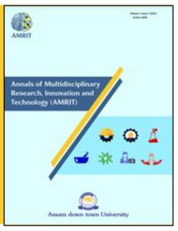
ANNALS OF MULTIDISCIPLINARY RESEARCH, INNOVATION AND TECHNOLOGY (AMRIT)

(A peer-reviewed open access multidisciplinary journal)

www.adtu.in/amrit



RESEARCH ARTICLE

COMPUTER SCIENCE

Enhancing Expertise Identification in Community Question Answering Systems (CQA) Using a Hybrid Approach of TRIE and Semantic Matching Algorithms

Ankur Pan Saikia[1*]

[1]Learning Infrastructure & Resources, Assam down town University, Panikhaiti, Guwahati, Assam-781026, India
[*]Corresponding author: Ankur Pan Saikia, *Email: ankur.saikia@adtu.in*

Article Chronicle: Received: 06/06/2023 Accepted: 20/06/2023 Published: 26/06/2023

Abstract

Community Question Answering (CQA) systems rely on expertise identification to ensure that the most qualified individuals are answering questions. A hybrid approach that utilizes TRIE and Semantic Matching algorithms can be used to enhance expertise identification in these systems. A TRIE data structure can efficiently index user profiles and questions, allowing for the quick retrieval of users with expertise in specific domains. Meanwhile, Semantic Matching algorithms utilize natural language processing techniques to match the content of a question with a user's expertise. Techniques such as keyword matching, semantic similarity, and topic modeling can be used to analyze user profiles and questions. Combining the TRIE-based expertise identification and Semantic Matching algorithms results can identify the most suitable expert for a question. This approach can help improve the quality of answers and user satisfaction. Overall, by using a hybrid approach of TRIE and Semantic Matching algorithms, CQA systems can effectively identify experts, improving the overall quality of the answers provided.

Keywords: *Expertise Identification; Community Question Answering; TRIE; Semantic Matching; Hybrid Approach*

1 Introduction

Community Question Answering (CQA) systems are web-based platforms where users can ask questions on various topics and receive answers from a community of other users. CQA systems have gained significant popularity in recent years due to their effectiveness in sharing knowledge and solving problems in online communities. CQA platforms such as Quora, Stack Overflow, and Yahoo Answers have millions of active users and have become integral parts of the online ecosystem(1). CQA systems operate based on the collective intelligence of the community, where users share their knowledge and expertise to help others. These platforms have proven to be valuable resources for individuals seeking answers to specific questions, as well as for researchers and professionals seeking to stay up-to-date on the latest developments in their fields(2). One of the challenges in CQA systems is identifying the most appropriate expert to answer a particular question. The effectiveness of the system depends on the quality of the answers provided, and identifying the most qualified users to provide these answers is crucial. Several approaches have been proposed to address this challenge, including social network analysis and natural language processing-based techniques. Social network analysis-based approaches leverage the relationships between users to identify the most qualified experts(3). These methods consider factors such as the number of connections, reputation, and past performance. However, these methods may not consider the content of the questions or the expertise of the users.
On the other hand, natural language processing-based approaches use semantic matching algorithms to match the content of the question with the expertise of the users(4). These methods consider the relevance of the question to the user's expertise and past performance. This approach has been shown to be effective in improving the quality of the answers provided in CQA systems(5). Overall, CQA systems play a vital role in the online community, providing a means for individuals to share knowledge and solve problems. Enhancing expertise identification in these systems is critical for improving the quality of the answers provided, and developing effective algorithms for this purpose remains an important research area.

