An Automated Essay Grading Framework Based on Neural Networks

A. E. Hassan
Electrical Engineering Dept.
Faculty of Engineering, Mansoura University, Egypt.
arwaahmedl@yahoo.com

A. M. Riad
Head of Information System Dept.
Faculty of Computer & Information Sciences, Mansoura University, Egypt.
amriad2000@yahoo.com

A. I. Shlabab
Teaching Assistant
Faculty of Computer & Information Sciences, Mansoura University, Egypt.
Shehab_083@yahoo.com

Abstract
This paper presents a proposed framework depending on Artificial Neural Networks (ANN) that enhance the process of assessing student's essays in order to save the time and cost of manual scoring of these essays. This framework consists of five phases and the UML model of that framework is defined. Also the activity diagram of that system are explained. During the learning algorithm phase, the network trained on a set of pre-graded student responses by teachers and then a set of essays not used in the training are introduced to the network to judge the essay. The proposed framework was tested on case study and presents good results compared to a previously published work using the same case study with another technique.

Keywords: Essay grading, Automatic grading, Artificial Neural Network, model.

1. Introduction
Teachers all over the world spend a great deal of time just assessing students' works. Hence, they have to cut down the time they can devote to their other duties. Even doing that, sometimes they do not have enough time to properly assess the big number of students they have.

The impacts of computers on writing have been widely studied for three decades. Computers could help in assessing students' work. Revision and feedback are essential aspects of the writing process. Students need to receive feedback in order to increase their writing quality. However, responding to student papers can be a burden for teachers. Particularly if they have a large number of students and if they assign frequent writing assignments, providing individual feedback to student essays might be time consuming [15].

Therefore, many authors believe that this situation has to be solved and some of them have presented the computer as a new assessing tool [13]. These authors do not attempt to substitute the teacher with the computer, but to help the teachers with the computer software. Automated Essay Grading (AEG) Systems can be very useful because they can provide the student with a score as well as feedback within seconds.

The grading system is a mechanism used to determine student's ability of the given material of the studying process. Essay is one form of grading, where there are no choices of answer, and the student must answer in sentence. Essay answers may vary greatly between each exam participant, depending on their own thought.

The automatic understanding of text is a complex undertaking. Computers that understand natural language have long been a goal within the Artificial Intelligence (AI) community. Many of the early approaches to understanding text were built on methods to parse unstructured text, use predefined knowledge, and generate natural language. Language's words are the basic units that are assembled in a sentence predicated by a grammar[18].

The work reported here automatically grades a set of student's responses written in English and proves...
that we achieve 85% for the agreement with Human Raters and compare these results with the results found in [13] and provide a detailed discussion.

2. Literature review
This section briefly presents an overview of the current approaches to the automated grading of free text answers (essays). Below is a description of a five systems currently available either as commercial systems or as the result of research in this field. These systems are widely used by testing companies, universities, and public schools.

2.1 Project Essay Grader (PEG).
The Project Essay Grader (PEG) was developed by Ellis Page in 1966 upon the request of the College Board, which wanted to make the large-scale essay scoring process more practical and effective. PEG uses proxy measures to predict the essential quality of the essays [14,20]. The scoring methodology is simple. The system contains a training stage and a scoring stage. PEG is trained on a sample of essays in the former stage. In the latter stage, proxy variables (proxes) are determined for each essay and these variables are entered into the prediction equation. Finally, a score is assigned by computing beta weights (coefficients) from the training stage [4]. This system achieves results reaching a multiple regression correlation as high as 0.87 with human graders [21].

2.2 Intelligent Essay Assessor (IEA)
Another AEG system, the Intelligent Essay Assessor (IEA), analyzes and scores an essay using a semantic text analysis method called Latent Semantic Analysis (LSA) [6]. Latent Semantic Analysis (LSA) is defined as “a statistical model of word usage that permits comparisons of the semantic similarity between pieces of textual information” [7]. A test conducted on Graduate Management Admission Test (GMAT) essays using the IEA system resulted in percentages for adjacent agreement with human graders between 85%-91% [21].

2.3 E-rater
The Electronic Essay Rater (E-rater) was developed by the Educational Testing Service (ETS) to evaluate the quality of an essay by identifying linguistic features in the text [2,3]. The E-Rater uses a combination of statistical and Natural Language Processing (NLP) techniques to take out linguistic features from the essays to be graded. Essays are evaluated against a benchmark set of how to mine text written by students such that these data can be present to a learning algorithm capable of learning and finally assign a grade.

human graded essays [2,12]. Over 75,000 Graduate Management Admission Test (GMAT) essays have been scored, with agreement rates between human expert and system consistently above 97%. By comparing human and E-Rater grades across 15 test questions, the empirical results range from 87% to 94% [21].

