature. Assessing without model answer minimize load on instructor. In addition, features used are not domain dependent and unsupervised. It is good because most of data is unlabeled. External knowledge used are learned from unlabeled data except WordNet. However, this approach treated assessment as presence of related keywords. Assessment is beyond looking for presence of concepts; we should care about context on which the concept exists. Simple word order change can change meaning. Moreover, if trained on domain dependent dataset Word2Vec can represent words in more specific to the task and WordNet taxonomy may not have concepts of domain and predictive models such as Word2Vec [26] and FastText [32] and count based word representation GloVe [29] can do better.

3.1.5 Deep Learning Based Approach

In 2016, Dimitrios et al., [47] introduced a model that forms word representations by learning the extent to which specific words contribute to the text's score using special kind of recurrent neural network (RNN), capable of learning long-term dependencies, called Long-Short Term Memory (LSTM) [35] networks to represent the meaning of texts. The aim here is to construct representations which, along with the linguistic information given by the linear order of the words in each sentence, are able to capture usage information and called score-specific word embeddings. With this approach, having no prior knowledge of syntactic structure of the language or the domain of the text, authors demonstrated SSWE outperform existing state-of-art word embedding's. Furthermore, no any further pre-processing of the text other than simple tokenization is done. This solve problem raised in earlier approaches that deal more linguistic preprocessing such as POS tagging and parser. Instead of simple LSTM [35] that encode text in forward direction, bi-directional LSTMs is utilized i.e., two independent RNN encode the essay (from left to right and from right to left) and the result of two LSTM layers is concatenated together and passed to next layer. Finally, they passed encoded essay vectors to a linear unit in the output layer which predicts

the essay score. Authors experimented LSTM, BLSTM, Two-layer LSTM, Two-Layer BLSTM with SSWEs and word2vec models in addition to baseline SVM and doc2vec model. SSWE + Two-layer BLSTM model that trained on domain (essay) achieved state-of-art result by improving correlation of Spearman (ρ) to 0.91 and Pearson (r) to 0.96.

Usually when students write answer, possibility for spelling error is high. Word with spell error are not occur in globally released word vectors Word2vec [31]. Even training from domain essay as SSWE, with Word2Vec do not detect spelling errors. Moreover, with such embeddings rare words are poorly estimated, leading to high perplexities for rare words (and words surrounding them). This is especially problematic in morphologically rich languages with long-tailed frequency distributions or domains with dynamic vocabularies. Additionally, out-of-vocabulary (OOV) words are left zero embedding while are relevant to infer text wise meaning. Hence, sub-word information can play an important role in improving the representations for infrequent words and even OOV words [48]. Using FastText embedding or character level language modeling we can fix such problem. Moreover, not all encoded essay terms are equally relevant to scoring an essay. With recent attention approaches we can get most informative words from an essay. Moreover, essay text is hierarchically structured and usually need coherency.

3.2 Amharic Subjective Question Assessment

Current research in Amharic Natural Language Processing (NLP) covers different aspects of the language such as morphological Analysis, syntax and speech recognition, Part of Speech Tagger, Parses, Word Sense Disambiguation etc. This is very promising, but these researches mainly focused on lower level of NLP applications. Though morphological analysis is often considered as the first phase of a more complex NLP application, a significant research needs to be done in other areas in educational domain like computer based assessment.

Automatic subjective question assessment system is being extensively researched in English and other languages and has shown good as discussed in previous section. But there is only one attempt done by Abel in 2010 [15] in Amharic despite the aforementioned benefit which mainly focus on content of the text for Amharic factual essay. The author used Latent Semantic Analysis (LSA) method to evaluate and score Amharic factual essay. LSA fist processes a corpus of machine-readable language and then represents the words that are included in a sentence, paragraph, or

essay through statistical computations. LSA measures of similarity are considered highly correlated with human meaning similarities among words and texts. Moreover, it successfully imitates human word selection and category judgments. It uses a 'bag-of-words' approach in which similarity and co-location of words is evaluated. It is a corpus-based text comparison approach and uses an algebraic technique to determine the level of similarity between the text and the corpus. Two texts that use similar words would be considered semantically similar using LSA. The underlying idea is that the meaning of a passage is very much dependent on its words and changing even only one word can result in meaning differences in the passage. On the other hand, two passages with different words might have a very similar meaning.

When LSA is used to compute sentence similarity, a vector for each sentence is formed in the reduced dimension space, similarity is then measured by computing the similarity of these two vectors [10]. Because of the computational limit of SVD, the dimension size of the word by context matrix is limited to the several hundred. As the input sentences may be from an unconstrained domain (and thus not represented in the contexts) some important words from the input sentences may not be included in the LSA dimension space. Secondly, the dimension is fixed and so the vector is fixed and is thus likely to be a very sparse representation of a short text such as a sentence. Like other statistical methods, LSA ignores any syntactic information from the two sentences being compared and is understood to be more appropriate for larger texts than the sentences dealt with in this work. Therefore, with LSA the sentences "ፈጣኑ ዉሻ ደካማዉን ቀበሮ ዘሎ አለፈ (The quick dog jumped over the lazy fox)" and "ፈጣኑ ቀበሮ ደካማዉን ዉሻ ዘሎ አለፈ (The quick fox jumped over the lazy dog)" would be considered semantically similar while they are very different. Moreover, LSA has no ability to check technical correctness of the sentence. Beyond methodology used, Automatic Amharic Essay Scoring system proposed by Abel [15] is limited to assess content of an essay.

CHAPTER FOUR

4. DESIGN OF AUTOMATIC SUBJECTIVE QUESTION MARKING (SQM)

4.1 Overview

The literature review has revealed that the majority of the work done in automatic subjective question evaluation relies on hand crafted feature based approaches or restrictive external vocabularies such as Ontology**.** Handcrafting features is time-consuming**.** Moreover, extracted features are often over-specified and incomplete. In other way feature extracted for one domain is not fit to other domain or task. With recent advances in Artificial Intelligence, computers can do representations for learning and reasoning same way as human can do by learning context of words, characters or sentences in the text. In this thesis, motivated by the recent breakthroughs in NLP with deep learning, we proposed to design attention based neural network for subjective question marking. This chapter sets out to provide an overview of the proposed approach used to develop SQM system. The chapter begins with explaining over all architecture of the proposed model. The technical aspects regarding each part of the proposed model is detailed and as part of this investigation along with design decision justifications is discussed. Finally, summary of the chapter is included.

4.2 SQM Architectural Model

Though there are basic components such input module, preprocessing module, matching module, and scoring module that every automatic subjective question assessment system comprises of, the internal structures and algorithms of every SQM system differs from system to system depending on approach used. Hence, we will briefly describe the main components of SQM explored in this thesis work in details. In this study, we have identified seven fundamental components: preprocessing, building word vectors, sequence generator, word representation, context encoding, attention, modeling and scoring module as shown in Figure 4.1.

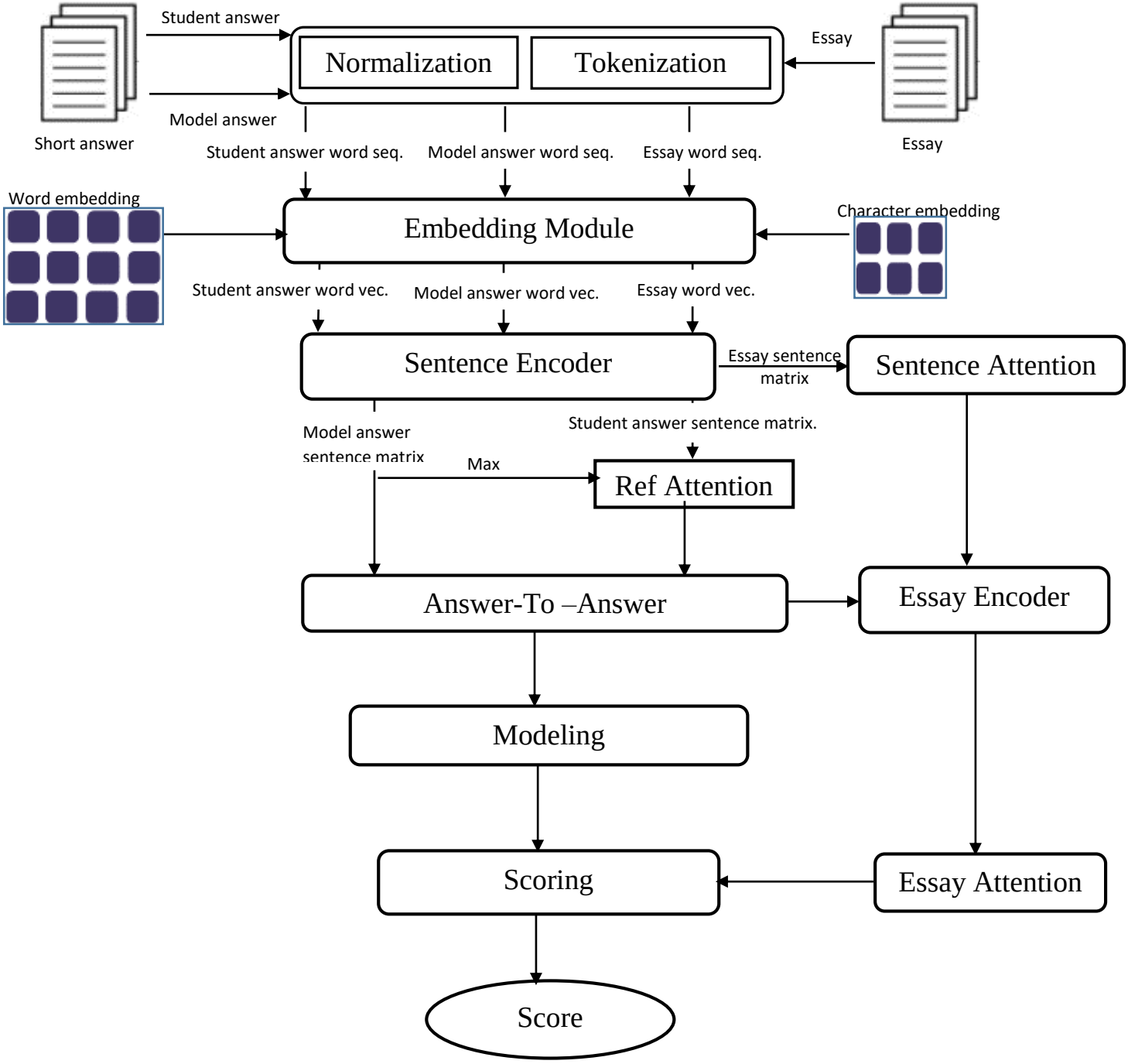


***Figure 4-1**: General Architectural Model of SQM*

4.3 Preprocessing Module

Preparing quality data is the primary step in a machine learning task. Preprocessing module of SQM is responsible to make the input data collected from different source to a format applicable to each modules. The primary use this module is to standardize data collected to train our FastText

embedding as unstructured Amharic data is collected from different source such as Amharic Wikipedia, Amharic news, course modules, fictions, spiritual files, examinations answer sheets, and etc. which exhibit heterogeneity in writing style (i.e., use of words as well as character language property). In addition, we incorporate preprocessing to normalize mismatch exist in model and student answer because of heterogeneity in writing style (e.g., ፅይል in one answer can be expresed as ሀይል in another answer). Unless input text is normalized to one standard style, assessment will severely be affected. Therefore, to minimize error prediction, we need to deal with language variations. Here under sub modules of preprocessing are described in detail. The proposed algorithm for preprocessing text is depicted in Algorithm 4.1.

i. Tokenization

Given a character sequence and defined delimiters, tokenization is the task of chopping a text into pieces usually characters, words and or sentences. This module applies character, word, and sentence level splitter over original text. Character splitter is used for character level answer modeling task, whereas word and sentence level tokenization's are used for word sequence generation module and word vector building module respectively.

Character level splitter treats each alphabet as unique token and outputs vocabulary of character to their index. All characters except whitespace are recognized under character vocabulary. To make splitting task easy, a whitespace character is append before any Amharic pucutaion mark[5]. Any pucutaion mark detected is treated as word, then our encoder can consider it as single time step and learn context of punctuation. In Amharic, the individual words in a sentence are separated by two dots (: ሁለትነጥብ). The end of a sentence is marked by Amharic full stop (። አራት ነጥብ). The symbol (፣ ነጠላ ሰረዝ) represents a comma, while (፤ ድርብ ሰረዝ) correspond to a semicolon. '!' and '?' punctuations are used to end exclamatory and interogative sentence respectively.

ii. Normalization

Normalization is the process of canonizing tokens to a standard format by avoiding differences in the character sequences of the tokens. In this work, three level tasks are identified and addressed.

[5] List of puncutation marks, short hand form words, and interchangeably written characters in Amharic language are collected from Jimma University Amharic Literature and Language Department and documented in Annex A and B.

The first task is character and word level normalization. Character level mismatch. Amharic has different characters that are interchangeably used in writing and reading such as (ሀ, ኀ, ሐ, and ኸ), (ሰ and ሠ), (ጸ and ፀ), (ው and ዉ) and (አ and ዓ). Amharic words with suffix such as ቷል are also written as ቱዋል. We normalize any character under such category to common canonical representation. The second variation in Amharic language that need normalization in Amharic text is short form[5] expression. For example, ትምህርት ቤት can also be represented as ት/ቤት in Amharic text. To deal with such difference, the list of short forms in Amharic language are consulted (c.f. Annex C) to expand a short form expression to its long form. The third task is data standardization. The dataset used for word embedding module is collected from different source and it has many non geez characters. To make our data in regular format, we preserve only geez characters. Also, we omitted any numbers from dataset collected from multiple source.

Algorithm 4-1*: Proposed Algorithm for tokenizing and normalizing Amharic Text*

	Amharic Text Tokenizer and Normalizer Algorithm
1.	**Input**:
2.	INPUT_FILE_DIR: STRING //name of directory for documents to be normalized
3.	IS_WORD_LEVEL: BOOLEAN // If True tokenizer split text into word level otherwise character Level
4.	ABREVATIONS: DICTIONARY //all identified short forms in Amharic to their expanded form (e.g., ት/ሚ to ትምህርት ሚኒስቴር)
5.	REPLACEABLE _CHARS: DICTIONARY //dictionary of characters with same sound and used interchangeably
6.	**VARIABLE:**
7.	**OUTPUT**:
8.	WORD_PER_SENTENCE: Nested List //returns words in given input as list of tokenized and normalized words
9.	CHARACTER_PER_SENTENCE: Nested List //returns list of Non-space geez characters in a text including Amharic punctuations (?!፡።፤;፦፥፧※፨)
10.	**BEGIN:**

	Amharic Text Tokenizer and Normalizer Algorithm
11.	READ Content as TEXT IN INPUT_FILE_DIR // read content of file
12.	SENTENCES=TOKENIZE (TEXT, [?።!]) // Tokenize to sentence level using delimiters
13.	FOR EACH sentence IN SENTENCES
14.	FOR EACH common_char, char_to_replace in REPLACEABLE _CHARS
15.	IF any existence of char from REPLACEABLE _CHARS IN sentence THEN
16.	REPLACE common_char // For example if any char or sequence match from [ሃኅኃሐሓኻ]' group replace with 'ሀ'
17.	ELSE IF any NUMBER or Non Amharic Character or Punctuation THEN
18.	REPLACE by WHITE SPACE
19.	ELSE // is considered as Amharic punctuation
20.	CONCATENATE with WHITE SPACE // Concatenate white space with character and replace character
21.	END IF
22.	IF IS_WORD_LEVEL TRUE THEN
23.	WORD_SEQUENCE=TOKENIZE (sentence, SPACE) // Tokenize by whitespace Character
24.	FOR EACH word IN WORD_SEQUENCE
25.	IF word IN ABREVATIONS THEN // get value based on key from abbreviations dictionary
26.	REPLACE word by expanded form ABREVATIONS
27.	APPEND to WORD_PER_SENTENCE // append word to inner list
28.	NEXT
29.	ELSE
30.	FOR EACH non_space_char IN sentence
31.	APPEND to CHAR_PER_SENTENCE // append to inner list that hold character per Sentence
32.	NEXT

	Amharic Text Tokenizer and Normalizer Algorithm
33.	END IF
34.	RETURN CHAR_PER_SENTENCE and WORD_PER_SENTENCE
35.	**END**

4.4 Word Vector Building Module

The result of pre-processing unstructured Amharic text is used as input to this component. The result of preprocessing is list of small sized files contains sentence per line for efficient use when training model. Then from preprocessed document, we first create vocabulary used as input for both input and output layer as one-hot vector. Then using created vocabulary and list of sentence word level chunked, we train both CBOW and Skip-gram neural model[6]. The architecture of Amharic FastText model is depicted in Figure 4.2.

FastText use a simple neural network with a single hidden layer to learn the weights of the hidden layer are actually the "word vectors". First step is building a vocabulary of words from our training data (output of preprocessing module). Vocabulary builder module generate dictionary of unique words to their sub-words (i.e., character n-grams). Our vocabulary breaks down each word to different character n-grams. As recommended by author [32], practical approach is chunking to tri-grams and hexa-gram. For example our vocabulary for word 'በፅሁፍ' contains (<በፅ, <በፅሁ, <በፅሁፍ, <በፅሁፍ>), (በፅሁ,በፅሁፍ,በፅሁፍ>),(ፅሁፍ,ፅሁፍ>),(ሁፍ>). Special characters '<' and '>' are appended to show start and end of word respectively. So the resulting word vector for the word will be the collection of the n-grams along with the word.

