**Documentation of Automatic Subjective Answer Evaluation**

**By Rupesh Yadav**

**Subjective Answer Evaluation System**

by Aditi Tulaskar, Aishwarya Thengal, Kamlesh Koyande, (2017), (Cited by 2)

*(Very basic paper, not of much use. You can go through it if you are new to this field, otherwise leave it.)*

It evaluates each answer by matching the keywords or the key concepts as well as its synonyms with the standard answer. It also checks the grammar and spellings of the words. The process consists of 3 steps: keywords and synonyms extraction, matching of keywords and generating score.

**Subjective Answer Evaluation using Machine Learning**

by Piyush Patil, Sachin Patil, Vaibhav Miniyar, Amol Bandal, (2018), (Cited by 2)

*(If you’re a beginner, then take a look, otherwise leave it)*

The proposed algorithm performs tasks like Tokenizing words and sentences, Part of Speech tagging, Chunking, chinking, Lemmatizing words and Wordnetting to evaluate the subjective answer. This model used Naive Bayes classifier. It is based on three parameters i.e. Keywords, Grammar and Question Specific things (QST). These parameters are passed to NB classifier as input and a Class value is returned b/w 1-10 as output.

**An Introduction to Latent Semantic Analysis**

by Thomas K Landauer, Peter W. Foltz, Darrell Laham, (1998), (Cited by 5965)

This paper introduces a machine learning technique called Latent Semantic Analysis (LSA). It is an Information Retrieval (IR) technique used for automatic essay scoring. This technique mimics the human understanding of the meaning of the natural language. In LSA, a training text is represented as a matrix using the frequency of the word in the text, followed by Singular Value Decomposition (SVD) of the matrix. The result is a 100-500 dimensional “semantic space”, where the original words and passages are represented as vectors. The relation between two columns (documents) can be identified using Spearman's correlation. Another way is cosine similarity.

**Automated Essay Scoring Using Generalized Latent Semantic Analysis**

by Md. Monjurul Islam, A. S. M. Latiful Hoque, (2010), (Cited by 62)

The main concepts in Generalized Latent Semantic Analysis (GLSA) are the same as in LSA. The difference is that GLSA preserves the word order in a sentence. Therefore, in GLSA, n-gram by document matrix is created instead of a word by document matrix of LSA. The whole system architecture is divided into two main parts: The generation of training essay set and the evaluation of submitted essays using training essay set.

**Note**: Essays are generally evaluated for their writing quality and style.

**Automatic Essay Scoring with e-rater V.2**

by Y. Attali and J. Burstein, (2006), (Cited by 798)

This paper discusses the Automated Essay Scoring (AES) system. It discusses it on a surface level and in-depth details have not been provided due to proprietary reasons. It is based on automated statistical methods which uses a small set of meaningful and intuitive features. The feature set used with e-rater V.2 include measures of grammar, usage, mechanics, style, organization, development, lexical complexity, and prompt-specific vocabulary usage.

**Automatic Assessment of Students' Free-text Answers Underpinned by the combination of a BLEU-inspired Algorithm and Latent Semantic Analysis**

by Diana Perez, Alfio Gliozzo, Carlo Strapparava, Enrique Alfonseca, Pilar Rodriguez, Bernardo Magnini, (2005), (Cited by 72)

This paper compares the BLEU-inspired algorithm (*Papineni et al. 2001*, used for evaluating Machine Translation Systems) (which they called ERB) with the Latent Semantic Analysis (LSA). The best accuracy has been obtained by combining ERB and LSA with a lower weight assigned to ERB. For evaluation purposes, the Pearson’s correlation coefficient between the humans’ scores and the system’s scores is calculated.

* **Advantages of Statistical methods:**

The advantage is that they are designed to find optimal solutions with respect to some measure of agreement between human and machine scores.

* **Disadvantages of Statistical methods:**

A writing feature might be an important determinant of the score in one solution and absent from another. Moreover, features might contribute positively to the score in one solution and negatively in another. That is because they are based on a large number of features that are relatively highly correlated among themselves and have relatively low correlations with the criterion (the human scores).

Another disadvantage is that such models may be difficult to describe and explain to users of the system. Difficulty in communicating the inner structure of the scoring model is a threat to the face validity of these systems.

**A Neural Approach to Automated Essay Scoring**

by Kaveh Taghipour and Hwee Tou Ng, (2016), (Cited by 166)

In this paper, a RNN based approach has been developed to learn the relation between an essay and its assigned score, without any feature engineering. Since the system is based on RNNs, it can effectively encode the information required for essay evaluation and learn the complex patterns in the data through non-linear neural layers. The best model identified is a LSTM neural network and is trained as a regression method.

*Model Architecture*:

1. **Lookup Table layer** - (Let ‘M’ words are there) Each word’s one-hot representation is converted into a ‘dLT’ dimensional space vector by multiplying with an embedding matrix.