2.4 IntelliMetric
IntelliMetric, an AEG system developed by Vantage Learning, is known as the first essay-scoring tool that was based on AI [6,22,23]. IntelliMetric relies on NLP, which determines “the meaning of a text by parsing the text in known ways according to known rules conforming to the rules of the English language” [25]. The analysis shown confirms that IntelliMetric can reliably and accurately score student essays. Across all educational levels and literary writing, IntelliMetric is able to achieve a very high rate of agreement with expert scores. IntelliMetric achieved an exact agreement rate of 76% and an adjacent agreement rate greater than 99%. The majority of models achieved a 100% adjacent agreement rate. Finally, the Pearson correlations between the human and IntelliMetric scores averaged 0.93 across all forty prompts, indicating that the linear relationship between human scores and IntelliMetric scores was extremely strong [21].

2.5 Bayesian Essay Test Scoring System (BETSY)
The final automated essay scoring system to be mentioned in this paper is the Bayesian Essay Test Scoring System (BETSY), which was developed by Lawrence M. Rudner. There are two Bayesian models widely used in text classification: the Multinomial Bernoulli Model and the Multinomial Model. In the Bernoulli model, the conditional probability of presence of a specific feature is estimated by the proportion of essays within each category that include the feature. In Multinomial model, on the other hand, the probability of each score for a given essay is computed as the product of the probabilities of the features included in the essay [19]. The Bayesian approach includes key concepts such as stemming, stop words, and feature selection. Stemming denotes the process of eliminating suffixes to get stems. For example, obtaining “educ” as a stem for educate, education, educates, educational, and...
edicated. Stop words refer to various articles, pronouns, adjectives, and prepositions. One approach to feature selection is the reduction in entropy. By minimizing entropy, it is possible to pick the items with maximum potential information gain [19]. The authors in [19] used about two text classification models that were calibrated using 462 essays with two score points. The calibrated systems were then applied to 80 new pre-scored essays, with 40 essays in each score group. An accuracy of over 80% was achieved with the described dataset [21].

3. The proposed framework

The proposed framework is shown in figure 1. The mission is more similar to document clustering [16]. Assume there are five groups (output clusters) with score 0's through 2.0's as shown in figure 1. Then the documents which act as the student answers are presented as an input to the network and the ANN learning algorithm can be trained to learn weights for each of the connections in the network using a set of scored responses.

![Diagram](https://example.com/diagram.png)

**Fig.1. Abstract view of the proposed system**

3.1 Overall schematic diagram of the proposed framework

Figure 2. shows a schematic diagram of the proposed framework which indicates that there are five phases of that proposed system (check spelling, document tokenization, remove stopwords, stemming to a root, and learning algorithm). This system works as shown in figure 2:

![Diagram](https://example.com/diagram2.png)

**Fig.2. Automated essay grading system’s overall schema**

3.1.1 Check spelling

Before we introduce the keywords (words found in the answer) to the ANN, we first make a process called check spelling in order to assure that the data that was written by the student is free of errors. The check spelling process is a very critical step in our algorithm because if we assume that a student for example had written an answer as follows:

"Standard for network connection"

Here the word network is misspelled and hence the computer will treat the network as a different word than network (i.e. the network keyword takes a 0 weight). On the other hand, if the check spelling used, it automatically correct network to network (i.e network take here 1 weight). The check spelling module notifies the student with the suggestions and the student has the decision whether to accept the modification or to ignore.

3.1.2 Document tokenization

The second phase is to identify useful features from the student answer. This is done by breaking the stream of characters into words or, more precisely, tokens as shown in figure 3. This is a fundamental step to further analysis. Without identifying the tokens, it is difficult to extract higher-level information from the document. Breaking a stream of characters into tokens is trivial for a person
familiar with the language structure. A computer on the other hand would find the task more complicated [24]. The reason is that certain
characters are sometimes taken as a token delimiters and sometimes not, depending on the nature of the problem being solved. The characters
space, tab, and newline we assume are always delimiters and are not counted as tokens. They are often collectively called white space.
The characters ( ) < > ! ? " are always delimiters and may also be tokens. Also the characters . . . : - ' may or may not be delimiters, depending on
their environment.
In the case under study, only the white space is taken into consideration as a delimiter. For example, if the first answer in the data sample is
passed to the application, the output would be a matrix of “I”, “have”, “not”, “got”, “a”, and “clue” terms.
To get the best possible features, one should customize the tokenizer for the available text—otherwise extra work may be required after the
tokens are obtained. The reader also should note that the tokenization process is language-dependent. The proposed work focuses on
documents in English. For other languages, the general principles will be the same and the details will differ.