Then context builder module builds training samples based on given sliding window size (number of words taken as context at a time). If window size is 2 that means the network is training on 2 words-to-the-left of the target and 2 words-to-the-right of the target. Based on this window, our CBOW model define 'context' as the window of words to the left and to the right of a target word and tries to predict probability of target word based on context words and Skip-gram predicts

[6] https://radimrehurek.com/gensim/models/wrappers/fasttext.html

probability of context words being appear nearby target word. The following example shows how our context builder module works for both CBOW and Skip-gram FastText models:

Given the sentence “በፅሁፍ የሚቀርብ ዘገባ አቅራቢው በአካል ተገኝቶ መልዕክቱን እንዲያስተላልፍ አያስገድድም” with window size 2, our context builder generates:

በፅሁፍ	የሚቀርብ	ዘገባ

አቅራቢው በአካል ተገኝቶ መልዕክቱን እንዲያስተላልፍ አያስገድድም

- Skip-gram training samples: (በፅሁፍ, የሚቀርብ), (በፅሁፍ, ዘገባ)
- CBOW training samples: ([የሚቀርብ, ዘገባ], በፅሁፍ)

በፅሁፍ	የሚቀርብ	ዘገባ	አቅራቢው

በአካል ተገኝቶ መልዕክቱን እንዲያስተላልፍ

- Skip-gram training samples: (የሚቀርብ, በፅሁፍ), (የሚቀርብ, ዘገባ), (የሚቀርብ, አቅራቢው)
- CBOW training samples: ([በፅሁፍ], የሚቀርብ), ([ዘገባ, አቅራቢው], የሚቀርብ)

በፅሁፍ	የሚቀርብ	ዘገባ	አቅራቢው	በአካል

ተገኝቶ መልዕክቱን እንዲያስተላልፍ አያስገድድም

- Skip-gram training samples: (ዘገባ, በፅሁፍ), (ዘገባ, የሚቀርብ), (ዘገባ, አቅራቢው), (ዘገባ, በአካል)
- CBOW training samples: ([በፅሁፍ, የሚቀርብ], ዘገባ), ([አቅራቢው,በአካል], ዘገባ)

በፅሁፍ

የሚቀርብ	ዘገባ	አቅራቢው	በአካል	ተገኝቶ

መልዕክቱን እንዲያስተላልፍ አያስገድድም

- Skip-gram training samples: (አቅራቢው, የሚቀርብ), (አቅራቢው, ዘገባ), (አቅራቢው, በአካል), (አቅራቢው, ተገኝቶ)
- CBOW training samples: ([የሚቀርብ, ዘገባ], አቅራቢው), ([በአካል, ተገኝቶ], አቅራቢው)

.

.

.

በፅሁፍ የሚቀርብ ዘገባ አቅራቢው በአካል

መልዕክቱን	እንዲያስተላልፍ	አያስገድድም

- Skip-gram training samples: (አያስገድድም, መልዕክቱን), (አያስገድድም, እንዲያስተላልፍ)
- CBOW training samples: ([መልዕክቱን, እንዲያስተላልፍ], አያስገድድም)

Word in shaded column is target word selected at a time and words under white column are nearby words (context) in sliding window of size 2. Each words are constituent of character n-grams, so it constructs the vector for a word from character n-gram vectors that constitute a word and the training processed on each n-grams in contexts including word itself as n-gram. Order of word in a sentence is not preserved, but order of n-grams in each word is preserved. For example, the vector for the word "ከመረጃ" is not the same as the vector for the word "ከደረጃ", because the n-grams constituting both these vectors are very different. But vector for "በመረጃ" is more similar to "ከመረጃ" as they share multiple n-grams in addition to sharing same context as a word. This allows us to cluster word with same meaning, but has different syntax because of extended morphemes without using external tools such as stemmers or morphological analyzers. When training through whole dataset, our network cluster not only words semantically or syntatically related, but also words with spell error based on shared character n-grams. Based on extracted training samples the network is going to learn the statistics from the number of times each pairing shows up.

When *training* network on word pairs, instead of feeding words, we represent each vocabulary words as one-hot vector representing the input word (target) by placing 1 in the position corresponding to the target word, and 0s in all of the other positions.

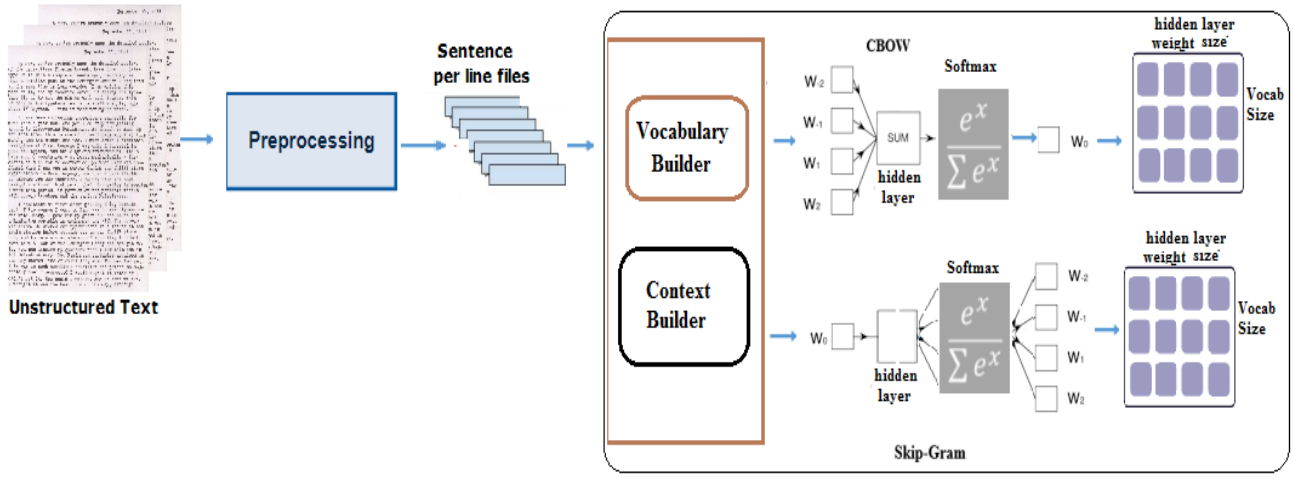


***Figure 4-2**: Amharic FastText Word Vector Generator Model Adapted from Joulin et al., [32]*

At input layer, for each alphabetically sorted unique vocabulary terms as target word, we create one hot vector of size *C*. i.e., for a given context word, only one out of *V* units, $\{x_1 \cdots x_v\}$, will be 1, and all other units are 0. Hidden layer of the network is based on this one hot encoded vector and represented in DxV matrix where D is column (i.e., number of neurons (a.k.a. features) one

for every neuron) and *V* is size of vocabulary. This *D* dimensional feature is finally printed as word vector for the word where size of *D* is defined at the time of training. The matrix is initially randomized and later updated by stochastic gradient learning. When computing the hidden layer output *h*, the CBOW model takes the average of the vectors of the input context words, and use the product of the input layer to hidden layer weight matrix *W* and the average vector as the output as shown Equation 4.1 and 4.2 below.

$$h = \frac{1}{C} W \tag{4.1}$$

$$= \frac{1}{C} (v_{w1} + v_{w2} + \cdots + v_{wc}) \tag{4.2}$$

, where *C* is the number of words in context, $w_1, \cdots, w_c$ are the words in context and v_w is the input vector of word *w* averaged and *W* is learnable input layer to hidden layer weight matrix.

While, the input vector of Skip-gram is the only word on the input layer, and thus we have the same definition of the hidden outputs as in CBOW, which means output *h* is simply copying (and transposing) a row of the input to hidden weight matrix, associated with the input word. Since the input vector is one-hot encoded, the weights coming from the nonzero element will be the only ones contributing to the hidden layer as indicated in Equation 4.3. Therefore, for the input *x* with $x_k = 1$ and $x_{k'} = 0$ for all $k' \neq k$ the outputs of the hidden layer *h* will be equivalent to the k^{th} row of input layer to hidden layer weight matrix *W*.

$$h = W_k := v_w \tag{4.3}$$

, where W_k is k^{th} row of weight in which one hot vector position is 1 (target word) and $v_w$is transposed word vector.

The output vector of hidden layer is fed to output layer. At this layer we use Softmax log-linear [52] classification model to calculate the probability distribution of the target word given a specific context for CBOW and probability distribution of contexts given target word for Skip-gram.

Specifically, each output neuron has a weight vector which it multiplies against the word vector from the hidden layer, then it applies the Softmax to the result. Finally, in order to get the outputs to sum up to 1, we divide this result by the sum of the results from all vocabulary size output nodes. Both input to hidden and hidden to output weight matrix is learned by stochastic gradient update. Final output is hidden layer weight matrix (float value between -1 and 1) with shape *V x D*, where *V* is vocabulary size and *D* is hidden layer neuron size (feature dimension) that show how each vocabulary word dimension is distributed in vector space. In that situation words with similar meaning fall to most likely similar space.

4.5 Embedding Module

Word embedding module of SQM has three sub components that allows us to represent meaning of words in answer. In the following sub section, we discuss each component in detail.

i. Sequence Generator Module

This module generates sequence of strings into sequence of integers (a.k.a. indices). Sequence shows how words are ordered in a sentence and or how characters are ordered in a word. It takes two input. One is preprocessed training data and the other is embedding matrix generated in word vector building module. Then we create character and word vocabulary that contain unique character to index and unique word to index respectively from input sequences. Using created word and character vocabulary, we generate sequence of character indices and word indices. Since we have two different task (essay and short answer as subjective answer), way of sequence generated depends on task approach. For essay we proposed hierarchical encoding of text i.e., we first encode sequence of words in a sentence then we encode how sentences are organized in essay paragraphs. So the output of sequence generator for essay should be three dimensional. The first dimension is essay size (indicates total number of essay used for training). The second dimension is number of sentences in each essay and the last is number of words in a sentence. For character sequence we generate same sequence for the first two dimension, but the last dimension is sequence of characters in a sentence. For short answer task, we have two input one is model answer and other is student answer. Unlike, essay statement, for short answer task our model expect optional model answer (abstract and summarized correct answer) as reference. So, our sequence generator module looks for how words and or characters are arranged in answer statement. This

shows dynamic nature of our model for model answer dependent and non-dependent short answer questions. In two input case (student and model answer), we generate 2 dimensional output for both input otherwise it generates single sequence like essay. Similarly, the first dimension of short answer is total number of answers used for training. The second is number of words in answer or number of characters in answer for character sequence. This is because of the nature of answer for short answer is short (usually from phrases to sentences).

Since, embedding layer expects fixed length sequence, the generated sequence less than selected threshold is padded and sequence greater than threshold value is truncated. For example, if we have 12000 essay and maximum threshold value selected are 5 words and 3 sentence, we create (1200, 3, 5) dimensional array stored in multi-dimensional Array. If the length of sentence in essay is less than 3 we fill it with <PAD> special token for 3 minus length sentence in essay times. If it exceeds 3, any sentence from greater than three is truncated. We do same for word and character sequence. To minimize information loss, we will consider threshold value based on maximum value on which more than 96% of dataset satisfy.

The other important task performed in this module is generating word and character embedding for each words and character vocabulary items. For word embedding, we use the result word vector building module FastText embedding matrix. Using character model, with one hot vector means taking each character as meaningful vector. But, we can infer meaning of characters from our training dataset. To train character embedding from large dataset, it is computationally inefficient approach. We can infer embedding of characters from word embedding as words are constituent of characters. For example, from the embedding of the word "ዘገበ", we can infer the embedding for "ዘ", "ገ", and "በ", and average the ዘ/ገ/በ vectors from *all* words in the dataset corpus. The proposed algorithm for word and character embedding generator is depicted in Algorithm 4.2.

Algorithm 4-2*: Algorithm proposed to extract word and character embedding from pre-trained FastText model SQM*

	Word and character Embedding extractor Algorithm
1.	**Input**:
2.	E: 2D ARRAY //FastText pre-trained word embedding matrix
3.	CHAR_VOCAB: DICTIONARY // character level vocabulary that contain unique characters in SQM dataset to their index

	Word and character Embedding extractor Algorithm
4.	WORD_VOCAB: DICTIONARY // word level vocabulary that contain unique words in in SQM dataset to their index
5.	EMBEDDING_DIM: INTEGER //embedding dimension of the vector. It should be equal to the feature dimension of pre-trained embedding
6.	**VARIABLE:**
7.	CHAR_VECTOR: DICTIONAR // variable that hold cumulative sum of word vectors on which character exist to frequency of characters. For example, if two words are 'ጠረጃ' and 'ደረጃ', the variable holds summation of two word vectors from our embedding and 2 its occurrence as value and the character 'ረ' as key. Here dictionary takes character as key and tuple with two elements (vector, frequency) as value.
8.	WORD_VOCAB_LENGTH: INTEGER // length of WORD_VOCAB
9.	CHAR_VOCAB_LENGTH: INTEGER // length of CHAR_VOCAB
10.	**OUTPUT**:
11.	CHAR_VECTOR_MATRIX: 2D ARRAY // FastText character embedding for character in character dictionary. Shape is length of CHAR_VOCAB times EMBEDDING_DIM
12.	WORD_VECTOR_MATRIX: 2D ARRAY // FastText word embedding for word in word vocabulary. Shape is length of WORD_VOCAB times EMBEDDING_DIM
13.	**BEGIN:**
14.	READ word vectors from E // loading and reading pre-trained FastText embedding. E is matrix of N-dimensional vector representation of each unique words in word embedding training dataset (the global or domain dataset).
15.	WORD_VOCAB_LENGTH = LENGTH(WORD_VOCAB)
16.	INTIALIZE WORD_VECTOR_MATRIX with shape WORD_VOCAB_LENGTH times EMBEDDING_DIM filled by ZEROS // Filling with zeros allows us to initialize zero embedding for 'PAD' key word. Our vocabulary has

	Word and character Embedding extractor Algorithm
	especial word 'PAD' in first index to assign common index for padded dummy word.
17.	FOR EACH Word W IN WORD_VOCAB
18.	WORD_VECTOR_MATRIX**[**WORD_VOCAB[W]] = E[W] // WORD_VOCAB[W] is index of word W. Here we assigning vector for word W from E. If word exist in E it extracts its vector otherwise it infers vector for new word based on character n-grams (morphemes) it share it words in vocabulary of E**.**
19.	FOR EACH char C IN W
20.	IF C IN CHAR_VECTOR THEN
21.	CHAR_VECTOR[C]=(CHAR_VECTOR[C][0] + V, CHAR_VECTOR[C][1] +1) // Increment occurrence of C and add new word vector to existing. The second index on tuple takes occurrence.
22.	ELSE // C is occurring for first time so we set frequency 1
23.	CHAR_VECTOR[C]=(V,1)
24.	END IF
25.	NEXT // Repeat step 19 for each character
26.	NEXT // Repeat step 17 for each word in WORD_VOCAB
27.	CHAR_VOCAB_LENGTH = LENGTH(CHAR_VOCAB)
28.	INTIALIZE CHAR_VECTOR_MATRIX with shape CHAR_VOCAB_LENGTH times EMBEDDING_DIM filled by ZEROS // Initializing character vector variable is same except size of CHAR_VECTOR_MATRIX first dimension depends on length of CHAR_VOCAB.
29.	FOR EACH char C IN CHAR_VOCAB
30.	IF C IN CHAR_VECTOR
31.	CHAR_VECTOR_MATRIX[C]= CHAR_VECTOR[C][0] / CHAR_VECTOR[C][1] // Average cumulative sum of C's vector with its

	Word and character Embedding extractor Algorithm
	occurrence. CHAR_VECTOR[C][0] if first index of value in CHAR_VECTOR with key 'C'. It is summed vector of C
32.	END IF
33.	NEXT //Repeat step 29 for each character in CHAR_VOCAB
34.	RETURN CHAR_VECTOR_MATRIX and WORD_VECTOR_MATRIX
35.	**END**

Finally, word and character vocabulary, FastText word and character vector matrix and sequence of word and charcters indices is passed to embedding layer.

i. Character Representation Module

Given a sequence of character index with character embedding and vocabulary of characters, our character representation module learns context of each characters in a word using convolutional neural network. Let $\{a_1, \cdots a_t\}$ represent a sequence of words in input answer where *t* is maximum sequence length of the sentence. Character representation module use CNN to convolve through sequence using characters bi-grams, tri-grams, quarter-grams, etc., and learn organization of characters in a word. Character level modelling enables us to deal with common miss-spellings and different morphological variety of words that are more common in languages like Amharic.

Below, we will give detail description of the proposed character-level temporal convolution neural network (2-dimensional convolutional network).

Convolution Layer

Let *C* be the vocabulary of characters, *d* be the dimensionality of character embedding's, and $Q \in \mathbb{R}^{d\ x\ |C|}$ be the matrix character embedding's. Suppose that word $K \in V$ is made up of a sequence of characters in $\{m_1, \cdots m_l\}$ answer, where *l* is length of word *K* in sequences. Given $C \in \mathbb{R}^{d\ x\ |l|}$ matrix representation of word (of length *l*), $Q \in \mathbb{R}^{d\ x\ w}$ convolutional filter matrix where d is dimensionality of character embedding and *w is* width of convolution filter (e.g., 1, 2, 3, 4, 5). Our character representation module represents word context in the following two steps:

1. Apply 2D convolution between C and Q. After which we add a bias and apply a nonlinearity to obtain a vector feature map $f \in \mathbb{R}^{l-w+1}$:

$$f^k[i] = RELU(\langle C[*, i{:}\, i + w - 1], Q\rangle + b) \quad (4.4)$$

, where $C[*, i{:}\, i + w - 1]$ is the *i*-to-(*i*+*w*−1)-th column of C^k and

- $\langle A, B\rangle$ is a Frobenius inner product (component-wise inner product of two vectors matrices).
- $b \in \mathbb{R}$ is a bias term
- ReLU*[64]* is a nonlinear kernel function layer that applies an element-wise activation function such as, $\max\{0, x\}$ threshold at zero.
- *Q* is a filter applied to each possible window of characters to produce a feature map $\boldsymbol{f^k}$ for word *K* in *V*.