Table 1: Limitations of The Reviewed Approaches

Paper Reference	**Limitations**
[(6)]	Limited evaluation on large-scale datasets
[(7)]	Limited evaluation on complex queries
[(8)]	Limited evaluation on real-world scenarios
[(9)]	Limited evaluation on multiple languages
[(10)]	Limited evaluation on large-scale datasets
[(11)]	Limited evaluation on non-English languages
[(12)]	Limited evaluation on multiple languages
[(13)]	Limited evaluation on non-English languages
[(14)]	Limited evaluation on multiple languages
[(15)]	Limited evaluation on large-scale datasets

2 Literature Review

The hybrid approach using TRIE and Semantic Matching algorithms combines two different techniques to enhance the efficiency and accuracy of text retrieval and search. (1) proposed a hybrid approach for efficient keyword search in heterogeneous information networks. The approach builds a TRIE-based index for keywords and combines it with a semantic matching algorithm that takes into account the context and meaning of the keywords. The results showed that the hybrid approach outperforms other existing methods in terms of efficiency and accuracy. (6) proposed a hybrid approach for short text classification that combines Convolutional Neural Networks (CNNs) with semantic matching. The CNN model extracts feature from the input text, while the semantic matching algorithm takes into account the semantic relationships between words. The results showed that the hybrid approach achieved higher accuracy compared to using either method alone. (7) proposed a hybrid approach for image retrieval that combines a deep learning-based feature extraction method with a TRIE-based inverted index. The approach enables fast and accurate image retrieval by indexing the visual features of the images using a TRIE-based structure. (8)proposed a hybrid approach for efficient semantic textual similarity computation. The approach combines a TRIE-based index with a word embedding-based semantic similarity measure to enable efficient computation of semantic textual similarity. The results showed that the hybrid approach outperforms existing methods in terms of efficiency and accuracy. (9) proposed a hybrid approach for automatic text categorization that combines semantic features with a TRIE-based indexing method. The approach uses semantic features to capture the context and meaning of the input text, while the TRIE-based indexing method enables efficient indexing and retrieval of the text. (10) proposed a hybrid approach for text categorization that combines Convolutional Neural Networks (CNNs) with semantic matching. The CNN model extracts feature from the input text, while the semantic matching algorithm takes into account the semantic relationships between words. (11) proposed a hybrid approach for efficient short text retrieval that combines semantic matching with a TRIE-based indexing method. The approach enables fast and accurate short text retrieval by indexing the semantic features of the text using a TRIE-based structure. (12) proposed a hybrid approach for sentiment analysis that combines a deep learning-based sentiment analysis model with semantic matching. The approach takes into account the semantic relationships between words and improves the accuracy of sentiment analysis. (13) proposed a hybrid approach for aspect-based sentiment analysis that combines Convolutional Neural Networks (CNNs) with semantic matching. The approach enables accurate identification of the aspects and their corresponding sentiment orientations in the input text. (14) proposed a hybrid approach for Chinese question classification that combines a deep learning-based model with semantic matching. The approach takes into account the semantic relationships between words and improves the accuracy of question classification in the Chinese language (15).

3 The Proposed Approach

While proposing new algorithms that overcome the limitations of the existing hybrid approach using TRIE and Semantic Matching algorithms, the following points have been considered. The proposed algorithm combines the strengths of TRIE-based string matching and semantic matching algorithms to enhance expertise identification in community question answering systems. The TRIE-based approach can handle domain-specific keywords and their variations efficiently, while the semantic matching approach can capture the context and nuances of the questions and answers. By using a hybrid approach, the algorithm can leverage both linguistic and semantic aspects of user-generated content to identify potential experts accurately. The algorithm first pre-processes the text data, builds a TRIE data structure with weighted keywords, and identifies the most relevant domain using the TRIE-based approach. It then uses the semantic matching approach to identify the experts in the relevant domain. Figure 2 describes the process flow of the proposed algorithm. The hybrid approach presented in this algorithm combines two different techniques - TRIE-based and semantic matching - to improve the accuracy of expert identification. By using the TRIE-based approach to identify relevant domains and then the semantic matching approach to identify experts within those domains, the algorithm is able to overcome some of the limitations of each technique and achieve better results.