Fig.3. Document tokenizes into a set of terms through tokenizer.

3.1.3 Remove stopwords
Stop words, or stopwords, is the name given to words which are filtered out prior to, or after, processing of natural language data (text)[9].
We trim the stopwords (e.g. “a”, “the”, “in”, “was”, “got”, “have”) from the student answer in the shown documents and treat the remaining words as
the keywords that are used to distinguish good essays from bad ones.
For example, the output from the previous step (tokenization) is taken and introduced as an input to this current phase (remove stopwords); the output
would be only the word “clue” as a keyword ready to the next phase.
Hence, the presence or absence of a word is to be a measured attribute for each document (binary weights). The results of the proposed case study is
shown in Table 1. The table is a spreadsheet where the cells are filled with one for the presence of a word and zero for its absence.

3.1.4 Stemming to a root
Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form — generally a written word form. The
stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is
not in itself a valid root [8].
Removing suffixes by automatic means is an operation which is especially useful in the field of Information Retrieval (IR). In a typical IR
environment, one has a collection of documents, each described by the words in the document title and possibly by words in the document abstract.
Ignoring the issue of precisely where the words originate, we can say that a document is represented by a vector of words, or terms. Terms
with a common stem will usually have similar meanings, for example: CONNECT, CONNECTED, CONNECTING, CONNECTION, CONNECTIONS
Frequently, the performance of an IR system will be improved if term groups such as this are conflated into a single term. This may be done by
removal of the various suffixes “-ED, -ING, -ION, IONS” to leave the single term CONNECT. In addition, the suffix stripping process will reduce
the total number of terms in the IR system, and hence reduce the size and complexity of the data in the system, which is always advantageous.
The nature of the task will vary considerably depending on whether a stem dictionary is being used, whether a suffix list is being used, and of
course on the purpose for which the suffix stripping is being done [17].
The suffix stripping program will be given an explicit list of suffixes, and, with each suffix, the criterion under which it may be removed from
a word to leave a valid stem.
Our stemming efforts in our research will be restricted to the more traditional Porter Stemmer which follows the affix removal approach [17].
Hence the network trained on the keywords without stemming and also with using stemming words based on Porter stemmer algorithm. Table 2 shows
the original words and the stemmed words of the case under study.

Table 1. Words mentioned in documents and their occurrence in each document after removing the stopwords

<table>
<thead>
<tr>
<th>Words</th>
<th>Doc 0</th>
<th>Doc 1</th>
<th>Doc 2</th>
<th>Doc 3</th>
<th>Doc 4</th>
<th>Doc 5</th>
<th>Doc 6</th>
<th>Doc 7</th>
<th>Doc 8</th>
<th>Doc 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>acceptance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>allows</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>audio</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>clue</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>connection</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>conversation</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>h.323</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>internet</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>iso</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>level</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>management</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>meeting</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>network</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>project</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>protocol</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>provided</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>received</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>related</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>remote</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sending</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sent</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>standard</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>streaming</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>style</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>telephone</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>transmissions</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>video</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>videoconferencing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2. Original words and stemmed words

<table>
<thead>
<tr>
<th>Words before stemming</th>
<th>Words after stemming</th>
<th>Words before stemming</th>
<th>Words after stemming</th>
</tr>
</thead>
<tbody>
<tr>
<td>acceptance</td>
<td>accept</td>
<td>protocol</td>
<td>protocol</td>
</tr>
<tr>
<td>allows</td>
<td>allow</td>
<td>provided</td>
<td>provide</td>
</tr>
<tr>
<td>audio</td>
<td>audio</td>
<td>received</td>
<td>receiv</td>
</tr>
<tr>
<td>clue</td>
<td>clue</td>
<td>related</td>
<td>relat</td>
</tr>
<tr>
<td>connection</td>
<td>connect</td>
<td>remote</td>
<td>remot</td>
</tr>
<tr>
<td>conversation</td>
<td>converers</td>
<td>sending</td>
<td>send</td>
</tr>
<tr>
<td>h.323</td>
<td>h.323</td>
<td>sent</td>
<td>sent</td>
</tr>
<tr>
<td>internet</td>
<td>internet</td>
<td>standard</td>
<td>standard</td>
</tr>
<tr>
<td>iso</td>
<td>iso</td>
<td>streaming</td>
<td>stream</td>
</tr>
<tr>
<td>level</td>
<td>level</td>
<td>style</td>
<td>style</td>
</tr>
<tr>
<td>management</td>
<td>manage</td>
<td>telephon</td>
<td>telephon</td>
</tr>
<tr>
<td>meeting</td>
<td>meet</td>
<td>transmissions</td>
<td>transmis</td>
</tr>
<tr>
<td>network</td>
<td>network</td>
<td>video</td>
<td>video</td>
</tr>
<tr>
<td>project</td>
<td>project</td>
<td>videoconferencing</td>
<td>videoconference</td>
</tr>
</tbody>
</table>
3.1.5 Learning algorithm