2. Take the max-over-time as the feature corresponding to the filter Q (when applied to word k). The idea is to capture the most important feature the one with the highest value for a given filter. A filter is essentially picking out a character n-gram, where the size of the n-gram corresponds to the filter width. Maximum pooling used to get the representative maximum features is given as:

$$y^k = \max_i f^k[i] \quad (4.5)$$

The following example demonstrate how the proposed character representation works:

Let we have filter weight matrix $Q \in \mathbb{R}^{d\,x\,w}$, where dimension d=4 and filter w=3 and $C \in \mathbb{R}^{d\,x\,s}$, is our FastText character representation with dimension d=4 and s=5 characters of word '  ':

w=3

d=4

-0.1	0.5	2.2
0.7	0.9	0.3
-0.2	-0.2	0.7
1.3	-0.1	-1.1

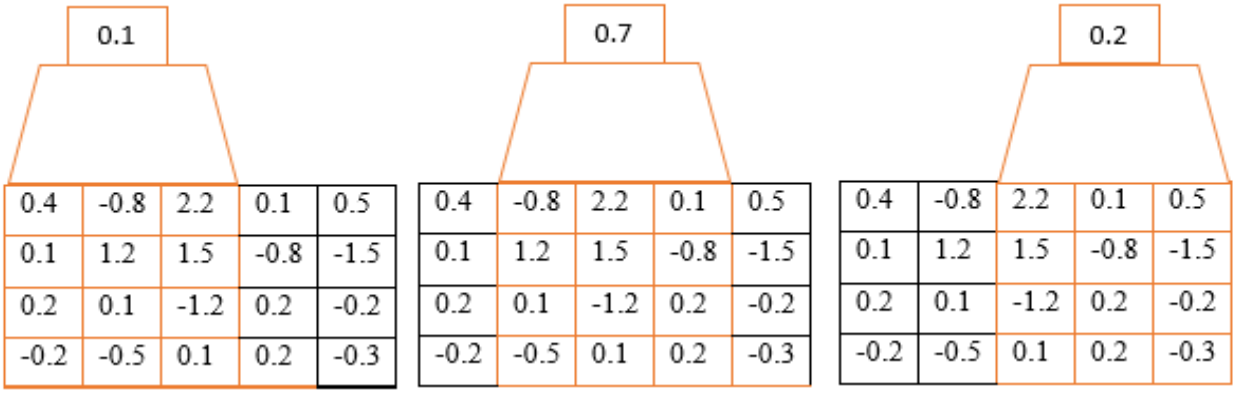
l=5 (Five characters)

0.4	-0.8	2.2	0.1	0.5
0.1	1.2	1.5	-0.8	-1.5
0.2	0.1	-1.2	0.2	-0.2
-0.2	-0.5	0.1	0.2	-0.3

d=4

Our Char CNN model first apply total of 3 (*s-w+1)* convolution over **C** and extract 1 submatrix for each filter of size 3 and applies component-wise inner product with Q(common for all filters) to get single representative value.

- $f^k[1] = \langle C[*, 1:3], Q \rangle$) applied on vectors of the first three characters '‍ᑭᕼᒣ'
- $f^k[2] = \langle C[*, 2:4], Q \rangle$) applied on vectors of the second to fourth characters 'ᕼᒣᑌ'
- $f^k[3] = \langle C[*, 3:5], Q \rangle$) applied on vectors of the third to fifth characters 'ᒣᑌᔆ'



Then we apply non-linearity RELU function with bias vector $b \in \mathbb{R}$ (similarly as Q, bias b is also learned by backpropagation) and on each feature map f^k.

- $f^k[i]$=RELU($f^k[i]$+$\vec{b}$)

Finally, we apply max-over-time pooling strategy over resulting feature maps to get only maximum value output vector.

- $y = \max_1 f^k[i]$

From our example when we apply max operation overf^k, we get character trigram 'ᕼᒣᑌ' as salient character sequence as it has maximum value 0.7.

We have described the process by which *one* feature is obtained from *one* filter matrix and how our max-over time function works. Our character CNN uses multiple filters of varying widths to obtain the feature vector for *k*. So if we have a total of *h* filters[7] $Q_1, \cdots, Q_h$ then $y^k = [y_1^k, \cdots, y_h^k]$ is the input representation of *k*.

[7] When we say filters region (a.k.a. kernel) it is character n-gram on which our Char CNN convolves over. We use varying size character n-grams (bi-gram, tri-gram, quarter-gram, etc.). We also use varying features (filters). Features are number of feature maps extracted from one n-gram size. If we use 2 filter region with 3 filters, that means we are applying 3 convolutions over 2 size n-grams. Multiple times in same region. So, output of our character CNN model is *l* times summation of filters *q* where *l* is sequence length.

ii. Word Representation (Embedding)

Word representation module represents each word in the answer with d-dimensional vector. We construct d-dimensional vector with two components: word embedding and character-composed embedding. The word embedding is a fixed vector for each individual word, which is pre-trained with FastText. The character-composed embedding is the output of character representation module. The input to this module is results of previous two modules which are generated sequence for input answer, FastText word embedding matrix for each vocabulary words and CNN character representation and generate combined vector sequence that represent character and word meaning in answer.

The first step here is replacing sequence of indices returned from sequence generator sub module into sequence of vectors. For simplicity this work as lookup table. In word embedding matrix rows are indices and column is vector, so using index and vector we transpose sequence of indices to sequence of vectors. Here we take $E \in \mathbb{R}^{vxd}$ where E is word embedding matrix, V is size of word vocabulary and *D* is dimension, and copy *D* sized vector of word W_t from sequence *S* with size *l* where *t* is time-step and *l* is total length of word in a sequence to get $S \in \mathbb{R}^{lxd}$. Since, sequence is padded and truncated to fixed length we create fixed length embedding sequence with length *l*.

Once we have sequence of word level embedding matrix, next step is concatenating each word vectors in a sequence to respective character embedding's from char CNN.

The output from our CNN character model output $C \in \mathbb{R}^{qx|l|}$is sequence of matrices where q is dimension equal to summation of filters used. Each matrix C_t in *C* is sequence of character vector as words are given to the model as a sequence of characters.

So, when we concatenate word embedding to its sub word character CNN representation we get embedding *E*:

$$E_t = S_t \oplus C_t \tag{4.6}$$

, where C_t is the CNN encoding of characters in a t^{-th} word of *S*

- S_t is t^{-th} word embedding from sequence *S.*
- E_t is the concatenation of two embedding's for t^{-th} word in sequence *S*

- $\oplus$ is concatenation operator.

When we apply our concatenated embedding E for all words in a sequence S with length l we get sequence of word representation enhanced by its sub word $E \in \mathbb{R}^{(q+d)x|l|}$ where $(q+d)$output feature dimension is summation of character dimension q and word dimension d:

$$E = [E_1, E_2, \cdots, E_l] \tag{4.7}$$

By concatenating the embedding's, we implicitly preserve the order of the characters: the embedding for e.g., the first character of a word will always correspond to the same portion of the input vector. Even if word is not occurring in our FastText embedding vocabulary (possibly occurs because FastText predict for unknown words if word share character n-grams with FastText vocabulary words), we can still model the embeddings for out of vocabulary (OOV) words with the help of their characters. By doing so our model reduces the number of errors made immediately after OOV words.

4.6 Encoding Module

In SQM, input text is not restricted i.e., it may range from phrase to paragraphs; may also extends to multiple paragraphs for essay type questions. So beyond word level context, sentence and paragraph level semantics between input answer is needed. To utilize contextual information appearing in input answer, we proposed to apply two level of contextual encoder that are sentence and paragraph level. As RNN analyze data sequentially for problems that work on sentence level it is suitable encoder. Since words are constituent of sentences, we have to know the meaning of word to understand or represent the meaning of sentence. To this analogy the bidirectional RNN encoder use word embedding vectors as input and sequentially analyze these word vectors in forward and backward direction. The output at each end will be merged to represent contextual information that the sentence holds.

In this process RNN analyze words how they are structured through a sentence using sequence and what meaning is encapsulated in a word with the help of word and character level meaning vectors provided. It accepts d-dimension word vector for each word in the answer and output answer matrix that represent contextual information. Again when we lift up to essay we should know the information that a sentence denotes in essay. Following [53], we again apply same encoder with sentence level vector inputs at essay level that allows to learn coherences with how

sentences are organized across the essay. By doing so our model can learn how sentences are structured in whole training set and learn their representation.

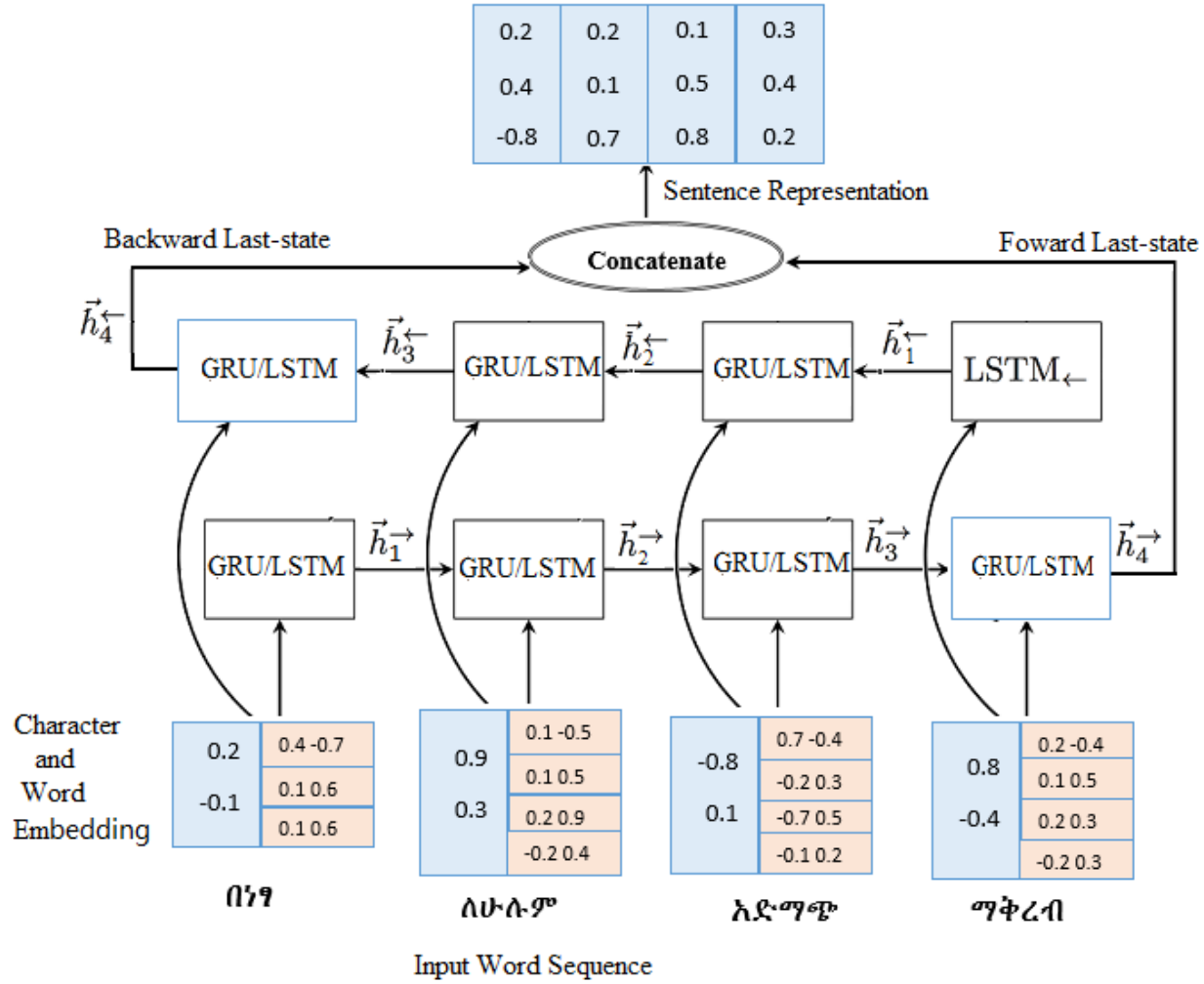


Figure 4-3*: Proposed Bi-directional RNN (GRU/LSTM) Encoder that represent contextual representation of words in input answer. In the above figure each word input is represented with our FastText 2-D embedding and 2D CNN-character representation for each characters in a word. Each hidden state of previous input is passed as past information to current input for both forward and backward representation and the concatenation of both forward and backward representation is taken as sentence context. The above figure depicts 1-layer bidirectional RNN (GRU/LSTM). When we use more than one layer the last representation of 4-vector matrix is passed as input to next layer and same process is applied to extract more enhanced feature.*

We use deep bidirectional recurrent neural network (LSTM/GRU) [54] to get context of words by capturing important information from both directions for sequence of words in input answer. At each time step *t* (for each word), the model maintains two hidden states, one for the left-to-right $\overrightarrow{h^t}$ (forward direction) and the other for the right-to-left $\overleftarrow{h^t}$(backward direction). Then we

concatenate the hidden state of two forward and backward hidden states as depicted in Figure 5.3. We use deep bidirectional RNNs by replacing each hidden sequence h^n vectors with the forward and backward sequences $\overrightarrow{h^n}$and $\overleftarrow{h^n}$ and ensuring that every hidden layer receives input from both the forward and backward layers at the level below. This allows us to detect enhanced sentence level or essay level representations generated by multi-layer bidirectional RNN by encapsulating the character and word levels information (vector). Figure 4.3 shows how our bidirectional encoder encode sentence context.

Given sequence of word previous module output word vector sequence $E \in \mathbb{R}^{dx|l|}$, our Bidirectional RNN (LSTM/GRU) encode each sequence in a sentence and results sentence matrix $S \in \mathbb{R}^{cx|l|}$,where *C* is low dimensional space representation of sequence, using Equation 4.9 and 4.10.

$$h_t = z_t \odot \tilde{h}_t + (1 - u_t) \odot h_{t-1}$$

$$\tilde{h} = ReLu(W[x_t] + U(r_t \odot h_{t-1}) + b)$$

$$z_t = \sigma(W_z[x_t] + U_z h_{t-1} + b_z)$$

$$r_t = \sigma(W_r[x_t] + U_r h_{t-1} + b_r)$$

RNN 1: Proposed GRU [55] Transformation Equation: where r and z are reset and update gates respectively; $h_{t-1}$is previous hidden-state output and $x_t$is current input (word vector); W and U learnable weights and b is bias; ReLu and σ(Sigmoid) are non-linearity activation functions

$$h_t = O_t \odot \mathrm{ReLu}(C_t)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

$$\tilde{C}_t = ReLu(W_c[x_t] + U_c h_{t-1} + b_c)$$

$$O_t = \sigma(W_o[x_t] + U_o h_{t-1} + b_o)$$

$$i_t = \sigma(W_i[x_t] + U_i h_{t-1} + b_i)$$

$$f_t = \sigma\left(W_f[x_t] + U_f h_{t-1} + b_f\right)$$

RNN 2:Proposed LSTM [35] Transformation Equation: where $x_t$is input at time-step t; W and U learnable weights and b is bias; ReLu and σ(Sigmoid) are non-linearity activation functions; and i, f, o and c are the input, forget, output gates and the cell activation vectors respectively.

$$\vec{h}_i = \overrightarrow{RNN}(x_i) \qquad i = 1, \cdots, m \tag{4.9}$$

$$\overleftarrow{h}_i = \overleftarrow{RNN}(x_i) \qquad i = m, \cdots, 1 \tag{4.10}$$

, where $\overrightarrow{RNN}$ is forward and $\overleftarrow{RNN}$ backward GRU/LSTM, $x_i$is input word vector (concatenation of word and character representation) at time *i* where *i* ranges from 1 to sequence length *m* for

forward and *m to* 1 for backward direction. Same equation is applied for both backward and forward RNN using equation represented in the above *RNN 1* and *RNN 2* for GRU and LSTM respectively. Finally, we concatenate sequence hidden matrix of answer $\vec{h} \in \mathbb{R}^{d\ x\ l}$with $\overleftarrow{h} \in \mathbb{R}^{d\ x\ l}$ to form the sentence representation $S \in \mathbb{R}^{(n*(2*d))xl}$. We refer to *n* as the number of RNN layers used, *d* as last hidden state dimensionality for forward or backward direction RNN and *l* is number of time-step (or sequence length).

For essay type question, we repeat the step in sentence encoder representing each sentence as single vector as shown in equation 4.9 and 4.10. Then, concatenation of forward and backward network is passed to attention layer.

4.7 Attention Module

This module is the core layer within our model that clues the next layer to predict score. For both short answer and essay type questions, we proposed different level attention mechanism. As discussed, in related work section, short answer assessment depends on two input strategy. One is strategy is only using student answer and the other is providing model answer as reference to predict score of student answer. The attention mechanism we proposed to employ on short answers is to infer which student answer vectors are more informative to given reference model answer from all word vectors encoded in student answer. The purpose of this attention is to couple the model and student answer vectors and produces a set of model answer aware feature vectors for each words in student answer. Under this module we proposed two step attention. The first is to reward sentences that are clues to correctly assess student answer, here we use attention mechanism at sentence level context vector by measuring how each sentence vectors in student answer are important in context of model answer. For simplicity, we called it *reference attention flow*. The other is responsible for fusing information from the model and the student answer concepts. Unlike popular attention approaches used in language modeling tasks [56], the answer-to-answer attention is not used to encapsulate the model and student answer concepts into single feature vectors. Instead, we adapted state-of-art bi-directional attention [57] model proposed for machine comprehension task with slight modification at comparison layer and we called it *answer-to-answer attention flow*.

For essay questions, since it has no reference answer usually raters looking for organization of an idea and searching for whether each terms included are informative to what the essay taking about or not. Even if it is challenging to get main topic about the essay without reference answer, we can still infer representative vector by matching each word vectors element wise [47]. Essay statements may range to multiple paragraphs and usually domain raters expect coherence analysis for essay than short answer. Not all terms included under student statement are informative to essay score. As shown in Figure 4.1, to get more important content when constructing the essay representation, we will be using Hierarchical Attention mechanism [53] that mirrors hierarchical structure of an essay. Two level of attention is applied in a given input with hierarchical attention. One is to look at words that are more relevant in sentences. In this case we first get maximum representative vectors using Global Maximum Pooling[8] from entire essay and apply *reference attention* on each sentences in an essay. The other attention is applied at essay level based on output of sentence level attention that aims to get most relevant sentence vector in essay.

i. Reference Attention Flow

The idea of *reference attention flow* was inspired by the observation of human raters when scoring student answer. When human rater assesses one answer, people usually can roughly form an intuition about which part of the answer is more important according to reference answer provided or meaning of words included in a sentence for the case of answer with no reference answer. First they skim all paragraph then point out the attentive sentence or phrase in student answer based on reference answer information. Using this idea, we design sentence level attention for each sentences in student answer. Specifically, we first encapsulate maximum average pooling of answer as context vector and use this vector to measure relevancy of each sentences in student answer. Here the context vector m_c can be seen as a high level representation of a model answer over all concepts used in memory networks by representing model answer. The attention mechanism is formalized as follows:

$$R_t = Relu\left(m_c + WS_{h_t} + b\right) \tag{4.11}$$

$$\alpha_t = softmax(R_t) \tag{4.12}$$

[8] https://keras.io/layers/pooling/

$$O = \sum \alpha_t S_{h_t} \tag{4.13}$$

, where m_c is global maximum pooling[4] of answer statement (if answer has model answer we use it as reference instead of self-representative vectors) given as weight, $S_{h_t}$is matrix consisting of output vectors of BRNN at time step *t*, *W* and *b* are learnable weight and bias respectively, α_t is attention vector at time step *t*, and *O* is attention weighted sentence vector in student answer.