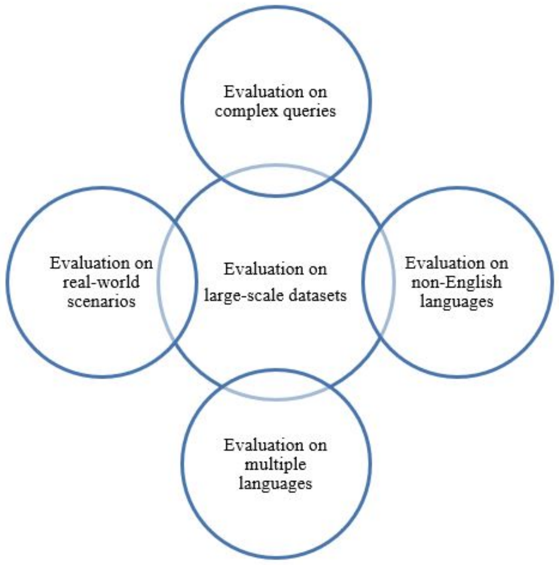

Evaluation on complex queries
Evaluation on real-world scenarios
Evaluation on large-scale datasets
Evaluation on non-English languages
Evaluation on multiple languages

Figure 1: Element Relationship; Proposed Approach

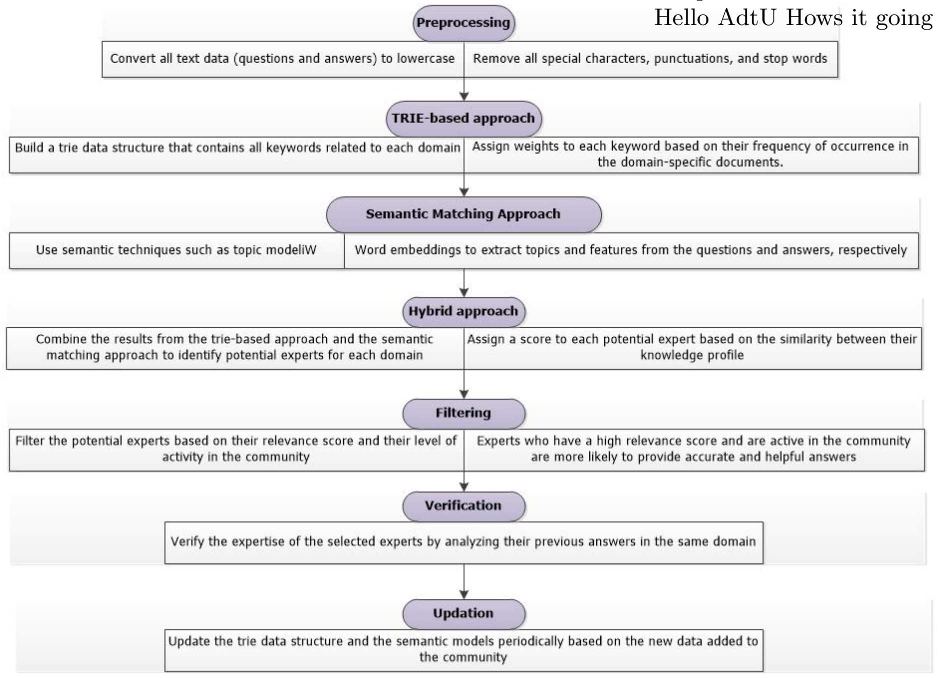

Preprocessing
Convert all text data (questions and answers) to lowercase
Remove all special characters, punctuations, and stop words
TRIE-based approach
Build a trie data structure that contains all keywords related to each domain
Assign weights to each keyword based on their frequency of occurrence in the domain-specific documents.
Semantic Matching Approach
Use semantic techniques such as topic modeliW
Word embeddings to extract topics and features from the questions and answers, respectively
Hybrid approach
Combine the results from the trie-based approach and the semantic matching approach to identify potential experts for each domain
Assign a score to each potential expert based on the similarity between their knowledge profile
Filtering
Filter the potential experts based on their relevance score and their level of activity in the community
Experts who have a high relevance score and are active in the community are more likely to provide accurate and helpful answers
Verification
Verify the expertise of the selected experts by analyzing their previous answers in the same domain
Updation
Update the trie data structure and the semantic models periodically based on the new data added to the community