2. **Convolution layer** - It can be used as a function that extracts feature vectors from n-grams. It can capture local contextual dependencies in the essay.

3. **Recurrent layer** - LSTM outperforms the basic RNN and GRU.

4. **Mean over time** - This layer receives ‘M’ vectors as input and calculates an average vector of the same length.

5. **Linear layer with Sigmoid activation** - It maps the input vector generated by the mean-over-time layer to a scalar value. All the target scores are normalized to [0,1] when training.

For training, RMSProp optimization algorithm is used. Dropout regularization and gradient clipping is also used. Quadratic Weighted Kappa (QWK) is used as the evaluation metric.

**C-rater: Automated Scoring of Short-Answer Questions**

by Claudia Leacock, Martin Chodorow, (2003), (Cited by 351)

C-rater is a short-answer scoring engine, developed by ETS, which recognizes paraphrase or equivalent meaning. In order to recognize that answers express a common meaning, c-rater generates a *canonical (normalized) representation* of each answer. To build this representation, it normalizes across four primary sources of variation among sentences: 1. Syntactic Variation (Active vs Passive); 2. Pronoun reference; 3. Morphological variation (believed, believing, beliefs); 4. And the use of synonyms and similar words.

After that, it tries to match the concepts in the student’s answer to the concepts in the correct answer and then assigns a score depending on the number of concepts that are matched.

**Using a MaxEnt Classifier for the Automatic Content Scoring of Free-Text Responses**

by Jana Z. Sukkarieh, (2011), (Cited by 20)

This paper reduces the task of automatic content scoring to a Textual Entailment (TE) task (paraphrase or inference). Given a concept (C) and a student response (A), find whether A entails C or not. For this purpose, they are using a Maximum Entropy model (a probability distribution estimation technique) as a classifier with some features as input. The feature functions used are not very clear. It claims that MaxEnt achieves a higher accuracy than a rule-based approach.

**Note:** If you are proceeding with the rule-based matching approach, there are some example pairs given in the above paper which will give an idea on how to create rules.

**Towards Robust Computerised Marking of Free-Text Responses**

by T. Mitchell, T. Russell, P Broomhead, N Aldridge, (2002), (Cited by 187)

This paper discusses a software system, Automark, developed for scoring short answers. The system employs a mark scheme (as a syntactic-semantic template) that specifies acceptable and unacceptable answers for each question. After the preprocessing of the student response, it is matched with the answer template. Later, the paper discusses the analysis of the system through various experiments and tests.

**CAA of short non-MCQ answers**

by DH. Callear, J. Jerrams-Smith, V.Soh, (2001), (Cited by 94)

This paper presents an Automated Text Marker (ATM) prototype for an effective conceptual pattern matching of a student's answer with those of a model examiner’s answers. The focus of this paper is on the semantic analysis of text contents. Text passages are broken down into their smallest viable unit of concepts. Basic concepts and their dependencies are reclustered and grouped together (examples are shown in the paper). Successively larger concepts or dependency groups are derived with pointers to other dependency groups or case frames.

**Automatic Short Answer Marking**

by S. Pulman, J. Sukkarieh, (2005), (Cited by 126)

This paper also uses pattern matching for scoring short-answers. The pattern basically is all the paraphrases collapsed into one. There are some cases provided where it is challenging to evaluate the answers (Pg-2). Then, the results of machine learning techniques, like Inductive Logic programming, decision tree learning and Naive Bayesian learning, used for this purpose have been discussed.

**Kaggle Competition (The Hewlett Foundation): Short Answer Scoring**

Winners, 1st Place: Luis Tandalla, (2012)

Misspelled words are corrected. All words are stemmed. To predict the scores of the essays, Random Forests were trained using the following features: “the counts of the important words”, “the counts of the important bigrams”, “the answers found by the regular expressions (manually generated)” and “the probabilities of having the answers”.

The Boruta algorithm was used to select the relevant words and bigrams to predict the scores, was used to select the relevant words, bigrams and trigrams to predict the labels (which will be used to predict the probabilities) and then also used to select the relevant answers and probabilities to predict scores.

**A Word-Order Based Graph Representation For Relevance Identification**

by Lakshmi Ramachandran, Edward F. Gehringer, (2012), (Cited by 10)

Word-order based graph representation performs better than a dependency tree representation while identifying the relevance of one piece of text to another. Parsers may be used to generate dependency trees. Dependency trees succeed in capturing only governance information. They do not capture ordering information. Word-order graph representation extends the dependency-tree based representation to capture word-ordering information.

Graph generation includes the following steps: (1) Dividing text into segments. (2) Part-of-speech (POS) tagging. (3) Vertex and Edge creation (steps are explained in Algorithm 1 in paper): Basically, consecutive subject components are combined to form a subject vertex (similarly with consecutive verbs, adjectives and adverbs). Then, while creating an edge, order is maintained i.e., if a verb vertex was formed before a subject vertex, a verb-subject edge is created, else a subject-verb edge is created. (4) Labeling graph edges: Graph edges are labeled with dependency (word-modifier) information.