In general, the intuition behind reference attention flow is it select the most important vectors from each time step of student answer and weight it with a learned multiple of a provided reference answer vector. Finally, we get attention weighted student answer vector.

Before passing final result to next module, we again contextualize the result of reference attention flow with BRNN to get enhanced context information of student answer.

ii. Answer-to-Answer Attention

This attention used when we score short answer with reference model answer. Unlike Reference attention, this attention analyzes a given input pair in two directions i.e., from model answer to student answer and student to model answer. The difference from *reference attention flow* is it allows us to capture how two vectors in encoded sequence are related whereas reference attention flow give clue which vector does the network attend to predict.

The inputs to the layer are contextual vector representations of the model answer M and the attention weighted student answer S. The outputs of the layer are the model answer-aware vector representations of the student answer concepts, G, along with the contextual embedding from the previous layer.

The attention is computed in two directions: from model to student as well as from student to model. Both of these attentions, which will be discussed below, are derived from a shared similarity. The enhancement we made here is the original paper used dot product to define shared similarity whereas we design cosine similarity between two tensor objects. The inputs are processed in two directions and the final result is merged using element-wise concatenation.

The similarity matrix $S \in \mathbb{R}^{t\,x\,j}$ shared between the contextual embeddings of the model answer (M) and the student answer (S), where S_{tj}indicates the similarity between t^{-th} model answer concept and j^{-th} student answer concept, is given by

$$S_{tj} = cosine\left(M_t, S_j\right) \qquad t, j = 1, \cdots, N \tag{4.14}$$

Here cosine similarity is applied for each word vectors in both student and model answer and concatenated across the row. Now we use **S** as weight to obtain the attentions and the attended vectors in both directions.

Model Answer-to-Student Answer Attention (M2S): signifies which student answer concepts are most relevant to each model answer concepts. Let $a_t \in \mathbb{R}^J$ represent the attention weights (S_{tj}) on the student answer concepts by t^{-th} model answer concept, $\sum a_{tj} = 1$, for all *t*. The attention weight is computed by

$$a_t = softmax(S_{t:}) \in \mathbb{R}^J \tag{4.15}$$

, and subsequently each attended student answer vector is

$$\widetilde{U}_t = \sum_j a_{tj} U_{:j} \tag{4.16}$$

, where $\widetilde{U}$ is a 2*d*-by-*T* matrix containing the attended student answer vectors for the entire model answer.

Student Answer-to-Model Answer Attention (S2M): signifies which model answer concepts have the closest similarity one of the student answer concepts and are hence critical for scoring the student answer. We obtain the attention weights on the model answer concepts by

$$b_s = softmax(max_{col}(S) \in \mathbb{R}^T) \tag{4.17}$$

, where the maximum function (max_{col}) is performed across the column. Then the attended context vector is

$$\tilde{h}_s = \sum_j b_{st} h_{:t} \in \mathbb{R}^D \quad (4.18)$$

This vector indicates the weighted sum of the most important concepts in the model answer with respect to the student answer. $\tilde{h}$ is tiled *T* times across the column, thus giving $\widetilde{H} \in \mathbb{R}^{DxT}$.

Finally, the contextual embeddings of model answer and the attention vectors are combined together to yield G, where each column vector can be considered as the student answer-aware representation of each model answer concept.

We define G by

$$G_t = concatenate\left(M_t, \widetilde{U}_t, \widetilde{H}_t\right) \in \mathbb{R}^d \quad (4.19)$$

, where G_t is the t^{-th} column vector (corresponding to t^{-th} model answer concept), $concatinate$ is a method used to merge input vectors ($M, \widetilde{U}, \widetilde{H}$), d is the output dimension.

iii. Hierarchical Attention

Unlike reference and answer-to-answer, we will be using hierarchical attention for essay questions. The idea of hierarchical attention is same for both word and sentence level vectors except input varies. For sentence level attention we apply encoded word vectors whereas essay level attention we apply sentence vector. Similarly, we apply Equation 4.11, 4.12, and 4.13 as we are applying similar idea with reference attention except reference attention expects summarized model answer vector whereas with hierarchical sentence and essay level attention, vector is maximum pooling of sentence or essay. Hierarchical attention mechanism used for essay sentence level vector and essay level vectors is formalized as follows:

$$R_t = Relu(m_c + WE_t + b) \quad (4.20)$$

$$\alpha_t = softmax(R_t) \quad (4.21)$$

$$O = \sum \alpha_t E_t \quad (4.22)$$

, where m_c is global maximum pooling of entire essay as weight, $E_t$is matrix consisting of output vectors of BRNN at time step *t*, *W* and *b* are learnable weight and bias respectively, α_t is attention vector at time step *t*, and *O* is attention weighted sentence or essay vector in for given essay.

4.8 Modeling Module

The input to the modeling layer is G, which encodes the student answer-aware representations of model answer concepts. The output of the modeling layer captures the interaction among the model answer concepts conditioned on the student answer. This is different from the contextual embedding layer, which captures the interaction among model answer concepts independent of the student answer concepts. We use multi layers of bi-directional RNN, with the output size of *d* for each direction. Hence we obtain a matrix $M \in \mathbb{R}^{2d\ x\ T}$, which is passed onto the output layer to predict the score. Each column vector of M is expected to contain contextual information about the concept with respect to the entire model answer concepts and the student answer concepts.

4.9 Scoring Module

The input to this layer is output of modeling module matrix *M* and score range. Given *M* and score range (varying depending weight assigned to question), the model tries to predict probability distribution of scores. We consider short answer assessment task as regression problem. Here the model predicts continuous score in the expected range of scores (0 to 5) and our objective is to minimize the square error between the predicted scores and the actual scores. The objective in this case objective was Mean Squared Error. Given a set of predictions $\hat{y}$ and the true grades y, we sought to minimize:

$$J = \frac{1}{N}\sum_{i}^{N}\left(\hat{y}^{i} - y^{i}\right)^{2} \tag{4.23}$$

CHAPTER FIVE

5. EXPERIMENTATION AND EVALUATION OF SUBJECTIVE QUESTION MARKING (SQM)

5.1 Overview

This chapter aims to provide a detailed evaluation of the approach in addition to experimental environment used to develop SQM. The chapter evaluates SQM system component wise. As subjective question we experimented and evaluated essay as subjective question and short answer questions. The following sub sections give detailed description about component wise evaluation we conducted and including dataset preparation.

5.2 Data Preparation and Analysis

SQM needs two different data for different dataset for FastText model training and for evaluation of the design scorer. The proposed model semantics highly depend on quality of word vectors created. So, generating Amharic FastText vector is core step in SQM. To achieve informative vectors, we have collected large sized data from different sources. Table *5.1* depicts statistics of data collected from different sources. In addition to word embedding dataset, we need number of scored Amharic short answer questions for training and testing the model. SQM has two sub components. One is a model that assess essay questions as subjective examination and the other deal with short answers. As we discussed in previous chapter both models has different but related architecture. So we need different dataset for both models. We experiment our essay model with publicly available English dataset prepared for Hewlett Foundation's Automated Student Assessment Prize competition by Kaggle[9]. For short answer, we are evaluating on both Amharic datasets prepared for this thesis work and publicly available English dataset [50]. In the following section we discuss statistics behind the datasets for both essay and short answer in addition to Amharic FastText word embedding training dataset.

5.2.1 Dataset for Word Embedding

The only feature provided for SQM model is word vector created from unlabeled dataset. So, better achievement of scoring module totally depends up on quality of word vectors created from word

[9] https://www.kaggle.com/c/asap-aes

representation module. As we discussed in chapter four, we have proposed FastText word embedding model. To identify word context in different situation we have considered social, sport, political, and business sub domains for news domain; bible, blogs, and written documents from spiritual domain; Amharic Wikipedia; and three selective course modules[10] collected from Jimma University Department of Amharic Language and Literature. In addition we comprised of all student answers collected for evaluating SQM as additional domain dataset. To collect web dependent data, we used HTTRACK Website Copier as offline crawler to copy files from web. Python based BeautifulSoup[11] library is used to extract text content from web files crawled. Then after small preprocessing such as tokenization and normalization we used to train FastText model that can able to extract Amharic word meaning from given corpus.

Table 5-1*: Table that depict statistics of data collected to train FastText for Amharic word vectors*

Domain	**Statistics**
Total Document	32,941
Total Tokens	40,816,929
Vocabulary Size	275,829

5.2.2 Dataset for SQM

As discussed above, we used two distinct datasets for essay and short answer assessment. For short answer we experiment on Kaggle short answer dataset for English and Amharic dataset collected for this thesis purpose. Because of time constraint essay part of our model is experimented with only standard publicly available Kaggle dataset. In the following subsection will narrate preparation and statistics of two datasets.

[10] The three selected modules are "የዘገባ አፃፃፍና አስተውሎታዊ እሳቤ (Report Writing and Critical Thinking), የቋንቋና ሥነልሳን ጥናት, and ህዝብ ግንኙነት (public relation)". No special criteria is used to select courses, except a copy of modules is easily accecible from the department.
[11] https://pypi.python.org/pypi/beautifulsoup4

i. Essay Dataset

In 2012, the Hewlett Foundation sponsored a competition on Kaggle called the Automated Student Assessment Prize (ASAP) and prepared standardized dataset as Kaggle AES dataset. The dataset contains 12,976 essays ranging from 150 to 550 words each, marked by two raters (Cohen's κ = 0:86). There are eight different sets of essays written by students ranging from Grade 7 to Grade 10. Each prompted by eight different prompts, with distinct marking criteria and score range. For our experiments, we use the resolved combined score between the two raters, which is calculated as the average between the two raters' scores (if the scores are close), or is determined by a third expert (if the scores are far apart).

***Table 5-2**: Kaggle AES dataset statistics*

Essay set	Essay Type	Score Range	Average word length	Total
1	Persuasive / narrative / expository	2-12	350	1785
2	Persuasive / narrative / expository	1-6	350	1800
3	Source dependent responses	0-3	150	1726
4	Source dependent responses	0-3	150	1772
5	Source dependent responses	0-4	150	1805
6	Source dependent responses	0-4	150	1800
7	Persuasive / narrative / expository	0-30	250	1730
8	Persuasive / narrative / expository	0-60	650	918

The competition used quadratic weighted kappa to measure the similarity between the human scores and the automated scores. Currently, the state-of-the-art on this dataset has achieved a

Cohen's κ = 0.*96* (using quadratic weights) [65]. The dataset originally released by Kaggle competition has not gold score annotated test set. However, following state-of-art Dimitrios *et al.*, [65] work, for our experimentation we are splitting the given training set to create a new test set. We follow same setup as Dimitrios *et al.*, [65] 80% of the entire dataset is used for training and validating the model and the left 20% is used for testing. In absolute term we split 64% training, 16% validation and 20% testing of entire dataset. Table 5.2 summarizes some characteristics of Kaggle dataset.

ii. Short Answer Dataset

An automatic short answer marking system is one that automatically assigns a grade to an answer provided by a student, usually by inferring from provided one or more reference correct answers. Traditionally, automatic assessment tasks more focus on essay questions than short answer. One reason is lack of standardized dataset. In 2012, Kaggle sponsored short answer scoring part of Automatic Student Assessment Prize (ASAP). The aim was to select best predictor system by giving graded short answer responses and their corresponding prompts. For task completion Kaggle released about 17,000 graded short answer. Unlike essay, achieved result in short answer was not promising. Best result of first winner was 0.771 kappa. That is because short answers are more subjective and more diverse than essay. Factual essay usually focus on fact and organization of sentences and ideas is required. Whereas, for short answers no common way to express the idea. It is open and left for student as they want. In addition to Kaggle, Mohler *et al.*, [50] released small sized data focused from introductory computer science course assignments with answers provided by a class of undergraduate students. The data set consists of a total of 2273 student answers.

1. Kaggle Short Answer Dataset

Kaggle short answer dataset[12] consisted of answer texts of approximately 50 words that cover a broad range of disciplines (from English Language Arts to Science) which were written by 10th grade students. Approximately 17,000 answer with two scores graded by two different people in total of 10 questions (sets) is provided for training. On average, each answer is approximately 50 words in length. Most training sets consist of about 1,800 responses. With training dataset, Kaggle also provided test data which consists of approximately 6,000 answers. However, the test set was

[12] https://www.kaggle.com/c/asap-sas

released without the gold score annotations, rendering any comparisons ineffective, and we are therefore restricted in splitting the given training set to create a new test set as we did in essay dataset.

As default feature, our SQM model for short answer expects two input answers as model and student answer. But, Kaggle dataset has no reference answer explicitly provided. Still we can train our model using by inferring answer representative vector using sentence level attention proposed. This shows the dynamic nature of our model to work with both situations. The following *Table 5.3* depicts some statistics on Kaggle short answer dataset including score distribution to data size in selected each sets.

From total of 17000 graded answer from Kaggle short answer dataset, 80% of the entire dataset is used for training and validating the model and the left 20% is used for testing. In total we split 80% training, 10% validation and 10% testing of entire dataset using 10-fold cross validation.

Table 5-3*: Kaggle short answer scoring dataset statistics per each scores assigned. '-' means score ranges between 0-2 for set*

Question /Set	**Average word per score points**				**Score distribution (%)**				**Data Distribution**	**Human Agreement (Kappa)**
	0	1	2	3	0	1	2	3	Total	
1	38	50	57	62	23	26	31	30	1672	0.86
2	36	50	68	-	38	58	8	-	1278	0.68
5	23	43	62	93	77	18	3	2	1795	0.91
6	22	41	55	76	84	9	5	3	1797	0.89

2. Amharic Short Answer Dataset

Unlike previous standard datasets, this dataset is created for the purpose of thesis completion to evaluate performance of SQM in Amharic short answer questions. We use two techniques to collect answer. One and ease technique applied was collecting pre-graded answer for selected course modules. We have collected examination papers already graded by course instructor for third year Amharic Language and Literature Department undergraduate students. Total of 84 student answers each with on average 7 answer set for 2 different courses named "የዘገባ አፃፃፍና አስተውሎታዊ እሳቤ (Report Writing and Critical Thinking) and አጣርኛ ቋንቋ ፎክሎር መግቢያ (Introduction to Foklore in Amharic)" is collected. Other technique applied is providing purposeful examination. With the collaboration to Jimma University Amharic Language and Literature department, we incorporate especial exam for third year summer students to "Public Relation (የህዝብ ግንኙነት)" course. Seven questions are pre-prepared from course module with the help of course instructor focused on objective of the thesis. Students are not informed anything about purpose of question except ordered by course instructor about structure and type of exam content. Total of 155 students sit on examination. From both techniques we have collected 1112 answers and provided to two independent raters. The answers were independently graded by two human domain raters, using an integer scale varying on question set as provided by instructor for each questions. Both human raters were Lecturer at Jimma University Amharic Language and Literature department; one is course instructor currently and the other is also familiar with the course as he instructed the course for regular class. We treat the average grade of the two raters as the gold standard against which we compare our SQM. The annotators were given no explicit instructions on how to assign grades. Both raters gave the same grade for 747 answers from total and approximately near grades for 202 answers. Inter rater correlation between two rates is 87 % Pearson and 89 % Spearman. Table 5.4 shows two question-answer pairs with three sample student answers each to show poor match or perfect match between student answer and raters score provided[13].

[13] We also included sample questions their model answer and score assigned by two raters in Annex E

***Table 5-4**: A sample question with short answers provided by students and the grades assigned by the two human raters*

	Sample question, model answer, and student answers	**Score**	
		Rater 1	Rater 2
Question	ሚዲያ ለህዝብ ግንኙነት ሙያ /ስራ ያለውን ጠቀሜታ አስረዳ/ጂ። （4 ነጥብ）		
Model Answer	ሚድያ ለህዝብ ለህዝብ ግንኙነት የሚያበረክተው ጠቀሜታ ብዙ ነው ። የተቋሙን የድርጅቱን እንቅስቃሴ መልእክት ለማስተላለፍ ያስችላል በቀውስ ወቅት የተቋሙን ደህንነት መልስ ለመገንባት ስለተቋሙ የሚወሩ አሉባልታዎችን ለማወቅና ለማስወገድ የሚቻለው በሚድያ ነው ። የተቋሙን ጊዴታ መልሶ ለመገንባት የሚቻለው በሚዲያ አማካይነት ነው ።ለተቋሙ ለአገር ግንባታ የሚያስፈልጉ ገንዘብ ለማሰባባሰብ ያግዛል ። ህዝብን ተደራሽን ለማሳመን ይጠቅማል የህግ ባለሙያ ያዘጋጀውን ዕቅድ ወደተግባር ለመቀየር ያስችላል ። ምርትና አገልግሎትን ለማስተዋወቅ ።		
Student 1	ከድርጅቱና ከማህበረሰቡ ጋር ያለውን ግንኙነት ያጠናክራል፣ ድርጅቱ /ተቋሙ የሚያቀርባቸውን አገልግሎቶ በቀላሉ ሊያሳውቃቸው ይችልበታል፡ ፡ ለደራሹ ወቅታዊ የሆኑ መረጃዎችን ያስተላልፉላቸዋል።	4	4
Student 2	ሚዲያ ለህግ ስራ ዋነኛ መሳሪያው ነው። የህግ ሰራተኛው ስራውን የፃፈውን ጽሁፍ በሚዲያ አማካኝነት ነው ወደ ህዝቡ ለማድረስ የሚችለው ህዝብን በየቀኑ ስብሰባ መጥራት አይቻልም። ነገር ግን በሚዲያ አማካኝነት የህግ ሰራተኛው ለህዝቡ በርካታ ስራዎችን በየቀኑ ማስተላለፍ ይችላል።	1	2
Student 3	ሚዲያ ለህዝብ ግንኙነት ሙያ ስራ ያለው ጠቃሜታ ለህዝብ መልዕክት ማስተላለፍ፣ ከተለያዩ አቅጣጫ የሚፈጠሩትን ትኩስና አዳዲስ ወቅታዊ የሆኑ ዜናዎችን በማስተላለፍ ይጠቆማል። በሀገር ውስጥም ሆነ ከውጭ	3	3

	Sample question, model answer, and student answers	**Score**	
		Rater 1	Rater 2
	የሚከሰቱት ክስተቶችን ህዝብ የሚከታተለው ከሚዲያ ነው ማለት ይቻላል።		
Question	የህዝብ ግንኙነት ሙያተኞች ነጻ የመሆን ስነ ምግባር ሊኖራቸው የገባል ሲባል ምን ማለት ነው? በሚገባ ግለጪ/ጽ (4 ነጥብ)		
Model Answer	የህዝብ ግንኙነት ባለሙያ መቸም ቢሆን ተጨባጩን እውነት ለማንም ሲባል ማጣመም ስለሚያስፈልግ ነው ። የህግ ስራ በተጨባጭ እውነት ላይ ካልተመሰረተ ጥፋት ማስከተሉ ስለሚያቀርበው የህግ ባለሙያ ነፃ ካልሆነ ሥራውን በሚገባ ሊያከናውን አይችልም ።		
Student 1	ለሚደርስባቸው ተጽዕኖ በመሸነፍና ለይሉኝታ በመጋለጥ ነገሮችን ሳያጣምመው ያለውን እውነታና ሀቅ ባለው ይዘት፣ ተቋሙንም ወደግሽፈት በማያደርስ /በማይመራ/ መልክ ማቅረብ እንዳለባቸው የሚገልጽ እሳቤ ነው።	3	1
Student 2	የህዝብ ግንኙነት ባለው ሙያ ለህዝቦች በአግባቡ ማገልገል አለበት።	0	0
Student 3	የህዝብ ግንኙነት ሙያተኛ ነፃ የመሆን ስነ ምግባር ሊኖራቸው ይገባል። ስል በተቋሙና በተደራሹ መካከል እንደ ድልድይ ሆነው ያለ ስነምግባር ስልት በተለያዩ ነገሮች ሳይዳሰሱ ለአንዱ ላይ ወግኑ በሁለቱም እኩል በመቆም የተቋሙን ወደ ተደራሹ የተደራሹን ደግሞ ወደ ድርጅቱ ሚዛናዊ በሆነ መንገድ ማቅረብ ይጠበቅባቸዋል። ከምንም አይንት ግፊት፣ ድለላ፣ ሙስና ለአንዱ ሳይወግን ወይም ለሌላው ክፍተት ሳያሳይ ነፃ ሆኖ መልዕክቱን ማስተላለፍና ቀጥተኛና ተዓማን መሆን አለባቸው።	3	4

On average, each answer is approximately 60 to 80 words in length. Most question sets consist of about 150 responses. Table 5-5 shows some statistics behind prepared Amharic short answer dataset and Figure *5.1* shows how score is distributed in Amharic SQM dataset.