Learning vector quantization (LVQ) is a method for training competitive layers in a supervised manner. A competitive layer automatically learns to classify input vectors. However, the classes that the competitive layer finds are dependent only on the distance between input vectors. If two input vectors are very similar, the competitive layer probably will put them in the same class. There is no mechanism in a strictly competitive layer design to say whether or not any two input vectors are in the same class or different classes. LVQ networks, on the other hand, learn to classify input vectors into target classes chosen by the user [10].

LVQ is based on a set of input/target pairs (n pairs).

\[\{(p_1, t_1), (p_2, t_2), \ldots, (p_n, t_n)\}\]

Each target vector has a single 1. The rest of its elements are 0. The 1 tells the proper classification of the associated input. For instance, consider the following training pair:

\[
\begin{bmatrix}
1 \\
1 \\
0
\end{bmatrix}
\begin{bmatrix}
t_1 \\
t_2 \\
t_3
\end{bmatrix}
\]

Here there is an input vector (P1) of three elements and each input vector is to be assigned to one of four classes. The network is to be trained so that it classifies the input pattern (P1) as one of the four classes shown in "11".

Training phase: the teacher is responsible for feeding the net with a set of pre-graded answers, and then the motivation for the algorithm is to find the output unit (grade) that is closest to the input vector (answer). The teacher trains the network until it learns the weights that have the ability to correctly classify the different types of student's answers.

To train the network in order to obtain first-layer weights that lead to the correct classification of input vectors, first set the training epochs to 150. Then, use keyword train as follows:

```python
net.trainParam.epochs = 150;
net = train(net,p,t);
```

In the experiment, the parameters were 28 input neurons that represent the total number of keywords, 14 hidden neurons, 5 output neurons, 150 as a number of epochs. Figure 4 shows this architecture which is based on LVQ algorithm.

![LVQ architecture diagram](image)

**Fig. 4. LVQ architecture**

Testing phase: In this phase, the student's answer of the un-graded set is presented to the system. Then it follows the pre-mentioned steps until converting the answer into an input vector that can be presented to the LVQ net in order to assign a grade.

3.2 UML model of the proposed system

The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems [1].

Models are the blueprints for systems. A blueprint helps you plan an addition before you build it; a model helps you plan a system before you build it. It can help you be sure the design is sound, the requirements have been met, and the system can withstand even a hurricane of requirement changes.

**Visual modeling** is the process of taking the information from the model and displaying it graphically using a standard set of graphical elements [26]. In this section, both use case diagram and activity diagram are presented.

**A use case diagram**

It presents a graphical overview of the functionality provided by a system in terms of actors, their goals—represented as use cases—and any dependencies between those use cases [1]. An actor is anyone or anything that interacts with the system being built. A use case is a high-level piece of functionality that the system will provide. In other words, a use case illustrates how someone might use the system.

In brief, automated essay grading system's use case diagram shown in figure 5 include three actors:
1. Admin who is responsible for managing and controlling access to the system (give privileges).

2. Instructor who is responsible for creating an exam and browsing student's comments and suggestions on the system.

3. Student who is responsible for enrolling an exam, receiving feedback on his/her answer, and submitting complaints.

And the detailed use cases are as follows:

1. Take Exam which uses a Generate Exam use case in order to prepare an exam to the incoming student.

2. View feedback use case which is responsible for providing students with feedback on their writing.

3. Comments and suggestions use case where the student can submit his/her comments and suggestions and the instructor in turn views these comments and suggestions.

4. Get User Name & Password child use-case which gives student and instructor the ability to get their user name and password to login to the system.

5. Make Exam & Train network use-case which uses pre-graded essays where the instructor sets the exam and then trains the network on a set of these pre-graded essays.

6. Update student profile child use-case which notifies the student with any updates in the profile.

7. Update instructor profile child use-case which notifies the instructor with any updates in the profile.

8. Manage access to the system use-case which manages login privileges for both student and instructor.

Fig. 5. Use case diagram of the proposed system
Activity diagram

An activity diagram is a way to model the flow of events. The activity diagram is a UML model and it is a better way to describe the flow of events of the proposed framework components. Using text to describe the flow of events is more complex than using diagrams [1, 26]; activity diagrams are more readable from the prospective of customers. Activity diagrams present the same information as a textual flow of events would, and it is easy to handle and understand.

