

StatKeyEval

A Statistical Framework for Dynamic Keyword Extraction, Evaluation, and Assessment Automation

Aim:

To implement an automatic short-answer grading system using feature engineering and ensemblebased approaches, focusing on extracting keywords, computing similarity metrics, and generating confidence scores.

Algorithm:

1. Text Preprocessing

- Convert all text to lowercase.
- Remove punctuation marks and numbers.
- Remove common stop words (e.g., "the", "is", "and").
- Strip extra whitespace.

2. Keyword Extraction (Using IGRKE - Information Gain Ratio Keyword Extraction)

- Split preprocessed text into individual words.
- Compute Information Gain Ratio (IGR) for each word based on its entropy and conditional probability within reference answers and student responses.
- Adjust for word significance using the Frequency Adjustment Factor (FAF) to balance importance across reference and response texts.
- Rank words based on $IGR \times FAF$ score and extract the most informative keywords.
- Store extracted keywords for both reference answers and student responses.

3. Keyword Mutation (Using SCM - Statistical Co-occurrence Mutation)

- Group responses by question.
- Construct a co-occurrence matrix of word pairs based on document frequency.
- Compute Pointwise Mutual Information (PMI) between words to identify semantically related terms.
- Identify frequently occurring words ($\geq 65\%$ of responses) with high PMI similarity to existing reference keywords.
- Apply Uniqueness Filtering to ensure new keywords add distinct meaning.
- Add these mutated keywords to reference keyword sets for more flexible grading.

4. Vector Representation

- Create a universal keyword list combining all extracted and mutated keywords.
- Represent each reference answer and student response as a binary vector:
- 1 if the keyword is present, 0 if absent.
- Normalize vectors to account for varying answer lengths.

5. Similarity Calculation (Using Multi-Dimensional Similarity Function)

- Compute four similarity metrics between reference and student answer vectors:
- Cosine similarity (Simcos)
- Normalized Euclidean distance (Simeuc)
- Normalized Manhattan distance (Simman)
- Adjusted Pearson correlation (Simpearson)
- Compute a hybrid similarity score:
- $\text{Similarity}(A,S) = 0.4 \times \text{Simcos} + 0.3 \times \text{Simeuc} + 0.2 \times \text{Simman} + 0.1 \times \text{Simpearson}$

6. Score Generation

- Compute a weighted composite similarity score based on the hybrid similarity function.
- Scale the similarity score to match the original grading scale.
- Apply rounding to get the final predicted score.

7. Performance Evaluation

- Calculate error metrics:
- Root Mean Square Error (RMSE)
- Mean Absolute Error (MAE)
- Mean Absolute Percentage Error (MAPE)
- Generate correlation statistics (Pearson R, Spearman ρ).
- Compute coefficient of determination (R^2) to assess predictive accuracy.
- Perform error analysis across different score ranges to identify potential grading inconsistencies.

Title: Feature Engineering and Ensemble-Based Approach for Improving Automatic Short-Answer Grading Performance

Authors: Archana Sahu and Plaban Kumar Bhowmick.

Conference/Journal: Educational Data Mining Conference (2018)

Datasets:

1. UNT Dataset
2. SciEntsBank Dataset
3. Beetle Dataset

A	B	C	D	E
number	Questions	Answers	Texts	Score
1	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	High risk problems are address in the prototype program to make sure that the program is feasible. A prot	3.5
2	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	To simulate portions of the desired final product with a quick and easy program that does a small specific	5
3	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	A prototype program simulates the behaviors of portions of the desired software product to allow for error	4
4	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	Defined in the Specification phase a prototype simulates the behavior of portions of the desired software	5
5	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	It is used to let the users have a first idea of the completed program and allow the clients to evaluate the pi	3
6	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	To find problem and errors in a program before it is finalized	2
7	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	To address major issues in the creation of the program. There is no way to account for all possible bugs in	2.5
8	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	you can break the whole program into prototype programs to simulate parts of the final program	5
9	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	To provide an example or model of how the finished program should perform. - Provides foresight of so	3.5
10	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	Simulating the behavior of only a portion of the desired software product.	5
11	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	A program that simulates the behavior of portions of the desired software product.	5
12	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	A program that simulates the behavior of portions of the desired software product.	5
13	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	To layout the basics and give you a starting point in the actual problem solving.	2
14	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	To simulate problem solving for parts of the problem	4.5
15	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	A prototype program provides a basic groundwork from which to further enhance and improve a solution t	2
16	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	A prototype program is a part of the Specification phase of Software Problem Solvin. It's employed to illu	4.5
17	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	Program that simulates the behavior of portions of the desired software product	5
18	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	provides a limited proof of concept to verify with the client before actually programming the whole applic	2
19	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	It tests the main function of the program while leaving out the finer details. 	2
20	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	To get early feedback from users in early stages of development. To show users a first idea of what the pr	2.5
21	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	It simulates the behavior of portions of the desired software product	5
22	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	It simulates the behavior of portions of the desired software product.	5
23	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	A prototype program is used in problem solving to collect data for the problem.	1.5
24	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	To ease the understanding of problem under discussion and to ease the understanding of the program its	2.5
25	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	It simulates the behavior of portions of the desired software product	5
26	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	The role of a prototype program is to help spot key problems that may arise during the actual programing.	2
27	1.1 What is the role of a prototype program in problem solving?	To simulate the behaviour of portions of the desired software product.	To simulate the behaviour of portions of the desired software product.	2