Table 5-5*: Data visualization per question sets, number of answer per question set and inter-rater agreement in Amharic short answer dataset*

Question /Set	**Number of answer in each set**	**Inter rater agreement on question sets**
1	130	Totally agreed on 104 answer out of 130 (80%)
2	118	Totally agreed on 77 answer out of 118 (65%)
3	113	Totally agreed on 88 answer out of 113(78%)
4	113	Totally agreed on 83 answer out of 113(73.5%)
5	97	Totally agreed on 60 answer out of 97(62%)
6	115	Totally agreed on 78 answer out of 115(68%)
7	100	Totally agreed on 66 answer out of 100(66%)
8	36	Totally agreed on 35 answer out of 36(97%)
9	36	Totally agreed on 34 answer out of 36(94%)

Question /Set	Number of answer in each set	Inter rater agreement on question sets
10	36	Totally agreed on 31 answer out of 36(86%)
11	25	Totally agreed on 25 answer out of 25(100%)
12	25	Totally agreed on 23 answer out of 25(92%)
13	25	Totally agreed on 21 answer out of 25(84%)
14	25	Totally agreed on 22 answer out of 25(88%)

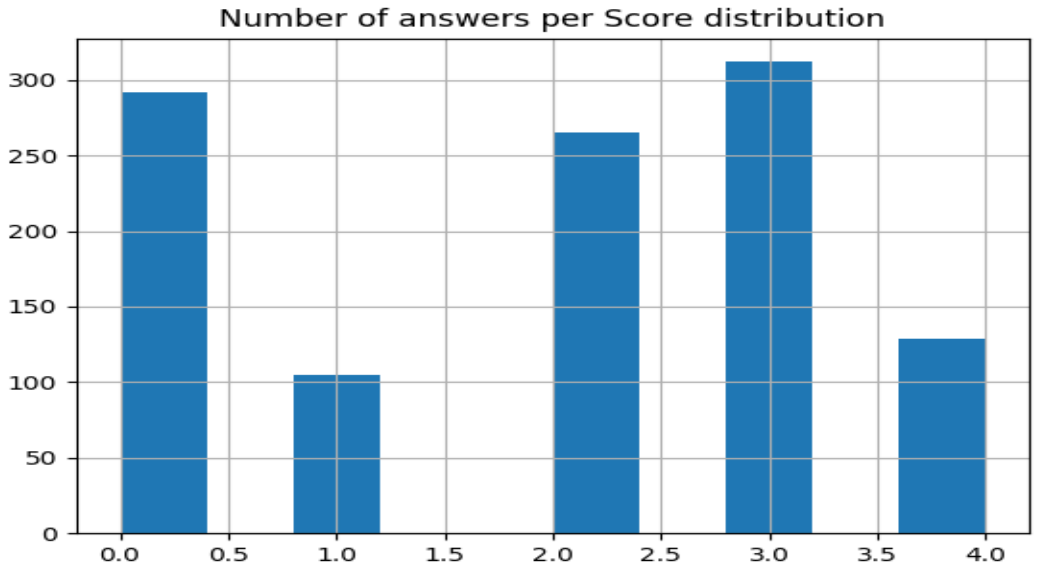


***Figure 5-1**: Visualizing Amharic short answer dataset how scores are distributed*

We follow same setup for data as we did in Kaggle short answer dataset. From total of 1112 Amharic short answer dataset we split 80% training, 10% validation and 10% testing of entire dataset using stratified k-fold cross validation techniques with k=10.

5.3 Experimentation

In this section, we evaluate our model for both essay and short answer on the above discussed datasets. Experimentation will be applied on each component to analyze how components favor for score prediction. We begin with visualizing and evaluating our FastText embeddings to ensure how vectors provide meaning to words appear in answer statement.

5.3.1 Experiments of FastText Word Vectors

We experimented FastText mainly for two objectives. One is Amharic FastText word vector that is used as the only feature to provide meaning words in answer. The other is English FastText word vector trained on domain dataset (essay and short answer).

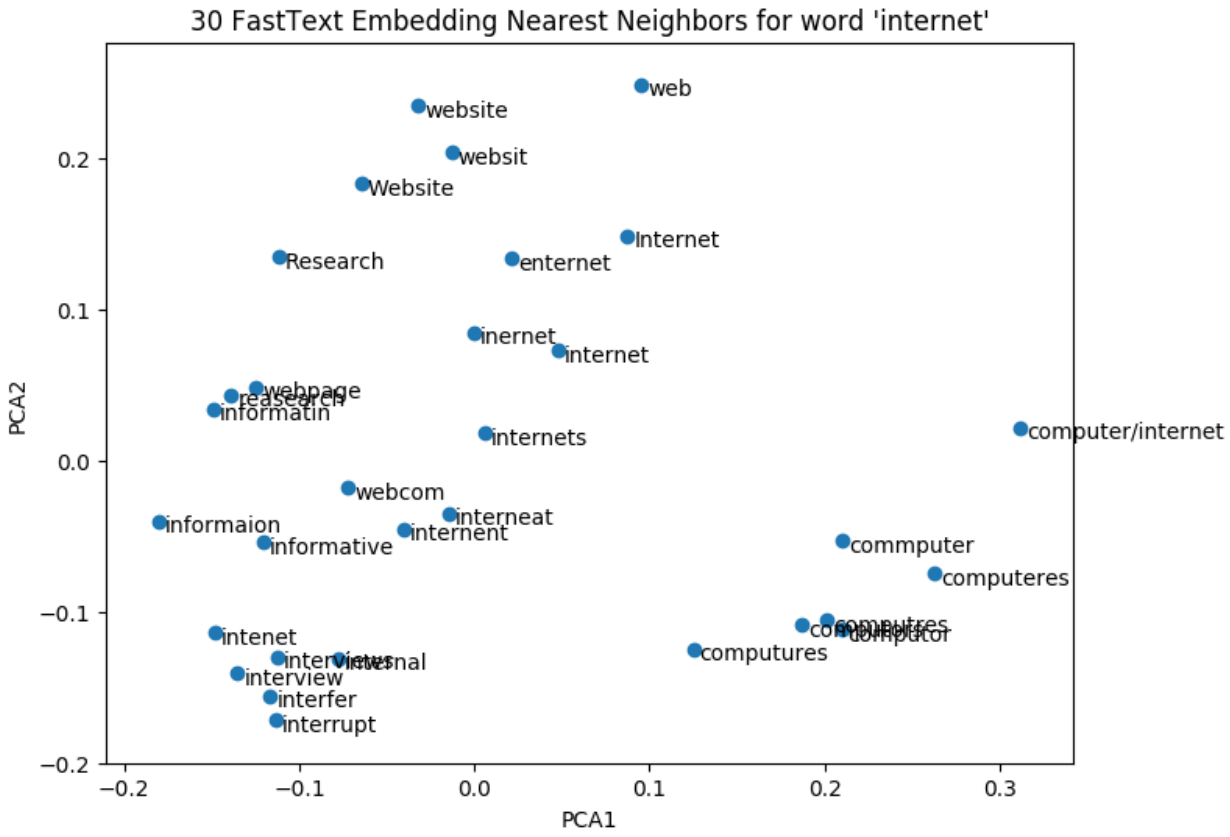


***Figure 5-2**: Visualizing most 30 similar words for 'internet' from FastText embedding trained on Kaggle*

To test how word vector models, perform well on domain data we trained FastText on Kaggle essay dataset using Skip-gram model with negative sampling. It is important to examine word embedding and see how words cluster together to their nearest words based on our FastText model. To visualize this, we applied Principal Component Analysis and reduce the word dimensionality to 2 components. We use PCA transformed word vectors and represent them on a 2D plot. We examine the top 30 most similar words to a word of our choice.

As shown on Figure 5-2, the result of our FastText embedding has ability to cluster words with spell-error such as ‘enternet and inernet’ to their semantically related words in vector space such as web, computer, information, and website. In addition, it also detected morphological variants of ‘internet’ such as ‘internets’ and detected capitalization also (‘Internet’).

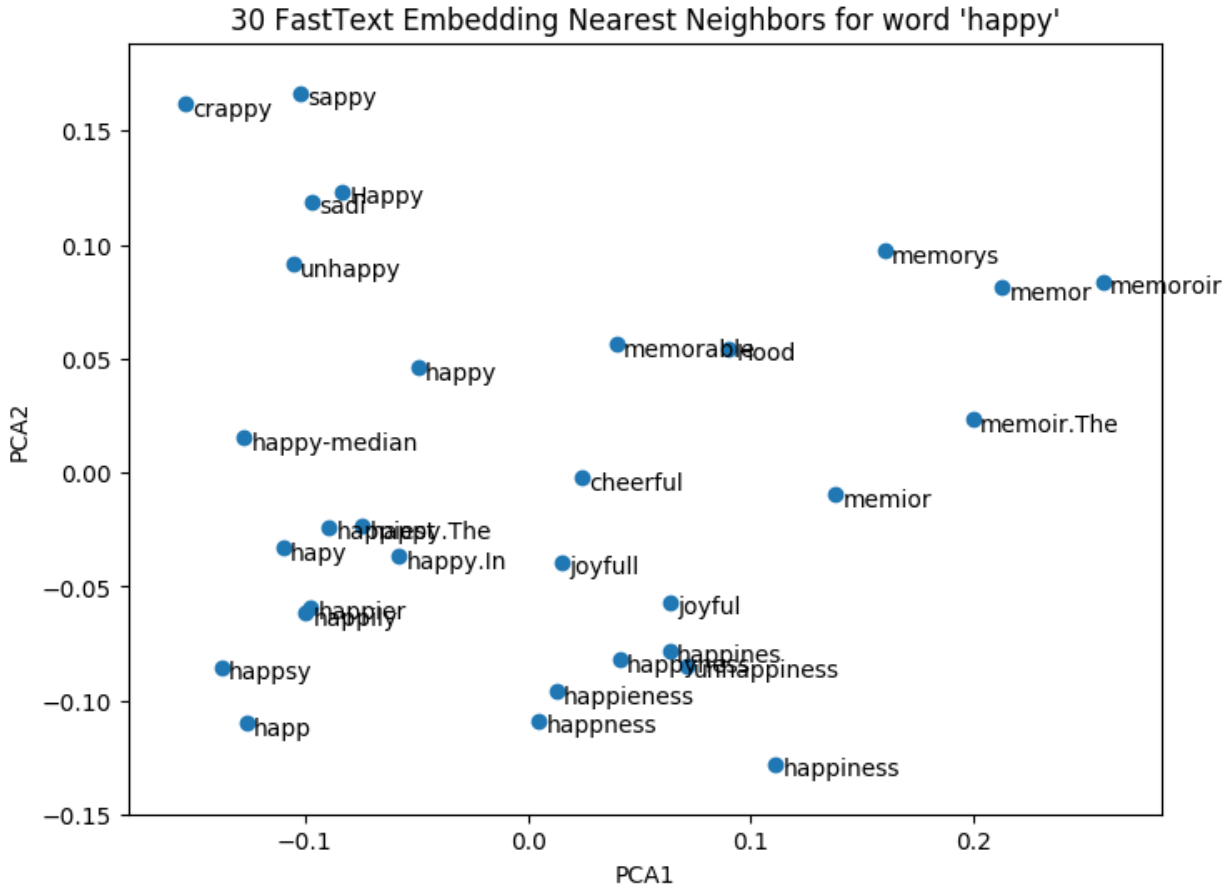


***Figure 5-3**: Visualizing top most nearest neighbors of word 'happy'*

We can also see from Figure 5-3, our FastText trained on domain Kaggle essay dataset clustered words such as sad and unhappy that are opposite words to given word ‘happy’ to same cluster and

synonyms such as cheerful and joyful are also detected. Another interesting feature is it detected vague word ‘memory’ as related word to happy. The word ‘memory’ also indicate celebration and our model related these two words based on contextual meaning.

Both CBOW and Skip-gram model trained on Amharic text with defined hyper-parameters. The parameters defined to train both CBOW and Skip-gram model is depicted *Table 5.6* below.

Table 5-6*: Hyper-parameters used to train both CBOW and Skip-gram models for our FastText vector builder*

Hyper parameter	**Value**
Window	5,10
Embedding dimension	100 ,300
Learning rate	0.05
Workers	30
Negative Sampling	10,15
Iteration	10
N-gram size	3,6

For Amharic we trained our model for both global and domain dependent dataset. The following Figure 5-4 and 5-5 shows how our FastText cluster word vectors related to same space. Moreover, we visualized how our FastText model handle morphologically related words.

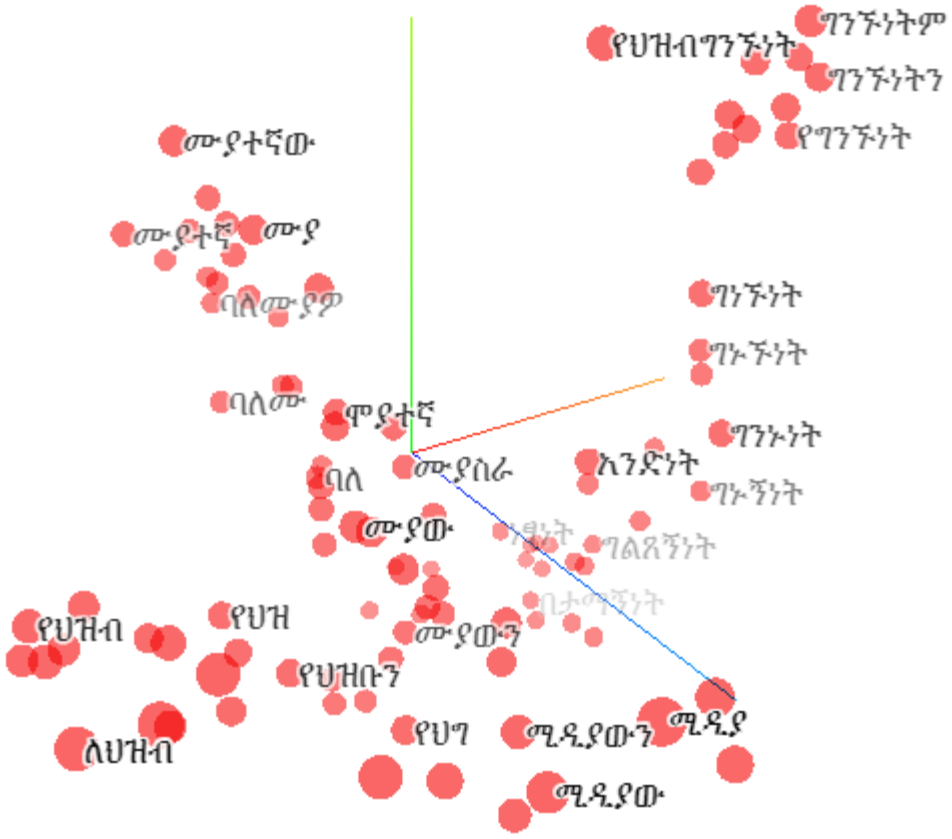


***Figure 5-4**: Visualizing morphologically related Amharic words in vector space*

From the above figure we can visualize that our domain trained FastText model is capable of clustering syntactically related word to their semantic space. Moreover, it detected word with spelling error ‘የህዝ’ to say ‘የህዝብ’. It interesting feature of our FastText model is its ability to cluster words with different morphological varietiy, but same in meaning. As we can see from Figure 5.4 different mophological variants of Amharic word is clustered to one their semaintically related words.