The following are three activity diagrams (student, instructor, and admin). These diagrams were created to depict the flow of events within the proposed framework models.

1- Student's activity diagram: student's activity diagram (as shown in Figure 6) works as follows:

1. Start
2. Get user name and password
3. If both user name and password are correct then goto step 5
4. If no of iterations reach five then goto 9
5. Display a question
6. If student fill a blank answer then notify student and goto step 5
7. If student's answer has any errors then goto check spelling engine else goto 8
8. Assess the answer and notify the student
9. Exit

Fig. 6. Student's activity diagram
2- Instructor's activity diagram: It works as follows as shown in Figure 7:

1. Start
2. Get user name and password
3. If both user name and password are correct then goto step 5
4. If no of iterations reach five then goto 11
5. If instructor like to set exam then goto 7
6. If instructor like to browse comments and suggestions then goto 10
7. Train network with a set of pre-graded essays
8. Test network to measure the performance of the network
9. Add the question to the DB and goto 11
10. Display students' comments and suggestions and instructor either reply or not
11. Exit

Fig. 7. Instructor's activity diagram

3- Administrator's activity diagram: It works as follows as shown in Figure 8:

1. Start
2. Manage account
3. New account or old account
4. If new account then goto 6
5. If old account then goto 9
6. Student or instructor
7. If student then set account, set student privileges, notify student, and goto 12
8. If instructor then set account, set instructor privileges, notify instructor, and goto 12
9. Modify or delete account
10. If modify then update account, notify either student or instructor and goto 12
11. If delete then delete account
12. Exit
4. Data sample
For this study, the same data sample found in [13] is used and the essay question is "what is H.323 standard? (2pts.)".

The following table shows the answers of students which are used as data sample in this case study. Both answer No.6 and No.8 are the same so that these answers are treated as an only one document (pattern) such that there are nine answers as a total number of document collections.

<table>
<thead>
<tr>
<th>Doc ID</th>
<th>Document Text</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>I have not got a clue</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>A level of acceptance</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>A standard for remote meeting protocol</td>
<td>1.5</td>
</tr>
<tr>
<td>3</td>
<td>This is a standard for videoconferencing</td>
<td>1.5</td>
</tr>
<tr>
<td>4</td>
<td>H.323 is an Internet standard that allows audio to be sent and received in the style of a telephone conversation</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>The standard used for videoconferencing and sending streaming audio and video via the Internet</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>Standard for videoconferencing transmissions</td>
<td>1.5</td>
</tr>
<tr>
<td>7</td>
<td>Y is the standard provided by ISO and related to project management</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>Standard for videoconferencing transmissions</td>
<td>1.5</td>
</tr>
<tr>
<td>9</td>
<td>Standard for network connection</td>
<td>1</td>
</tr>
</tbody>
</table>
5. Results
After training the neural network on the previously shown answers and trying testing the neural network, we found that all answers are correctly classified except only one answer was misclassified and also the results show no changes in the output either using stemming or not. The reason is that the datasets are small so it is not too large to make stemming affect the results obtained.

<table>
<thead>
<tr>
<th>Table 4. Final results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technique used</td>
</tr>
<tr>
<td>Technique used in [13]</td>
</tr>
<tr>
<td>Proposed technique with stemming</td>
</tr>
</tbody>
</table>

6. Conclusion and future work
In this paper, an essay grading system is proposed to automatically assess students' essays based on neural networks. This system is implemented using MATLAB environment[5]. The results show accuracy 85% which clearly better than results reached in [13]. Finally, it seems that ANN achieves good results and could be an effective approach to handle this kind of problems. It is noteworthy that the architecture of the essay grading system does not take the human reader out of the loop. Indeed, because the system requires initial training sets of manually graded essays. A drawback for this system is the requirement of a set of pre-graded responses that are needed to train the network. In some cases, the system may need more than 20 essays in order to build a reliable network be able to correctly grade the answers. This would be an obstacle in the datasets that are small. We are currently experimenting with an interactive system that can be used to improve, or extend, the model automatically even when the data sets are small. A future enhancement to the above mentioned framework would be the extraction of semantic features included in the answer such as length of essay, word count, amount of punctuation, and choice of vocabulary. Also how much the sentences are logically connected (organization of ideas) is an interesting open question. In the future work, Arabic essays will be investigated.

References