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA					
number	Questions	Answers	Texts	Score	Word2Vec	Jaccard	SimpleWo	ROUGE_1	ROUGE_2	ROUGE_W	ROUGE_L	TFIDF	NovLDA	Smil	Hellinger	RF1	RF2	RF3	WordAlign	Coverage	Question	Question	Edit	Distan	Char	Gram	Word	Length	Unique	WordPro	Proportio
1	1.1 what is the to simulat	high risk p		3.5	0.950483	0.117647	4	0.333333	0	0.416667	0.2	0.221154	0.999942	1	4	6	34	0.355023	0.333333	0.285714	0.285714	152	0.107527	1.167708	0.4						
2	1.1 what is the to simulat	to simulat		5	0.949583	0.228571	8	0.666667	0.4	0.666667	0.290099	0.425	0.999008	1	8	2	35	0.488678	0.666667	0.714286	0.714286	150	0.227027	1.321354	0.8						
3	1.1 what is the to simulat	a prototy		4	0.980101	0.421053	8	0.666667	0.6	0.75	0.5625	0.267857	0.999008	1	8	2	19	0.73436	0.666667	0.714286	0.714286	49	0.495496	1.003401	0.8						
4	1.1 what is the to simulat	defined in		5	0.977263	0.25	8	0.666667	0.6	0.75	0.333333	0.428571	0.999008	1	8	2	32	0.34981	0.666667	0.714286	0.714286	182	0.276302	0.980038	0.8						
5	1.1 what is the to simulat	It is used i		3	0.964475	0.142857	5	0.416667	0.1	0.5	0.230769	0.181452	0.999005	1	5	5	35	0.559793	0.416667	0.285714	0.285714	177	0.138096	1.035688	0.5						
6	1.1 what is the to simulat	to find pr		2	0.937734	0.095238	2	0.166667	0	0.083333	0.08	0	0.999559	1	2	8	21	0.138675	0.166667	0.142857	0.142857	56	0.04387	1.253268	0.2						
7	1.1 what is the to simulat	to address		2.5	0.952782	0.137931	4	0.333333	0.1	0.416667	0.212766	0.068182	0.999001	1	4	6	29	0.450433	0.333333	0.142857	0.142857	128	0.09697	1.246891	0.4						
8	1.1 what is the to simulat	you can b		5	0.945508	0.25	5	0.416667	0.2	0.333333	0.75862	0.107143	0.003138	1	5	5	20	0.49099	0.416667	0.285714	0.285714	68	0.172414	1.058017	0.5						
9	1.1 what is the to simulat	to provid		3.5	0.957649	0.125	4	0.333333	0.1	0.5	0.448688	0.13	1	1	4	6	32	0.527148	0.333333	0.142857	0.142857	166	0.105033	1.010648	0.4						
10	1.1 what is the to simulat	simulat		5	0.989196	0.4	6	0.5	0.4	0.666667	0.515395	0.089444	0.999089	1	6	4	15	0.787222	0.5	0.428571	0.428571	14	0.6375	1.110215	0.6						
11	1.1 what is the to simulat	a program		5	0.981542	0.466667	7	0.583333	0.6	0.75	0.666667	0.15	0.999089	1	7	3	15	0.763763	0.583333	0.571429	0.571429	17	0.650602	1.053571	0.7						
12	1.1 what is the to simulat	a program		5	0.98405	0.466667	7	0.583333	0.6	0.75	0.666667	0.15	0.999089	1	7	3	15	0.763763	0.583333	0.571429	0.571429	16	0.7	1.068481	0.7						
13	1.1 what is the to simulat	to lay out		2	0.928041	0.130435	3	0.25	0	0.333333	0.266667	0.096154	0.999631	1	3	7	23	0.371001	0.25	0.142857	0.142857	58	0.076923	1.340909	0.3						
14	1.1 what is the to simulat	to simulat		4.5	0.948303	0.357143	5	0.416667	0.2	0.333333	0.363636	0.082896	0.999088	1	5	5	14	0.505181	0.416667	0.285714	0.285714	42	0.244186	1.117424	0.5						
15	1.1 what is the to simulat	a prototy		2	0.952813	0.083333	2	0.166667	0	0.166667	0.212112	0	0.003358	1	2	8	24	0.184399	0.166667	0.142857	0.142857	82	0.050211	1.086842	0.2						
16	1.1 what is the to simulat	a prototy		4.5	0.965091	0.131579	5	0.416667	0.1	0.333333	0.145455	0.263889	0.999039	1	5	5	38	0.385704	0.416667	0.285714	0.285714	178	0.120773	1.142793	0.5						
17	1.1 what is the to simulat	program t		5	0.987233	0.5	7	0.583333	0.6	0.666667	0.64	0.125	0.999616	1	7	3	14	0.788241	0.583333	0.571429	0.571429	15	0.705128	0.939951	0.7						
18	1.1 what is the to simulat	It provide		2	0.966665	0.16	4	0.333333	0	0.333333	0.242424	0.107143	0.003222	1	4	6	25	0.45	0.333333	0.142857	0.142857	90	0.088608	1.042929	0.4						
19	1.1 what is the to simulat	It tests th		2	0.952399	0.142857	3	0.25	0.1	0.333333	0.75862	0	0.99999	1	3	7	21	0.5	0.25	0	0	60	0.104839	1.266414	0.3						
20	1.1 what is the to simulat	to get earl		2.5	0.93749	0.111111	4	0.333333	0	0.416667	0.196078	0.181034	0.999728	1	4	6	36	0.423114	0.333333	0.142857	0.142857	168	0.074796	1.102011	0.4						
21	1.1 what is the to simulat	to simulat		5	0.962926	0.538462	7	0.583333	0.6	0.666667	0.666667	0	1	1	7	3	13	0.8125	0.583333	0.571429	0.571429	6	0.774648	1	0.7						
22	1.1 what is the to simulat	It simulat		5	0.962926	0.538462	7	0.583333	0.6	0.75	0.72	0	0.999998	1	7	3	13	0.802955	0.583333	0.571429	0.571429	5	0.787832	1.083333	0.7						
23	1.1 what is the to simulat	a prototy		1.5	0.963397	0.142857	3	0.25	0	0.25	0.14286	0.15	0.999765	1	3	7	21	0.279508	0.25	0.142857	0.142857	61	0.07438	1.210256	0.3						
24	1.1 what is the to simulat	to ease th		2.5	0.945837	0.222222	4	0.333333	0.1	0.416667	0.333333	0.019231	0.999995	1	4	6	18	0.574524	0.333333	0.142857	0.142857	70	0.112069	0.994382	0.4						
25	1.1 what is the to simulat	It simulat		5	0.952926	0.538462	7	0.583333	0.6	0.666667	0.666667	0	1	1	7	3	13	0.8125	0.583333	0.571429	0.571429	6	0.774648	1	0.7						
26	1.1 what is the to simulat	to get earl		2	0.965524	0.16	4	0.333333	0	0.333333	0.242424	0.116667	0.003159	1	4	6	25	0.45	0.333333	0.142857	0.142857	78	0.074023	1.206581	0.4						
27	1.1 what is the to simulat	the role of		3	0.934308	0.178571	5	0.416667	0	0.416667	0.243902	0.068182	0.999002	1	5	5	28	0.105027	0.416667	0.285714	0.285714	104	0.11465	1.250731	0.5						
28	1.1 what is the to simulat	to show if		4	0.95029	0.190476	4	0.333333	0.1	0.25	0.14286	0.125	0.999993	1	4	6	21	0.412479	0.333333	0.142857	0.142857	51	0.102564	1.430303	0.4						
29	1.1 what is the to simulat	prototy		2.5	0.9554	0.089966	4	0.333333	0	0.333333	0.1	0.208333	0.999991	1	4	6	58	0.241191	0.333333	0.142857	0.142857	338	0.064815	1.028718	0.4						
30	1.2 what stage the testing	refining a		3.5	0.947604	0.115385	6	0.333333	0.058824	0.333333	0.155844	0.307181	0.999989	1	6	7	52	0.457168	0.333333	0.4	0.4	268	0.141304	1.003163	0.461538						