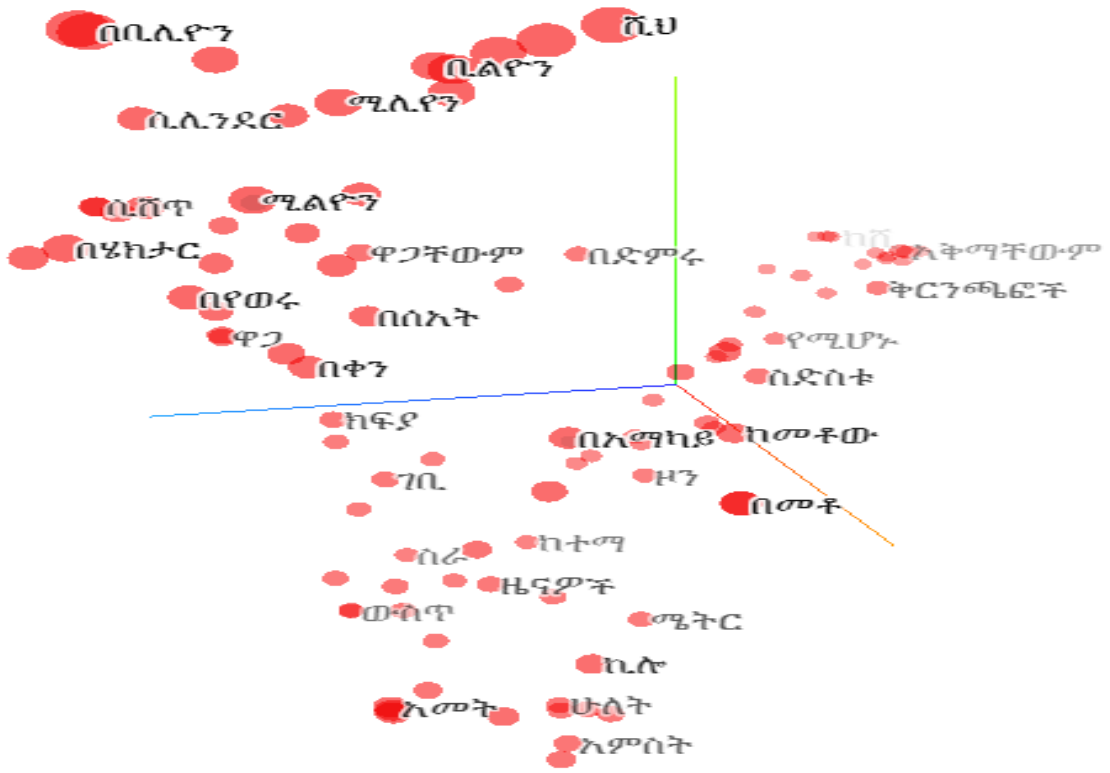


Figure 5-5*: Sample visualization of semantically clustered FastText embeddings in vector space*

Another interesting feature of our word representation module is its capacity to cluster words on their semantic category. From the above Figure our FastText model categorized Amharic words such ወር, ቀን, ሰዓት to one category using as time measurement. We can see that ሜትር and ኪሎ are clustered together. It clusters large number ሚሊየን, ቢሊዮን, ትሪሊዮን, and ሺህ to one; በአማካይ and በመቶ; ክፍያ, ወጪ and ገቢ at one category. Because of space limit we visualized only sample ones. This shows that our word vectors are semantically and syntactically rich and our SQM model easily get meaning of words using this embeddings.

5.3.2 Experiments of SQM

For both models different experimental setup is used. For both models we used Python version of Keras[14] 2 deep learning library that run on the top of Tensor flow. Keras made the complex nature of neural network user friendly. Keras is powerful, easy to customize and high level abstraction API for deep neural network. Keras made training neural network models easy by providing

[14] http://www.keraso.io/

training on line fashion and also has capability of storing current hidden layer best weight that support evaluation process. The other capability of Keras is it allows us to create user defined function for hidden layer network. As we have several attention approaches, Keras is suitable to experiment SQM.

For our FastText word vector building module, we used genism[15] wrapper class that allows us to use efficient original c-version Facebook FastText implementation. TSNE is used to visualize our embeddings on dimensionality space. Every algorithm we have developed are implemented using the Python programming language. Reason for using python is its ease feature and recent popularity in deep learning. In addition to the above main tools we used number of python dependency sub libraries. We have experimented our system on core i5, 1 Tera disk, 8 GB memory. This is because training deep learning requires powerful hardware components.

i. Experiments of Essay

Given student written essay, the essay as SQM task predict score in a defined range. For Kaggle essay score range differ from set to set. We created model friendly score by using the following equation: Given minimum score S_{min} and maximum score $S_{max}$of the given prompt or question set, we calculate model friendly score range that lay between 0 and 1 as:

$$S_i = \frac{S_i - S_{min}}{S_{max} - S_{min}} \quad (5.1)$$

, where S_i score for i-th answer question set. Similarly, for evaluation we reverse score to original range using:

$$S_i = S_i^* (S_{max} - S_{min}) + S_{min} \quad (5.2)$$

[15] https://radimrehurek.com/gensim/

Table 5-7*: Results of the different models on the Kaggle dataset. All resulting vectors were trained using linear regression. We used the dataset split released by Dimitrios at el. [47. ρ is Spearman and r is for Pearson's metric. MSE is mean squared error, RMSE is Root Mean Squared Error. In model names FT means FastText vector used is domain trained and Glove denotes we used GloVe word vector; Hie_att indicates our hierarchical encoding and attention model. no_sen_att denotes model with no sentence level encoding and attention, but encoded at essay level. no_att is model without any attention.*

Model	**MSE**	**RMSE**	***P***	***r***	**Kappa (QWK)**	**Kappa (Linear)**
Dimitrios *at el.* [47]	-	2.4	0.91	0.96	0.96	-
Glove_word_2BiLSTM_Hie_att	6.32	2.51	0.89	0.97	0.96	0.81
FT_word_2BiGRU_no_sen_att	3.41	**1.94**	**0.96**	**0.97**	**0.98**	0.87
FT_word_2BiGRU_no_att	4.90	**2.30**	**0.93**	0.96	0.96	0.85
FT_char_word_2BiGRU_ Hie_att	4.20	**2.01**	**0.95**	**0.98**	**0.97**	0.88
FT_char _2BiGRU_Hie_att	4.21	**2.12**	**0.95**	**0.97**	0.96	0.87
FT_word_2BiLSTM_Hie_att	3.99	**2.00**	**0.96**	**0.98**	**0.98**	0.88
FT_word _2BiGRU_Hie_att	3.35	**1.83**	**0.96**	**0.98**	**0.98**	0.88

For essay, we have evaluated our SQM system on different perspective to check performance of our system to predict score on new unseen essay. All experimentations are done using hyper-parameter included under Annex D. We evaluated how each component in our model affect score prediction by passing components individually as shown in *Table 5.7*. First we have evaluated how our FastText vector specific to the domain works well on scoring. We used GloVe word vectors with 840B[16] represented in 300D. The reason for choosing GloVe vectors, is it has less OOV words than Facebook pre-trained FastText vector in our domain. More than 1200 tokens were out-of-vocabulary from Facebook's FastText vectors. But, as compared only 102 words are treated as OOV in GloVe vectors. We can see from the result that using our least performed model that use character vectors of domain trained FastText vectors *FT_char _2BiGRU_Hie_att* only use character level information increased spearman's and Pearson's and quadratic kappa to +4, +1

[16] https://nlp.stanford.edu/projects/glove/

respectively and minimized RMSE to 2.01. Model trained on global vectors *Glove_word_2BiLSTM_Hie_att* is one that performed less in our experimentation from all tests. This shows that our domain trained vectors easily infer meaning of word in an essay than global vectors. Since the essays in the dataset were answers to a specific set of prompts, training the word vectors helped to capture the essence of the words in the domain of the essay prompts thus leading to better performance. Moreover, Kaggle essay has more noise words such with spell errors according to their report and our domain trained word vector easily detected such error.

In addition to vector level evaluation we experimented how hierarchical attention we proposed affect score prediction. From the Table 5.7, all models with suffix '*Hie_att*' indicates hierarchical model. We can see from the above table that both models *FT_word_2BiGRU_no_sen_att* (Only essay level encoding and attention) and *FT_word_2BiGRU_no_att* (without any attention) increased RMSE to 1.94 and 2.3 where as our best performing model with attention has 1.83 RMSE. From this result we can conclude that treating essay as hierarchically organized text allow us to learn essay more than word meanings. Figure 5.5 below shows performance of our regression model by minimizing loss rate per epochs at the time of training.

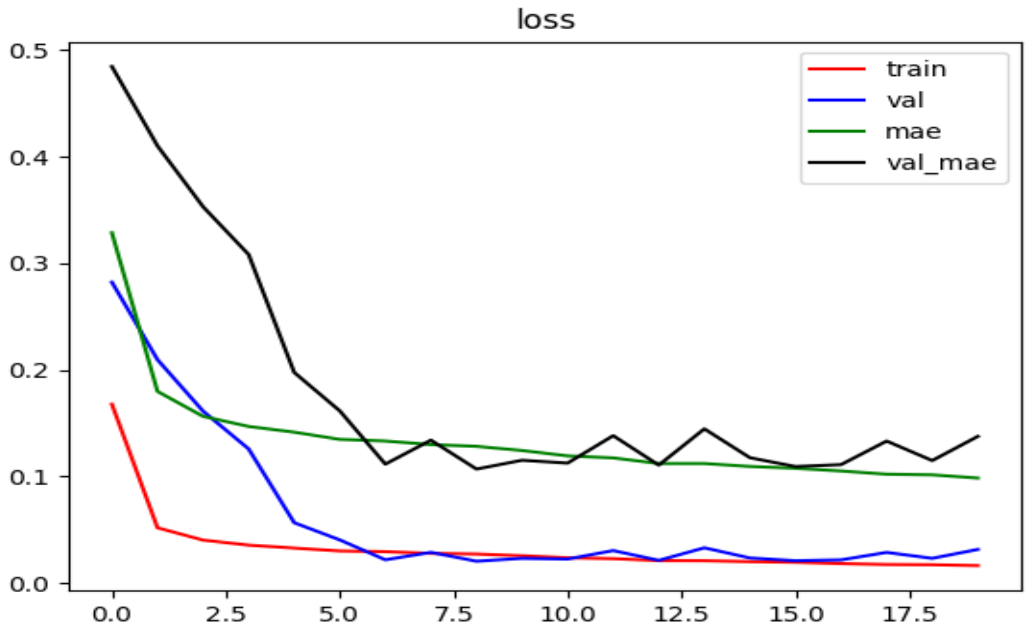


***Figure 5-6**: Loss and mean absolute error rate per epochs on training and validation set for our best performed model on Kaggle Essay dataset. Mae indicates mean absolute error.*

Moreover, we have conducted evaluation on how our character level language model helps on scoring. As expected even if it achieved better result than state-of-art work, it less performed than word level model for English. It is because in English words are not morphologically rich. We tested on different RNN encoders also, as expected both GRU and LSTM performed comparable result. In terms of correlation both shown equal result, but GRU outperformed by minimizing RMSE.

In all approaches tested our hierarchical essay evaluation approach significantly outperformed state-of-art result on Kaggle dataset by increasing performance +6% spearman, +2% Pearson, +2% Cohen's quadratic kappa from state-of-art result. Our best performed models is *FT_word _2BiGRU_Hie_att*, which use word level FastText vectors trained on essay dataset minimized RMSE to 1.83 by increasing +0.57. Given the results of the pre-trained FastText model, we believe that the performance of our best model will further improve should more training data be given to it and further analysis applied on hyper-parameters as deep learning approach highly dependent on parameter setting.

ii. Experiments of short answer

Recall from related word section, we see that short answers are evaluated in two strategies. One is inferring new student answer without having knowledge of reference answer or model answer. And the other is given specific, but not limited correct answer predicting relatedness level of student answer with respect to given model answer. To make our model dynamic to both situations we experimented both approach with and without model answer. Table 5.8 and 5.9 depicts short answer based evaluations experimented on Kaggle short answer dataset and Amharic short answer dataset respectively. Given pair of answers, our SQM short answer assessor predicts score how given student answer is related to provided reference model answer.

Implementation Detail

Unlike, essay we did not treat short answer as hierarchically structured text. For short answers that has no reference or model answer, we encoded given answer using our RNN encoder module and get maximum pooling of the whole answer as representative vector to student answer. Then we align each words in answer with attentive vector to get informative answer words. For answers with reference answer, we applied two level attention one is our *reference attention* that align

words in student answer to representative vector of model answer i.e., GlobalMaximumPooling[17] of model answer words. In both case we pre-processed input answer before passed to embedding layer. The result of preprocessing output is passed to sequence generator sub module that generate sequence by padding and truncating answer to defined threshold. For both Kaggle and Amharic short answer dataset, we used threshold value on which 96%[18] of the dataset shares. We trained FastText model on Kaggle short answer dataset and used as domain specific meaning bearer feature for our SQM model. For Amharic part, we trained all student answer plus course modules on which question sets defined as domain dataset and trained FastText CBOW and Skip-gram model with parameters defined in Table 5.6. We generated character embeddings from word vectors for Amharic part. We did not include character level information to English dataset, as we did not get promising result than word only embeddings for essay experimentation. We used same hyper-parameters with essay to experiment both models except the dropout rate and batch-size[19].

Table 5-8*: SQM short answer result on Kaggle short answer dataset*

Model	**MSE**	**RMSE**	**Spearman**	**Pearson**	**Kappa (QWK)**	**Kappa (Linear)**
Kaggle ASAP best performed						
Luis Tandalla [62]	-	-	-	-	77	-
Our Approach						
FT_word_2BiGRU	0.343	0.586	0.82	0.86	**0.81**	0.68
FT_word_2BiGRU_no_att	0.489	0.613	0.79	0.82	**0.79**	0.64

Discussion on Kaggle Short Answer Dataset

The aim of this experimentation is how our model works for well if we have no reference answer. We conducted two experimentations with or without *reference_attention.* As we can see from

[17] https://keras.io/layers/pooling/

[18] For Kaggle dataset maximum word sequence is 120 and maximum character sequence used is 10. For Amharic dataset we used 83-word length and 7 as character sequence per words. All value is used by inspecting

[19] For Amharic we experimented on 32, 64, and 100 batch-size and we get our best result with 100. The dropout rate used for English is 0.3 and 0.5 for Amharic as we have small data. Using 50% dropout with batch normalization enable us to control overfitting of our model in dataset.

Table 5.8, our model outperformed baseline on Kaggle dataset[20]and achieved encouraging result on short answer. Recently, reported result on short answer in 90% correlation [63] with human annotator using Pearson's metric, but their dataset is not publicly released. Our experimentation shows that applying our model is dynamic and support both condition with and without reference answer. Representing all words in answer equally dropped -0.02% from our best achieved and state-of-art result on Kaggle dataset. From this result we can conclude that our reference attention model can do best if it gets quality data and further analysis is applied to hyper-parameters.

Table 5-9*: SQM short answer result on Amharic short answer dataset. P is spearman and r is Pearson correlation. no_ref_att means model trained without reference attention; no_a2a_att means no answer_to_answer attention; char_embed is for model trained without pre-generated character embeddings but trained with one-hot embedding.*

Model	**MSE**	**RMSE**	***p***	***R***	***Kappa (QWK)***	***Kappa (LWK)***
FT_word_char_BiGRU_ref_a2a	1.35	1.16	**0.60**	0.61	**0.58**	0.41
FT_word_char_BiGRU_no_model	1.39	1.19	**0.56**	**0.56**	0.53	0.35
FT_word_char_BiGRU_model_ref	1.31	**1.14**	**0.62**	**0.62**	**0.59**	**0.42**
FT_word_char_BiGRU_merge	**1.15**	**1.07**	**0.65**	**0.66**	**0.62**	**0.44**
FT_char_embed_BiGRU_no_model	1.36	**1.16**	0.56	**0.57**	0.53	0.36
FT_word_char_BiGRU_no_ref_att	1.55	1.24	0.46	0.47	0.42	0.28
FT_word_char_BiGRU_no_ att_no_model	1.74	1.32	**0.57**	0.54	0.44	0.32
Skip_global_ word_char_BiGRU	**1.21**	**1.15**	**0.58**	0**.60**	**0.57**	**0.40**
Cbow_ global _word_char_BiGRU	**1.20**	**1.14**	0.56	0**.59**	**0.55**	**0.42**
FT_word_BiGRU_merge	**1.80**	**1.12**	0.60	0.61	**0.57**	**0.42**

Discussion on Amharic Short Answer Dataset

As far as our knowledge is concerned, there no system that automatically assess Amharic short answers. We conducted thorough evaluation on our small sized dataset and achieved promising

[20] Kaggle's technical report of winner's shows first winner achieved 77% correlation with human rater.

result that can be taken as baseline work. As opening work, we evaluated our model in various metrics; MSE, RMSE, Spearman, Pearson, Kappa (Quadratic), Kappa (Linear). Our best result achieved 0.65 spearman, 0.66 Pearson, 0.62 quadratic kappa and minimized mean squared error to 1.07.

We have conducted evaluation on our Amharic short answer marking system component wise. We valued how our character model works in Amharic as Amharic is one of morphologically rich language and as expected the model that contain domain trained FastText vector with concatenation to our character CNN that use generated character vector. All models except the one in last row (*FT_word_BiGRU_merge)*, use character word information. We evaluated our best performed model by skipping character model and as expected it drops -0.05 from best performed model Kappa (QWK) correlation and increase error rate to 1.12. But, it still shows competitive result it's because of our FastText has character n-gram information. Our character based model that use one-hot encoded input and later represent word vectors character from our char-CNN (*FT_char_embed_BiGRU_no_model*) perform better than the one that use pre-trained FastText character embeddings (*FT_char_word_BiGRU_no_model*). It is because the former represent characters by based on training dataset whereas the later use global information of character. But, in terms of Kappa both performed equally.

We also conduct how our *answer-to-answer* and *reference attention* affect score prediction. We can see from *Table 5.9* that all models with attention (*FT_word_char_BiGRU_ref_a2a (*with both attention*), FT_word_char_BiGRU_model_ref* (with reference attention only*))*, have promising result. Even if the data is limited, both attention helps reach better scoring by clueing the network. From result we may think that if we have enough data our attention techniques better clue the score prediction. Because of the data size the result with answer-to-answer achieved less than *reference only* attention. It is because as model become complex and data not able fit and is overfitting the model.

Based on result obtained by evaluating effect of word vectors, both domain trained[21]and global vectors achieved promising result in terms of correlation and RMSE. In our experimentation, model that use Skip-gram *(Skip_global_word_char_BiGRU*) slightly shown better than CBOW

[21] All tests except Cbow_ global _word_char_BiGRU and Skip_ global _word_char_BiGRU used domain trained vectors.