Finally, for matching the two graphs, their vertices and edges are matched. Vertex match between two texts is the average of the best match for each of their constituent vertices. For edge matching, if two edges have the same labels, then an average of their vertex match gives the edge match. The semantic match between tokens is determined using WordNet.

**Identifying Patterns For Short Answer Scoring Using Graph-based Lexico-Semantic Text Matching**

by Lakshmi Ramachandran, Jian Cheng and Peter Foltz, (2015), (Cited by 40)

Manually developed regular expressions can provide effective scoring, however manual development can be quite time consuming. Here, an approach to automatically identify patterns (regular expressions) using the rubric text and top-scoring student responses is being discussed.

Two types of patterns are generated: (1) content words and (2) sentence structure information (phrase patterns).

The approach is to (1) identify groups of semantically related words or phrases that a human evaluator would expect to see among the best answers, and (2) combining these semantic groups in a meaningful way to generate patterns.

**Extracting Content Tokens**: Stopwords are eliminated. The approach is given in Algorithm 1 in the paper. The tokens extracted do not have to appear in any particular order within the student answer.

**Extracting Phrase Patterns**: Here, the extraction process involves generation of *word-order graph representations* for the sample answers, and extracting edges representing structural relations. Stopwords are substituted with regular expressions. The approach is given in Algorithm 2 in the paper. Here, the order of tokens is to be maintained.

Then, instead of using these patterns to directly match and score student answers, they are supplied as features to a learning algorithm such as Random Forest in order to accurately predict scores. This approach is compared to the Tandalla’s approach and Mohler’s method and it results in comparable performance.

**Measuring the Semantic Similarity of Texts**

by C. Corley, R. Mihalcea, (2005), (Cited by 413)

This paper introduces a method that combines word-to-word similarity metrics into a text-to-text metric. The semantic similarity of texts is defined as a function of the semantic similarity of the component words. They did it by combining metrics of word-to-word similarity and language models into a formula that is a potentially good indicator of the semantic similarity of the two input texts.

To find the semantic similarity of words, the following word similarity metrics are used: Leacock & Chodorow, Lesk, Wu & Palmer, Resnik, Lin, and Jiang & Conrath. In addition to the semantic similarity of words, the specificity of words is also taken into account, for which the inverse document frequency is used.

For semantic similarity of texts, a separate set of open-class words is created for nouns, verbs, adjectives, and adverbs and then pairs of similar words across the sets corresponding to the same open-class in the two text segments is determined.

This is a bag-of-words approach which ignores many of the important relationships in sentence structure, such as dependencies between words, or roles played by the various arguments in the sentence.

**Note**: **Text-to-text Semantic Similarity for Automatic Short Answer Grading.** This paper is quite identical to the above paper and is of the same author published in 2009.

**Learning to Grade Short Answer Questions using Semantic Similarity Measures and Dependency Graph Alignments**

by M. Mohler, R. Bunescu, R. Mihalcea, (2011), (Cited by 202)

This paper combines several graph alignment features with lexical semantic similarity measures using machine learning techniques. It attempts to align the dependency graphs of the student and the instructor answers in order to make use of a structural component in the automatic grading of student answers. This approach is different from the related tasks of paraphrase detection and textual entailment as it provides a grade on a certain scale rather than make a simple yes/no decision. The dependency-graph approach used here is similar to those more commonly used in the textual entailment community.

It introduces a three-stage pipeline to grade short answers. In the first stage, node matching is done across the dependency graphs provided. In the second stage, graphs are aligned using the Hungarian Algorithm to find an optimal matching and the score related to it. In the third stage, the alignment score (calculated in the previous stage) as well as BOW similarity scores (previous paper) are used as features for Support Vector Machine (SVM) for grading.

**Robust Textual Inference via Graph Matching**

by A. Haghighi, A. Y. Ng, C. D. Manning, (2005), (Cited by 168)

This paper matches the graph-based representation of sentences for recognizing textual entailment. For that, first both input texts are converted into a graph by using the dependency relations obtained from a parser. Next, a matching score is calculated, by combining separate vertex and edge matching scores. The vertex matching functions use word-level lexical and semantic features to determine the quality of the match while the edge matching functions take into account the types of relations and the difference in lengths between the aligned paths.

A very intuitive approach is provided for node and edge matching. Some checks are also provided to counter some systematic source of errors.

Below are some papers which discusses some other papers, their classification and their comparison:

**Machine Learning Techniques with Ontology for Subjective Answer Evaluation**

by M. S. Devi, H. Mittal, (2016), (Cited by 15)

**The Eras and Trends of Automatic Short Answer Grading**

by S. Burrows, I. Gurevych, B. Stein, (2015), (Cited by 167)