Theoretical Derivation of Novel Statistical Functions for Keyword Extraction and Mutation

1. Information Gain Ratio Keyword Extraction (IGRKE)

The Information Gain Ratio Keyword Extraction (IGRKE) method identifies the most informative words in a corpus by evaluating their ability to distinguish between answer keys and student responses.

1.1 Entropy and Information Theory Foundation

For any word W in the corpus, its probability distribution is defined as:

- $P(W)$: Probability of word W occurring.
- $P(\neg W) = 1 - P(W)$: Probability of W not occurring.
- C : The document category (Answer Key vs. Student Response).

The total entropy of the word occurrence is given by:

$$H(W) = -P(W) * \log_2 P(W) - P(\neg W) * \log_2 P(\neg W)$$

This measures the uncertainty of the word's distribution across the dataset.

1.2 Conditional Entropy and Information Gain

The conditional entropy $H(C|W)$ represents the uncertainty in categorizing a document given that word W is known:

$$H(C|W) = -P(A|W) * \log_2 P(A|W) - P(R|W) * \log_2 P(R|W) \text{ where:}$$

- $P(A|W)$ is the probability of the document being an Answer Key given W .
- $P(R|W)$ is the probability of the document being a Student Response given W .

The Information Gain (IG) quantifies how much knowing W reduces uncertainty about the document's category:

$$IG(W) = H(C) - H(C|W) \text{ where } H(C) \text{ is the entropy of the category distribution.}$$

1.3 Normalization Using Split Information

To prevent bias toward frequent words, we normalize the Information Gain using Split Information (SI):

$$SI(W) = -P(W) * \log_2 P(W) - P(\neg W) * \log_2 P(\neg W)$$

This normalizes IG to Information Gain Ratio (IGR):

$$IGR(W) = IG(W) \div SI(W)$$

1.4 Frequency Adjustment Factor (FAF) for Balancing Word Significance

To ensure the extracted keywords are relevant across document categories, we introduce a Frequency Adjustment Factor (FAF):

$FAF = (F_answer * F_response) \div (Total_F_answer * Total_F_response)$ where:

- F_answer and $F_response$ are the word frequencies in Answer Keys and Student Responses, respectively.
- $Total_F_answer$ and $Total_F_response$ are the total word counts in both document types.

1.5 Final IGRKE Scoring Function

The final scoring function combines IGR and FAF using a logarithmic transformation:

$$Score(W) = IGR(W) * (1 + \log(1 + FAF * 1000))$$

2. Statistical Co-occurrence Mutation (SCM)

2.1 Co-occurrence Matrix Construction

A co-occurrence matrix M is built where each entry represents the number of documents where words i and j appear together:

$M[i, j]$ = count of documents where both words i and j appear

2.2 Pointwise Mutual Information (PMI) for Semantic Association

$$PMI(i, j) = \log_2 [P(i, j) \div (P(i) * P(j))]$$

2.3 Hybrid Similarity Measure for Keyword Comparison

$$Sim(K1, K2) = 0.6 \times Jaccard_Sim(K1, K2) + 0.4 \times PMI_Sim(K1, K2)$$

3. Similarity Scoring Function for Answer Matching

$$Similarity(A, S) = 0.4 \times Jaccard_Sim(A, S) + 0.4 \times Coverage(A, S) + 0.2 \times Position_Score(A, S)$$

where:

- Jaccard Similarity measures word overlap.
- Coverage measures the proportion of answer key words found.
- Position Score captures structural similarity:

$$Position_Score(A, S) = Average (1 - | Pos_A(w) \div |A| - Pos_S(w) \div |S| |)$$

Code:

IGRKE Function:

```
IGRKE <- function(word_occurrences, answer_docs, response_docs) { total_docs
<- length(answer_docs) + length(response_docs) docs_with_word

<- sum(word_occurrences > 0)

P_W <- docs_with_word / total_docs
P_not_W <- 1 - P_W H_W

<- 0 if (P_W > 0 && P_W
< 1) {

  H_W <- -P_W * log2(P_W) - P_not_W * log2(P_not_W)

}

answer_docs_with_word <- sum(word_occurrences[answer_docs] > 0) response_docs_with_word
<- sum(word_occurrences[response_docs] > 0) answer_docs_without_word <- length(answer_docs)
- answer_docs_with_word response_docs_without_word <- length(response_docs) -
response_docs_with_word

P_A_given_W <- answer_docs_with_word / docs_with_word

P_R_given_W <- response_docs_with_word / docs_with_word

P_A_given_not_W <- answer_docs_without_word / (total_docs - docs_with_word)

P_R_given_not_W <- response_docs_without_word / (total_docs - docs_with_word)

H_C_given_W <- 0 if (P_A_given_W > 0 && P_R_given_W > 0) {

  H_C_given_W_present <- -P_A_given_W * log2(P_A_given_W) - P_R_given_W *
log2(P_R_given_W)

} else {

  H_C_given_W_present <- 0

}

H_C_given_not_W <- 0 if (P_A_given_not_W > 0 &&
P_R_given_not_W > 0) {
```

```

H_C_given_not_W <- -P_A_given_not_W * log2(P_A_given_not_W) - P_R_given_not_W *
log2(P_R_given_not_W)

}

H_C_given_W <- P_W * H_C_given_W_present + P_not_W * H_C_given_not_W

P_A <- length(answer_docs) / total_docs

P_R <- length(response_docs) / total_docs

H_C <- -P_A * log2(P_A) - P_R * log2(P_R)

IG <- H_C - H_C_given_W

SI <- H_W

IGR <- if (SI > 0) IG / SI else 0

F_answer <- sum(word_occurrences[answer_docs])

F_response <- sum(word_occurrences[response_docs])

Total_F_answer <- sum(sapply(answer_docs, function(doc) sum(word_occurrences[doc])))

Total_F_response <- sum(sapply(response_docs, function(doc) sum(word_occurrences[doc])))

FAF <- (F_answer * F_response) / (Total_F_answer * Total_F_response)

Score <- IGR * (1 + log(1 + FAF * 1000)) return(Score) }

```

SCM Function:

```

SCM <- function(corpus, answer_keywords, student_keywords, threshold = 0.7) {
all_words <- unique(c(unlist(corpus))) n_words <- length(all_words) word_indices
<- setNames(1:n_words, all_words) M <- matrix(0, nrow = n_words, ncol =
n_words) for (doc in corpus) { doc_words <- unique(doc) for (i in
1:(length(doc_words) - 1)) { for (j in (i+1):length(doc_words)) { word_i <-
doc_words[i] word_j <- doc_words[j] idx_i <- word_indices[[word_i]]
idx_j <- word_indices[[word_j]]

M[idx_i, idx_j] <- M[idx_i, idx_j] + 1

M[idx_j, idx_i] <- M[idx_j, idx_i] + 1

}
}
}