(*Cbow_ global_word_char_BiGRU).* This shows ability of word vectors in representing word meaning. Moreover, we can conclude that using domain vector are good for subjective question assessment task than representing words in global domain.

We also evaluated the effect of using model answer, the result shown that correct reference answer helps to get better prediction. The model named (*FT_word_char_BiGRU_no_ref_att*) without model and also self-reference attention downs the result by increasing RMSE to 1.32. *FT_word_char_BiGRU_no_ att_no_model* model tested without model answer and no attention increased again RMSE to 1.74 even if it does well in terms of correlation. From our experimentation all best performed model use model answer as reference correct answer (see row 1, 3, 4 from Table 5.9). From this result we can conclude that our reference attention model rely on model answer can do best if given quality data and further analysis is applied to hyper-parameters.

In general, SQM experimentation shown that our deep learning approach can further improve the result with best working parameters.

CHAPTER SIX

6. CONCLUSION AND RECOMMENDATION

6.1 Conclusion

Evaluation of the students is a crucial issue in the teaching-learning process especially open questions are considered to be the most appropriate because they help to evaluate the understanding of ideas, the students' ability to organize material and to evaluate the originality of the thoughts. However, scoring subjective questions manually is challenging task for instructors. As the result objective question which is not suitable to evaluate skill of student is taken as a de facto question type used to assess student performance.

We designed subjective question marking system called SQM capable of assessing both short answer and essays questions automatically. SQM has five main components named pre-processing, word representation, encoding, attention and scoring. The pre-processing module normalize pre-graded student answer and provided to word representation also called embedding module. Based on output of pre-processing module primarily embedding module generate integer sequence by transposing word to their index. Then each word indices are replaced to FastText word and character vector that has meaning bearer units of the word and sequence of vectors are returned to answer or sentence representation module. We used two different word vectors; one that is trained on domain dataset that is specific to the question seen at the time of training and the other is global word vector. We trained our model using both global and domain FastText vectors for Amharic dataset and only train domain word vectors for English dataset. In addition to FastText word vectors we generate character vectors by averaging word vector of all words in vocabulary in which the character exists and incorporate to word vector as word sub-information. Using FastText vectors allowed our model to treat rare words based on their relevancy level. In addition because of concatenated character level word context, our model encoded out-of-vocabulary words based on their context. Moreover, with FastText ability to infer word vector using their shared character n-grams, our model considered words with spelling error to their meaningful words.

The result of word representation layer is passed to sequence encoders. At this layer we applied two task dependent encoders. To deal with coherency in essay, we first encode each sentence in essay to get sentence vectors and using each sentence vectors we encoded high level essay context.

At each phase of encoders in essay instead of passing all information about words in essay we applied sentence level and essay level attention that allows us to get most informative essay vectors only. By doing so, we shown dealing only with salient information in text allow us to get more answer context than treating each words equally relevant. To deal with short answer questions using deep learning model, we introduced new model that apply answer level encoder to get context of all words (vectors) in answer to fixed length low-dimensional space using bidirectional RNN. Both variants of RNN; GRU and LSTM are experimented. Our model works for both answer with model answer and without model answer. For answers with model answer we build attentive vector from model answer and aligned each words of student answer in to attentive vector. Then model answer aware student answer vector and model answer are matched using answer-to-answer attention. i.e., from model answer words to student answer and student answer words to model answer. Then contextual information of both side attention output is encoded with bidirectional RNN to model interaction between two vectors. Finally, Softmax linear regression is used to predict score based on range specified by question set.

We evaluated our SQM model component wise and shown that this kind of architecture is able to suppress systems developed using knowledge based approach as well as system that depend on manual feature engineering. Without having prior knowledge about grammar and any handcrafted features our model performed very human like way and outperformed all state-of-art works on Kaggle dataset by achieving 98% quadratic Kappa on essay dataset and 81% quadratic Kappa on Kaggle short answer dataset.

Our Amharic short answer model evaluation shown that our Amharic SQM system is the first Amharic short answer marking system that shows promising result on small sized dataset as compared to resourced languages. The best performed Amharic short answer assessing model achieved correlation Pearson, Spearman and Kappa as 66%, 65% and 62% respectively to human graded answer and minimized root mean squared error to 1.07. This shows that if we pass enough data with pre-trained model we can, it can score unseen subjective question from any domain as near exact correlation to human raters.

6.2 Contribution of the Study

The main contribution of this thesis works are:

- ✓ The study identifies architecture used in developing neural network based approach for subjective question marking
- ✓ We introduced hierarchical encoding and attention method to assess essay that can be taken as framework for education sector
- ✓ We made known deep learning based attentive neural model that can assess subjective questions from any domain without expecting domain dependent features.
- ✓ The Amharic dataset created by two raters for the purpose of evaluating our system performance for Amharic can be used for successive works on this area
- ✓ We show ability of FastText word vectors performance in subjective assessment domain as written answers are susceptible to spelling error
- ✓ Design and develop model requiring no appeal to natural language specific process beyond tokenization and simple normalization at character level
- ✓ We developed FastText word vectors that can be used with any NLP application as external knowledge by inferring word meaning. Our vectors are skillful on representing words with syntactic difference and can be used as tool to replace morphology analyzers and contribute on filling the gap on fundamental NLP tools
- ✓ The study clearly shown that when and how to use of sequence encoders in deep learning such GRU and LSTM for assessment task
- ✓ The study shown using the value of incorporating character level language model with concatenation to word vector to represent words by incorporating sub-word information and minimize out-of-vocabulary words
- ✓ The study shown that short answer questions can be assessed with and without model (correct reference) answer. But one with reference answer is best choice.

6.3 Recommendation and Future Work

The following enhancements are recommended for SQM.

- ✓ Deep learning models are transferable to best perform in this area quality dataset is has major relevance so preparing enough quality data is recommended
- ✓ Our work not considered subjective question with formulas and figures. To make SQM complete is recommended to analysis such question and incorporate to SQM.
- ✓ With recent advent in deep learning, we can visualize network hidden layer behavior in human understandable way. To make SQM applicable to educational sector we recommend to incorporate feedback that is specific and instructional to missed points made by certain student.
- ✓ FastText word vectors can be used as background knowledge for today's NLP application such as question answering, sentiment analysis, textual entailment etc. Evaluating performance of our FastText in such application is recommended.

References

[1] S. J. Hussain. "Validity and Credibility of Public Examinations in Pakistan" *Unpublished Ph.D. Thesis, Department of Education, Islamic University Bahawalpur, Pakistan,* 2002.

[2] B. Jill, K. Kukich, S. Wolff, C. Lu, M. Chodorow, L. Braden-Harder, and M. D. Harris, "Automated Scoring Using a Hybrid Feature Identification Technique", in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics,* Vol.1, pp. 206-210, 1998.

[3] B. Yigal, A. Jill, "Automated Essay Scoring with E-rater", *Journal of Technology Learning and Assessment,* Vol.4, No. 3, 2006.

[4] F. Peter, W. D. Laham, T. K. Landauer, "Automated Essay Scoring: Applications to Educational Technology", *in World Conference on Educational Multimedia, Hypermedia and Telecommunications*, Vol. 1, pp. 939-944, 1999.

[5] I. Tsunenori, M. Kameda, "Automated Japanese Essay Scoring System Based on Articles Written by Experts", *in Proceedings of the 21st International Conference on Computational Linguistics*, pp. 233-240, 2006.

[6] C. Tao-Hsing, C.-H. Lee, Y.-M. Chang, "Enhancing Automatic Chinese Essay Scoring System from Figures-of-Speech" *in PACLIC,* 2006.

[7] National Educational Assessment and Examination Agency, "Universities 2009 E.C Intake Capacity", retrieved from www.nae.gov.et/5/neaea_download_refereces, last accessed on October 20, 2016.

[8] "Harmonized Modular Curriculum: Ethiopian Language(s) & Literature – Amharic program", retrieved from http://www.kuc.edu.et/images/pdf/ETH.LL.pdf, last accessed on October 20, 2016.

[9] M.L. Bender, J. D. Bowen, C. R. Cooper, C. Ferguson, "Government Language Policy: Language in Ethiopia" *Oxford University Press,* 1976.

[10] L. Thomas, J. Psotka, "Simulating Text Understanding for Educational Applications with Latent Semantic Analysis: Introduction to LSA" *in proceeding of Interactive Learning Environments*, Vol. 8, No. 2, pp. 73-86, 2000.

[11] C. O'Farrell. "Enhancing Student Learning Through Assessment: A Toolkit Approach." *Centre for Academic Practice and Student Learning, Dublin*, 2004.

[12] E. Lorna, *Assessment as Learning: Using Classroom Assessment to Maximize Student Learning*, Corwin Press, 2003.

[13] P. Weeden, J. Winter, "Assessment what is not for School", *Routledge Falmer 270 Madison*, New York, USA and Canada, 2002.

[14] L. Iasonas, J. A. Athanasou, *A Teacher's Guide to Educational Assessment*, sense publisher 2009.

[15] Abel Teklemariam, "Automatic Amharic Essay Scoring System Using Latent Semantic Analysis", Unpublished Master Thesis, Department of Computer Science, Addis Ababa, 2010.

[16] M. Jang, J-C. Sohn, H. K. Cho, "Automated Question Answering using Semantic Web Services", *IEEE Asia-Pacific Services Computing Conference*, 2007.

[17] M. Syamala, and H. Mittal. "Review of Computerized Evaluation Tools in Education." *IJAICR*, Vol. 4, No. 2, pp. 111-117, 2012.

[18] S. Dikli, "An Overview of Automated Scoring of Essays." *Journal of Technology, Learning, and Assessment*, Vol.5, No.1, 2006.

[19] E. Batten, "Computer Grading of Student prose, using Modern Concepts and Software", *The Journal of experimental education*, Vol. 62, No. 2, pp.127-142, 1994.

[20] M. Shermis, M. Howard, J. Olson, S. Harrington, "On-line Grading of Student Essays: PEG Goes on the World Wide Web", Assessment *& Evaluation in Higher Education*, Vol. 26, No. 3, 247-259, 2001.

[21] J. Sukkarieh, S. Svetlana, "Automating Model Building in C-Rater" In *Proceedings of the 2009 Workshop on Applied Textual Inference*, pp. 61-69, 2009.

[22] R. Philip, "Using Information Content to Evaluate Semantic Similarity in A Taxonomy", *arXiv preprint cmp-lg/9511007,* 1995.

[23] H. Gomaa, A. Fahmy "A Survey of Text Similarity Approaches", *International Journal of Computer Applications,* Vol. 68, No.13, pp. 0975 – 8887, 2013.

[24] D. Lin, "Extracting Collocations from Text Corpora", *In Workshop on Computational Terminology,* Montreal, Canada, pp. 57–63, 1998.

[25] S. Boyce, C. Pahl, "Developing Domain Ontologies for Course Content." *Educational Technology & Society,* Vol. 10 No.3, pp.275-288, 2007.

[26] V. Senthil, A. Sankar, "Towards an Automated System For Short-Answer Assessment Using Ontology Mapping", *International Arab Journal of e-Technology,* Vol. 4 No. 1, 2015.

[27] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson, "One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling", in *arXiv preprint*, 2013.

[28] A. Conneau, H. Schwenk, Y. L. Cun, L. Barrault, "Very Deep Convolutional Networks for Text Classification" *arXiv: 1606.01781v2*, 2017.

[29] J. Pennington, R. Socher, D. Manning, "GloVe: Global Vectors for Word Representation", Empirical *Methods in Natural Language Processing*, pp. 1532-1543, 2014

[30] R. Socher, "Recursive Deep Learning for Natural Language Processing and Computer Vision" *Unpublished Ph.D. Thesis,* Stanford University, 2014.

[31] T. Mikolov, I. Sutskever, K. Chen, S. Corrado, J. Dean, "Distributed Representations Of Words And Phrases And Their Compositionality", *in Advances in Neural Information Processing Systems*, pp. 3111–3119, 2013.

[32] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jegou & T. Mikolov "Fasttext.zip: Compressing Text Classification Models", *arXiv: 1612.03651v1*, 2016.

[33] P. Ofir, L. Wolf, "Using the Output Embedding to Improve Language Models", in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*: Vol. 2, pp.157-163, 2016.

[34] I. Sutskever, O. Vinyals, V. Le, "Sequence to Sequence Learning with Neural Networks", in *Advances in neural information processing systems*, pp. 3104-3112, 2014.

[35] H. Sepp, J. Schmidhuber, "Long Short-Term Memory", *Neural computation,* Vol. 9 No.8, pp. 1735-1780, 1997.

[36] C. Junyoung, G. Caglar; C. KyungHyun, Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling" *arXiv:* 1412.3555, 2014.

[37] K. Cho, B. Merrienboer, D. Bahdanau, Y. Bengio, "On The Properties of Neural Machine Translation: Encoder-Decoder Approaches." *arXiv preprint arXiv: 1409.1259*, 2014.

[38] M. Shermis, J. Burstein, "Automated Essay Scoring: A Cross Disciplinary Derspective*", Routledge,* 2003.

[39] F. Peter, "Latent Semantic Analysis for Text-based Research." *Behavior Research Methods,* Vol. 28, No. 2 pp. 197-202, 1996.

[40] L. Yuhua, M. David, A. Bandar, D. O'Shea, and K. Crocketthe "Sentence Similarity Based on Semantic Nets and Corpus Statistics*", IEEE transactions on knowledge and data engineering*", Vol. 18 No. 8, 2006.

[41] J. Burstein, M. Chodorow, C. Leacock, "Criterion SM Online Essay Evaluation: An Application for Automated Evaluation of Student Essays", in *proceedings of the fifteenth annual conference on innovative applications of artificial intelligence,* Acapulco, Mexico, 2003.

[42] J. Lani "Statistics Solution Advancement Through Clarity," retrieved from http://www.statisticssolutions.com/pearsons-correlation-coefficient, last accessed on June 17 2017.

[43] G. Kendall, *Rank Correlation Methods,* Oxford University Press, 1990.

[44] S.Burrows, I. Gurevych, B. Stein "The Eras and Trends of Automatic Short Answer Grading", *International Journal of Artificial Intelligence in Education IOS Press,* Vol.25, pp.60 – 117, 2015.

[45] G. Wael, F. Aly, "Short Answer Grading Using String Similarity and Corpus-Based Similarity" *International Journal of advanced Computer Science and Applications (IJACSA),* Vol. 3, No. 11, pp. 115-121, 2012.

[46] R. Shourya, S. Himanshu, Y. Narahari, "An Iterative Transfer Learning Based Ensemble Technique for Automatic Short Answer Grading", *arXiv: 1609.04909v2 [cs.CL],* 2016.

[47] A. Dimitrios, Y. Helen, R. Marek, "Automatic Text Scoring Using Neural Networks", *arXiv: 1606.04289v2 [cs.CL],* 2016.

[48] Y. Kim, Y. Jernite, D. Sontag, M. Rush, "Character-Aware Neural Language Models", *arXiv: 1508.06615,* 2015.

[49] F. Gutierrez, D.Dou, S. Fickas, "Providing Grades and Feedback for Student Summaries by Ontology-based Information Extraction", *ACM,* 2012.

[50] K. Rajiv, R. Ramesh, *A Handbook of Principles, Concepts and Applications in Information Systems,* Oxford University Press, 2007.

[51] M. Mohler, R. Mihalcea, "Text-to-text Semantic Similarity for Automatic Short Answer Grading", in *Proceedings of the 12th Conference of the European Chapter of the ACL,* Athens, Greece, pp. 567-575, 2009.

[52] Y. Goldberg, "A Primer on Neural Network Models for Natural Language Processing" *arXiv: 1510.00726v1 [cs.CL],* 2015.

[53] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, E. Hovy, "Hierarchical Attention Networks for Document Classification" In *HLT-NAACL,* pp. 1480-1489, 2016.

[54] M. Schuster, K. Paliwal, "Bidirectional Recurrent Neural Networks" *IEEE Transactions On Signal Processing,* Vol. 45, NO. 11, pp. 2673-2681, 1997

[55] S. Arora, Y. Liang, M. Tengyu, "A Simple but Tough-to-beat Baseline for Sentence Embeddings", *ICLR,* 2017

[56] Sebastian Ruder "Deep Learning for NLP Best Practice" retrieved from http://ruder.io/deep-learning-nlp-best-practices/index.html#attention, last accessed on April 08, 2017.

[57] S. Minjoon, K. Aniruddha, F. Ali, H. Hannaneh, "Bidirectional Attention Flow for Machine Comprehension", arXiv: 1611.01603 *[cs.CL],* 2017.

[58] M. Syamala, H. Mittal, "Machine Learning Techniques with Ontology for Subjective Answer Evaluation*", International Journal on Natural Language Computing (IJNLC),* Vol. 5, No.2, pp. 1-11, 2016.

[59] C. Gulcehre, "Deep Learning" retrieved from http://deeplearning.net/software_links/, Last accessed on June, 11 2017.

[60] Y. Wenpeng, K. Katharina, Y. Mo, S. Hinrich, "Comparative Study of CNN and RNN for Natural Language Processing", *arXiv:1702.01923v1,* 2017

[61] X. Zhang, J. Zhao, Y. Lecun, "Character-level Convolutional Networks for Text Classification", in *Advances in Neural Information Processing Systems* pp. 649-657, 2015.

[62] L. Tandalla, "Scoring Short Answer Essays", retrieved from https://kaggle2.blob.core.windows.net/competitions/kaggle/2959/media/TechnicalMethodsPaper.pdf, last accessed on May, 09 2017.

[63] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-Based Learning Applied to Document Recognition." *Proceedings of the IEEE,* Vol.86 No. 11, pp. 2278–2324, 1998.

[64] L. Maas, Y. Hannun, Y. Ng, "Rectified Linear Units Improve Neural Network Acoustic Models*", in Proceedings of the 30-th International Conference on Machine Learning*, Atlanta, Georgia, USA, Vol. 28, 2013.

[65] J. Elman, "Finding Structure in Time", *Cognitive Science,* Vol.14, No. 2, pp.179–211, 1990.

[66] T. Mikolov, "Statistical Language Models Based on Neural Networks." *Unpublished Ph.D. thesis*, Brno University of Technology, 2012.

Annexes

Annex A: Amharic Homonym Characters

Homophone characters	Characters to replace
ሃ, ኅ, ኃ, ሐ, ሓ, ኻ	ሀ
ሑ, ኁ, ኹ	ሁ
ኒ, ሒ, ኺ	ሂ
ኌ, ሔ, ኼ	ሄ
ሕ, ኅ	ህ
ሖ, ሗ, ኾ	ሆ
ሠ	ሰ
ሥ	ስ
ሡ	ሱ
ሢ	ሲ
ሣ	ሳ
ሤ	ሴ
ሦ	ስ
ዓ, ኣ, ዐ	አ
ዑ	ኡ
ዒ	ኢ
ዔ	ኤ
ዕ	እ
ዖ	ኦ
ጸ	ፀ
ጹ	ፁ
ጺ	ፂ
ጻ	ፃ
ጼ	ፄ
ጽ	ፅ
ጾ	ፆ
ቀ	ቁ
ኵ	ኩ

Annex B: Amharic Labialized characters used interchangeably in writing

Form 1	Form 2	Labialized form
ሙዋ	ሙአ	ሟ
ቱዋ	ቱአ	ቷ

Form 1	Form 2	Labialized form
ሩዋ	ሩአ	ሯ
ሱዋ	ሱአ	ኗ
ሹዋ	ሹአ	ኟ
ቁዋ	ቁአ	ቋ
ቡዋ	ቡአ	ቧ
ቹዋ	ቹአ	ቿ
ሁዋ	ሁአ	ኋ
ኑዋ	ኑአ	ኗ
ኙዋ	ኙአ	ኟ
ኩዋ	ኩአ	ኳ
ዙዋ	ዙአ	ዟ
ጉዋ	ጉአ	ጓ
ዱዋ	ዱአ	ዷ
ጡዋ	ጡአ	ጧ
ጩዋ	ጩአ	ጯ
ጱዋ	ጱአ	ጿ
ፉዋ	ፉአ	ፏ

Annex C: Common Short forms to their expanded form in Amharic

Short Form	Expanded Form	Short Form	Expanded Form
ት/ቤት	ትምህርት ቤት	ዶ/ር	ዶክተር
መ/ር	መምህር	ሚ/ር	ሚኒስትር
ት/ክፍል	ትምህርት ክፍል	ተ/ሃይማኖት	ተክለ ሃይማኖት
ሃ/አለቃ	ሀምሳ አለቃ	ጠ/ሚኒስትር	ጠቅላይ ሚኒስትር
ሃ/ስላሴ	ሀይለ ስላሴ	ኮ/ል	ኮለኔል
ደ/ዘይት	ደብረ ዘይት	ሜ/ጀነራል	ሜጀር ጀነራል
ደ/ታቦር	ደብረ ታቦር	ብ/ጀነራል	ብርጋዳር ጀነራል
መ/ቤት	መስሪያ ቤት	ሌ/ኮለኔል	ሌተናሌ ኮለኔሌ
ጽ/ቤት	ጽህፈት ቤት	ሊ/መንበር	ሊቀ መንበር
ክ/ከተማ	ክፍለ ከተማ	ር/መምህር	ርእሰ መምህር
ክ/ሀገር	ክፍለ ሀገር	አ/አ	አዲስ አበባ
ወ/ር	ወታደር	ፕ/ት	ፕሬዝዳንት
ወ/ሮ	ወይዘሮ	ዓ.ም	አመተ ምህረት
ወ/ሪት	ወይዘሪት	ዓ.ዓ	ዓመተ ዓለም
ወ/ስላሴ	ወሌተ ስላሴ	ዶ.ር	ዶክተር
ቤ/ክርስትያን	ቤተ ክርስትያን	ፕ/ር	ፕሮፌሰር
ቤ/ያን	ቤተ ክርስትያን	እ.አ.አ	እንደ አዉሮፓዊያን አቆጣጠር

Short Form	Expanded Form	Short Form	Expanded Form
ም/ቤት	ምክር ቤት	ሰ/ት/ቤት	ሰንበት ትምህርት ቤት
ፍ/ስላሴ	ፍቅረ ስላሴ	ታ/የተ/የግ	ታላፊነቱ የተወሰነ የግል
ፍ/ቤት	ፍርድ ቤት	ሲ/ር	ሲስተር

Annex D: Experimental Hyper-parameters used to train SQM models

Parameter Name	Value Experimented for Essay	Value Experimented for Kaggle short answer	Value Experimented for Amharic short answer
RNN			
RNN Size	100	100	100
RNN	GRU/LSTM	GRU/LSTM	GRU/LSTM
Merge Mode	Concat	Concat	Concat
Number of Layer	2	2	1
Highway Layer	2	2	1
CNN			
Kernels	7,3,4,5	2,3,4,5	2,2,3
Pooling	MaxPooling	MaxPooling	MaxPooling
Feature maps	16,32,32,32	16,32,32,64	16,32,32
Padding	Valid	Valid	Valid
Training Parameters			
batch size	64	64	32
Epochs	20	100	100
Dropout	0.2	0.5	0.5
BatchNormalization	True	True	True
Activation	ReLu	ReLu	ReLu
Optimizer	Adam	Adam	Adam
Seed	3435	1024	1024
character length per word	10	10	7
Word length per sentence	40	120	83
Sentence per essay	35	-	-

Annex E: Sample Questions and answers with score assigned by two raters

ጅማ ዩኒቨርሲቲ

በሶሻል ሳይንስ እና ሂዩማኒቲስ ኮሌጅ

የአማርኛ ቋንቋ እና ስነጽሁፍ ትምህርት ክፍል

የህዝብ ግንኙነት መግቢያ አጠቃላይ ፈተና

ቀን: ነሐሴ 26፣ 2009 ዓ.ም

የተሰጠው አጠቃላይ ሰዓት 2:00

የተማሪው ስም: _____

መታወቂያ ቁጥር: _____

<u>ማስጠንቀቂያ</u>

ተማሪዎች የመታወቂያ ቁጥር በመልስ መስጫ ወረቀት ላይ ደግማቹ ጻፉ። የፈተና ወረቀቱ ዘጠኝ ገጾች መያዙን አረጋግጡ።ከተቀመጠው መልስ መስጫ ቦታ ውጪ መጠቀም አይቻልም።

መልካም ፈተና!

ትዕዛዝ ሶስት፡- ለሚከተሉት ጥያቄዎች ግልጽና የተብራራ መልስ ስጪ/ጥ (26 ነጥብ)

1. በቀውስ ወቅት የሚኖር ተግባቦት ሶስት መሰረታዊ ግቦችን ለማሳካት አልሞ መካሄድ እንደሚገባው ይታወቃል። እነኝህን ሶስት መሰረታዊ ግቦች በቅደም ተከተል ዘርዝሪያቸው/ራቸው (3 ነጥብ)

   ✓ መልስ : ቀውሱን በፍጥነት ማስቆም፤ ቀውሱን ያስከተለውን ጉዳይ በአፋጣኝ መቀነስና የመንግስት ወይም የተቋመን ተአማኒነት መልሶ መገንባት ናቸው።

2. ሚዲያ ለህዝብ ግንኙነት ሙያ/ስራ ያለውን ጠቀሜታ አስረጂ/ዳ (4 ነጥብ)።

- ✓ መልስ : ሚድያ ለህዝብ ለህዝብ ግንኙነት የሚያበረክተው ጠቀሜታ ብዙ ነው የተቋሙን የድርጅቱን እንቅስቃሴ መልእክት ለማስተላለፍ ያስችላል። በቀውስ ወቅት የተቋሙን ደህንነት መልስ ለመገንባት ስለተቋሙ የሚወሩ አሉባልታዎችን ለማወቅና ለማስወገድ የሚቻለው በሚድያ ነው። የተቋሙን ጊዴታ መልሶ ለመገንባት የሚቻለው በሚዲያ አማካይነት ነው። ለተቋሙ ለአገር ግንባታ የሚያስፈልጉ ገንዘብ ለማሰባባሰብ ያግዛል። ህዝብን ተደራሽን ለማሳመን ይጠቅማል። የህዝብ ግንኙነት ባለሙያ ያዘጋጀውን ዕቅድ ወደተግባር ለመቀየር ያስችላል። ምርትና አገልግሎትን ለማስተዋወቅ።

3. የህዝብ ግንኙነት ሙያተኞች ነፃ የመሆን ስነ ምግባር ሊኖራቸው ይገባል ሲባል ምን ማለት ነው በሚገባ ግለጪ/ጽ (4 ነጥብ)

- ✓ መልስ : የህዝብ ግንኙነት ባለሙያ መቸም ቢሆን ተጨባጩን እውነት ለማንም ሲባል ማጣመም ስለሚያስፈልግ ነው። የህዝብ ግንኙነት ስራ በተጨባጭ እውነት ላይ ካልተመሰረተ ጥፋት ማስከተሉ ስለሚያቀርበው የህዝብ ግንኙነት ባለሙያ ነፃ ካልሆነ ሥራውን በሚገባ ሊያከናውን አይችልም።

4. ከእቅድ ዝግጅት ዋና ዋና ተግባራት አንዱ የሁኔታዎች ግምገማ ማካሄድ መሆ ይታወቃል። ከዚህ አንፃር የሁኔታ ግምገማ ለማካለሄድ የሚያስችሉ ስልቶችን በመጠቆም ስለምንነታቸው አጠር አጠር ያለ ማብራሪያ አቅርቢ/ብ (4 ነጥብ)

- ✓ መልስ : በመጀመርያ የሁኔታቸው ግምገማ በጠራ መረጃ ላይ መመስረት አለበት ያለውን አመለካካት ለማወቅ ያስችላል። ጥናትና ምርምር ማካሄድ በታሰበው ጉዳይ ላይ ተደራሹ ማህበረሰብ ያለውን አመለካከት ለማወቅ ያስችላል የናሙና ጥናት ማካሄድ የናሙና ጥናት የተደረገላቸውን ማህበረሰብ አስተያየት ለማወቅ ያስችላል ናሙና በተወሰኑ መጠይቆች አማካነት የሚዘጋጅ ነው። ግብር መልክ በአንድ ጥናት በተላለፈ መልእክት ላይ ተደራሹ ማኅበረሰብ የሚፈጥረውን ግንዛቤና የሚያሳየውን አዝማሚያ መነሻ በማድረግ የሚሰበሰብና ለመልእክት አመንጩ የሚተላለፍ ምላሽ ነው።

5. በፕሬስ መግለጫ በዜና መግለጫ እና በፕሬስ ኪት መካከል ያለውን ተመሳስሎና ልዩነት አብራሪ/ራ (3 ነጥብ)

- ✓ መልስ ፡ ተመሳስሎ ሁሉም መልእትን ለማስተላላፍ ያግዛሉ። ወደ ተደራሽ ማህበረሰብ ለማቅረብ አይነተኛ መሳርያዎች ናቸው። ልዩነት ፡ የፕሬስ መግለጫ ለቆዩ ጉዳዮች ላይና ህዝቡ ሊያውቀው በሚገባ ጉዳይ ላይ ይዘጋጃል። የዜና መግለጫ ለትኩስና ባልቆዩ ጉዳዩች ላይ ይመሰረታል። የፕሬስ ኪት የህዝብ ግንኙነት ሙያተኛው የሚወክለውን ተቋም ወይም አገር ለማስተዋወቅ አስፈላጊ መረጃዎችን አሰባስቦ ለሚድያው ለተደራሹ ማህበረሰብ የሚቀርቡበት ጥራዝ ነው።

6. የህዝብ ግንኙነት ሙያተኛ ራሱን በተለያዩ ሚዲያዎች ካስተዋወቀ በኋላ ውጤቱን መመዘን እንዳለበት ይታመናል። ከዚህ አንፃር አንድ የህዝብ ግንኙነት ሙያተኛ የማስተዋወቁን ውጤት የሚለካባቸውን አስተማማኝ መንገዶች በመጠቆም ስለምንነታቸው ማብራሪያ አቅርቢ/ብ (4 ነጥብ)።

- ✓ መልስ ፡ አገሪቱ ተቋሙ በሚድያው ባገኘው ሽፋን መጠንና አይነት በተደራሹ ማህበረሰብ ዘንድ በተፈጠሩ አስተያየቶች እና ይህንን ተከትሎ በተወሰዱ ተግባራዊ እርምጃዎች በመመዘን የሚያረጋግጥ ነው። ይህንን ጉዳዩ በሶስተኛ ወገን በማስጠናት ማረጋገጥም ይቻላል ሶስተኛ ወገኑ ፕሮፌሽናል ወይም ለራስ ባለሙያም ማስጠናት ይቻላል።

7. በመልዕክቱና በቋንቋው ጥራት ታዳሚን የሚያማልልና ለተግባር የሚያነሳሳ ጥሩ የአደባባይ ወይም የንግግር ጽሁፍ ሲዘጋጅ የምትከተያቸውን/ላቸውን አራት ዋና ዋና ጉዳዮች ወይም አካሄዶች በመጠቆም አብራሪ/ራ (4 ነጥብ)።

- ✓ መልስ ፡ የአደባባይ የንግግር ጽሁፍ ለማካሄድ ለጽሑፍ መዘጋጀት ዝግጅት ማድረግ ይገባል። ዝግጅቱን ማን ምን የትና መቼ የሚሉት ጉዳዮች ሊመለሱ ይገባል ። ቃለ መጠይቅ ማድረግ፣ ከተናጋሪው ጋር የሚደረግ አጭር ቃለ ምልልስ ማደረግ፣ የሚዘጋጀውን ጽሁፍ በጥናት የማረጋገጥ ከፍተኛ ሂደት ያጎናጽፋል። ምርምር ጥናት ማካሄድ፤ በጉዳዩ ላይ ቀደም ተብሎ የተሠራ መጽሀፍ ጽሁፍ ሊፈተሽ ይገባል። በጉዳዩ ላይ የተሻለ እውቀት ያላቸውን ሰዎችም ማረጋገጥ ይገባል። ሃሳብ አደረጃጀት መፃፍ ከዚህ በኋላ ሃሳብን በማደራጀት መፃፍ ይገባል። በዚህ ጊዜ ጽሁፉ ጥሩ መግቢያ መሪ ሃሳብ ዋና አካል እና ጥሩ ማጠቃለያ ሊቀርብ ይገባል።

No	Student Answer	Score_1	Score_2
1	የቀውሱን መንስኤ ወይም ምክንያት ለማወቅ ለተከሰተው ቀውስ መፍሔ ለመፈለግ መንግስትን (ተቋምን) ከገባበት ቀውስ እንዲወጣ ለተደራሽ ለማሳወቅ	3	3
2	ሚዲያ ለህዝብ ግንኙነት ሙያ እጅግ የጎላ ጠቀሜታ አለው። ጠቀሜታውም በህዝብ ግንኙነት ሙያ ውስጥ ያሉ መረጃዎ፣ የተገኙ ጠቃሚ የሆኑና ለተደራሽ አስፈላጊ የሆኑ ነገሮ በሙሉ ለተደራሹ የሚደርሱት ሚዲያን በመጠቀም ነው። ስለሆነም ሚዲያ ለህዝብ ግንኙነት ሙያ አስፈለጊና ጠቃሚ ነው።	3	4
3	የህዝብ ግንኙነት ሙያተኛ ሊከተላቸው ከሚገቡት ስነምግባሮች አንዱ ነፃ መሆን 3 ነው። የህዝብ ግንኙነት ሙያተኛ ነፃ መሆን አለበት ሲባልም የትኛውንም አቅጣጫ ሳይዝ ማትም ተቋሙን ብቻ ወይም ተደራሹን ብቻ ሳይወግን ከሁለቱም ነፃ ሆኖ ፌትሐዊ በሆነ መንገድ መረጃ ማስተላለፍ ስላለበት ነው።	3	2
4	ግምገማ ለማስሄድ የሚያስችሉ ስልቶች፡- 1- ምርምር ማካሄድ፡- ዕቅድ ከመዘጋጀቱ በፊት ችግር ናቸው የሚባሉትን መለየትና በነዛ ጉዳዮች ላይ ጥናትና ምርምር ማካሄድ አለበት። 2- የናሙና ጥናት ማካድ፡- ይህ ሲባል የተወሰኑ ቡድኖችን ናሙና በመውሰድ የችግሩን አቅጣጫ የሚለይበት ነው። 3- ግብረ መልስ መውሰድ፡- ይህ ሲባል ደግሞ በተነሱት ችግሮ ዙሪያ የጠሰጡ ግብረ መልሶችን በመሰብሰብ ዕቅዱን የሚያዘጋጅበት ሂደት (ስልት ነው) ።	4	4
5	የፕሬስ መግለጫ ፡- የሚባለው ታዋቂነትና ቀልብ ሳቢነት ያለው ሆኖ ቀድሞ የተላለፈውም ቢሆን የህዝብ ፍላጎት ካለ ተደግሞ የሚቀርብ መግለጫ ነው። ዜና መግለጫ የሚባለው ደግሞ አዲስነት፣ ግጭት፣ ወቅታዊነት፣... ጉዳዮችን የያዘ ሆኖ በትኩስና ወቅታዊ ነገሮች ላይ የተመሰረተ ነው። ፕሬስ ኪት የሚባለው ደግሞ የተቋሙ መረጃዎች ማለትም ፎቶ ግራፎች፣ ፋስት ሺቶች፣ የዜና መግለጫዎችን፣ የፕሬስ መግለጫዎ... የሚቀመጡበት	4	4

No	Student Answer	Score_1	Score_2
	ተመሳስሎአቸው ደግሞ ሁሉም ሁሉም በሚዲያ አማካኝነት የሚቀርቡ ወይም ለሚዲያ የሚቀርቡ ጉዳዮች ናቸው።		
6	ራስንየማስተዋወቅ ውጤት የሚላከው 1- ከሚዲያ በተስጠው ሽፋን 2- ከማህበረሰቡ በተፈጠረ አስተያየት 3- በተወሰዱ ተግባራዊ እርምጃዎች	4	4
7	1.አዳዲ ነገሮች ማቅረብ 2.ተጨባጭና አካባቢያዊ ይዘቶችን ማቅረብ 3.አዎንታዊ ምላሽ (መልስ) የሚያሰጡ ነገሮች ማቅረብ 4.ለተግባር የሚያሰነሳሱ ጉዳዮች ማቅረብ	2	3

Declaration

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all sources of materials for the thesis have been duly acknowledged.

_____

Abebawu Eshetu Yigezu

This thesis has been submitted for examination with my approval as an advisor.

_____

Fekade Getahun, PhD

Addis Ababa

Jimma, Ethiopia November 2017