```

```

    }

}

doc_count <- length(corpus) word_probs <- numeric(n_words) for (i in 1:n_words)
{ word <- all_words[i] word_probs[i] <- sum(apply(corpus, function(doc) word %in%
doc)) / doc_count

}

PMI <- matrix(0, nrow = n_words, ncol = n_words)
for (i in 1:n_words) { for (j in 1:n_words) { if (i !=
j && M[i,j] > 0) { joint_prob <- M[i,j] / doc_count
expected_prob <- word_probs[i] * word_probs[j] if
(expected_prob > 0) {

PMI[i,j] <- log2(joint_prob / expected_prob)

}

}

}

} mutation_candidates <- list() for (word in student_keywords) {
if (!(word %in% answer_keywords) && word %in% all_words) {
word_idx <- word_indices[[word]] pmi_scores <- numeric() for
(ans_word in answer_keywords) { if (ans_word %in% all_words) {
ans_idx <- word_indices[[ans_word]] pmi_scores <-
c(pmi_scores, PMI[word_idx, ans_idx])

}

}

avg_pmi <- mean(pmi_scores) position <- match(word, student_keywords) /
length(student_keywords) position_weight <- 1 - (0.5 * position) score <- avg_pmi *
position_weight max_pmi <- max(pmi_scores) normalized_max_pmi <- (max_pmi + 10) /
20 uniqueness <- 1 - normalized_max_pmi if (uniqueness >= threshold) {
mutation_candidates[[word]] <- list(word = word, score = score, uniqueness = uniqueness)

}

```



```

    }

}

sorted_candidates <- mutation_candidates[order(sapply(mutation_candidates, function(x) x$score),
decreasing = TRUE)]

jaccard_sim <- length(intersect(answer_keywords, student_keywords)) /
length(union(answer_keywords, student_keywords)) pmi_scores
<- numeric() for (ans_word in answer_keywords)
{ if (ans_word %in% all_words) { ans_idx <-
word_indices[[ans_word]] for (stud_word in student_keywords)
{ if (stud_word %in% all_words) { stud_idx <-
word_indices[[stud_word]] pmi_scores
<- c(pmi_scores, PMI[ans_idx, stud_idx])

}

}

}

pmi_sim <- (mean(pmi_scores) + 10) / 20 hybrid_sim <- (0.6 * jaccard_sim)
+ (0.4 * pmi_sim) matched_keywords <- intersect(answer_keywords,
student_keywords) position_scores <- numeric() for (word in
matched_keywords) { pos_a <- match(word, answer_keywords) /
length(answer_keywords) pos_s <- match(word, student_keywords) /
length(student_keywords) position_scores <- c(position_scores, 1 -
abs(pos_a - pos_s))

}

position_score <- if (length(position_scores) > 0) mean(position_scores) else 0 coverage
<- length(matched_keywords) / length(answer_keywords) similarity_score <- (0.4 *
jaccard_sim) + (0.4 * coverage) + (0.2 * position_score) return(list( mutation_candidates
= sorted_candidates, similarity_score = similarity_score, jaccard_similarity =
jaccard_sim, coverage = coverage, position_score = position_score

))

```

```
}
```

Similarity Scoring Function:

```
calculate_similarity <- function(answer_keywords, student_keywords) {  jaccard_sim <-  
length(intersect(answer_keywords, student_keywords)) / length(union(answer_keywords,  
student_keywords))  coverage <- length(intersect(answer_keywords, student_keywords)) /  
length(answer_keywords)  matched_keywords <- intersect(answer_keywords,  
student_keywords)  position_scores <- numeric()  for (word in matched_keywords) {  
pos_a <- match(word, answer_keywords) / length(answer_keywords)  pos_s <- match(word,  
student_keywords) / length(student_keywords)  position_scores <- c(position_scores, 1 -  
abs(pos_a - pos_s))  
}  
position_score <- if (length(position_scores) > 0) mean(position_scores) else 0  similarity_score  
<- (0.4 * jaccard_sim) + (0.4 * coverage) + (0.2 * position_score)  return(similarity_score)  
}
```

Result:

This novel framework integrates:

- **IGRKE for statistical keyword extraction** using information theory and frequency normalization.
- **SCM for identifying semantically related keywords** based on co-occurrence and PMI.
- **A similarity function** that combines keyword overlap, coverage, and positional alignment.

By relying purely on statistical principles without pre-trained models, this framework ensures robustness and adaptability across domains.