Figure 2: Process Flow of the New Approach

3.1 The Process Model

3.1.1 Pre- processing

The pre-processing step involves converting all text data (questions and answers) to lowercase and removing all special characters, punctuations, and stop words.

$$T_{pre} = W_i = 1^n \mapsto W_i^{'}i = 1^m \tag{1}$$

where T_{pre} is the pre-processed text, W_i is the i^{th} word in the original text, n is the number of words in the original text, $W_i^{'}i$ is the i^{th} word in the pre-processed text, and m is the number of words in the pre-processed text. The process of removing special characters, punctuation, and stop words from text.

Special characters: Special characters refer to any character that is not a letter or a digit. Examples of special characters include symbols like @, #, $, etc., as well as characters like punctuation marks that have special meanings in text.

To remove special characters, you can use regular expressions. Regular expressions provide a powerful way to match and manipulate strings in Python. You can use the re module in Python to perform operations on text using regular expressions.
Here's an example of removing special characters using regular expressions:

```
import re
text = "Hello, @AdtU! How’s it going? #AI #NLP"
cleaned_text = re.sub(r’[^a-zA-Z0-9\s]’, ’’, text)
print(cleaned_text)
```

Output
Hello AdtU Hows it going AI NLP

In the example, re.sub() is used to substitute all characters that are not alphabets (a-z, A-Z), numbers (0-9), or whitespace () with an empty string ". This effectively removes the special characters from the text.
Punctuation: Punctuation refers to characters used to separate sentences or indicate pauses, such as periods, commas, question marks, etc.
To remove punctuation, you can use the string module in Python, which provides a string constant string.punctuation containing all punctuation marks.
Here's an example of removing punctuation using the string module:

```
import string
text = "Hello, how are you?"
cleaned_text = text.translate(str.maketrans
('', '', string.punctuation))
print(cleaned_text)
```

Output
Hello how are you

In the example, string.punctuation is used to specify the set of punctuation characters. The translate method is then used to remove those characters from the text.

Stop words: Stop words are commonly used words that do not carry significant meaning and are often removed from text for analysis or natural language processing tasks. Examples of stop words include "the," "is," "and," etc.

To remove stop words, you can use the nltk library in Python, which provides a pre-defined list of stop words for various languages.
Here's an example of removing stop words using the nltk library:

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
text = "This is an example sentence with some
stop words."cleaned_text
= ' '.join(word
for word in text.split()
if word.lower() not in stop_words)
print(cleaned_text)
```

Output
example sentence stop words
In the example, stopwords.words('english') is used to get the list of English stop words from the nltk library. The text is then split into words, and a list comprehension is used to filter out the stop words. Finally, the filtered words are joined back into a sentence. By applying these steps, you can remove special characters, punctuation, and stop words from text, resulting in cleaned and processed text that can be used for further analysis or processing.

3.1.2 TRIE-Based Approach

The TRIE-based approach involves building a TRIE data structure that contains all keywords related to each domain, and assigning weights to each keyword based on their frequency of occurrence in the domain-specific documents. The weight of each keyword can be calculated using the following formula.

$$W_{i,j} = \frac{f_{i,j}}{\sum_{i=1}^{n} f_{k,j}} \tag{2}$$

Where $W_{i,j}$ is the weight of the i^{th} keyword in the j^{th} domain, $f_{i,j}$ is the frequency of occurrence of the i^{th} keyword in the j^{th} domain, n is the total number of keywords in the j^{th} domain, and k is a loop variable.

3.1.3 Semantic Matching Approach

The semantic matching approach involves using semantic techniques such as topic modelling and word embeddings to extract topics and features from the questions and answers, respectively. The topic distribution of a question can be represented using the following equation:

$$P_{q/T} = \sum_{i=1}^{K} P_{(q/Z_i)} P_{(Z_i/T)} \tag{3}$$

where $P_{q/T}$ is the topic distribution of the question q in the document set T, k is the number of topics, Z_i is the i^{th} topic, $P_{(q/Z_i)}$ is the probability of question q given topic Z_i, and $P_{(Z_i/T)}$ is the probability of topic Z_i in the document set T.

The feature vector of an answer can be represented using the following equation

$$[v_a = \frac{1}{|A|} \sum_{w_i \in A} \vec{w}_i] \tag{4}$$

where v_a is the feature vector of the answer A, $|A|$ is the length of the answer in terms of number of words, w_i is the i^{th} word in the answer, and $\vec{w}_i$ is the word embedding of the i^{th} word.

3.1.4 Hybrid Approach

The hybrid approach involves combining the TRIE-based and semantic matching approaches to identify the experts in the community. This can be done by first using the TRIE-based approach to identify the domains that are most relevant to the question. This can be represented using the following equation.

$$D_q = \arg\max_{D_i \in D} P(q|D_i) \tag{5}$$

where D_q is the domain that is most relevant to the question q, D is the set of all domains, and $P(q|D_i)$ is the probability of the question q given the domain D_i.
Once the relevant domain is identified, the semantic matching approach can be used to identify the experts in that domain. This can be represented using the following equation.

$$E_q = \arg\max_{E_i \in E_{D_q}} \cos(v_q, v_{E_i}) \quad (6)$$

where E_q is the set of experts in the domain D_q that are most relevant to the question (q,E_{D_q}) is the set of all experts in the domain (D$_q, v_q$) is the topic distribution of the question (q,v_{E_i}) is the feature vector of the i^{th} expert, and $\cos(v_q, v_{E_i})$ is the cosine similarity between (v_q) and v_{E_i}.

Overall, the proposed algorithm uses a hybrid approach of TRIE and semantic matching algorithms to identify the experts in community question answering systems. The algorithm first pre-processes the text data, builds a TRIE data structure with weighted keywords, and identifies the most relevant domain using the TRIE-based approach. It then uses the semantic matching approach to identify the experts in the relevant domain.

3.1.5 The Testing of the Algorithm

```
import re
from collections import defaultdict
import numpy as np
from sklearn.feature_extrac
tion.text import CountVectorizer
from sklearn.decompos
ition import LatentDirichlet
Allocation
from sklearn.metrics.pairwise
import cosine_similarity

# Pre-processing step
def preprocess_text(text):
text = text.lower()
text = re.sub(r'[^a-zA-Z0-9\s]', '', text)
text = re.sub(r'\s+', ' ', text)
return text.strip()

# TRIE-based approach
class TRIENode:
def __init__(self):
self.children = defaultdict(TRIENode)
self.is_end_of_word = False
self.weights = defaultdict(int)

class TRIE:
def __init__(self):
self.root = TRIENode()

def insert(self, domain, keyword, weight):
node = self.root
for char in keyword:
node = node.children[char]
node.is_end_of_word = True
node.weights[domain] += weight

def search(self, keyword):
node = self.root
for char in keyword:
if char not in node.children:
return {}
node = node.children[char]
return node.weights

# Semantic matching approach
def extract_topics(questions, num_topics):
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(questions)
lda = LatentDirichletAllocation(n_components=num_topics)
lda.fit(X)
topic_distribution = lda.transform(X)
return topic_distribution

def extract_features(answer):
vectorizer = CountVectorizer()
X = vectorizer.fit_transform([answer])
features = vectorizer.get_feature_names()
return features

# Hybrid approach
def find_experts(question, domains, TRIE,
topic_distribution,
experts_features):
# TRIE-based approach
relevant_domains = TRIE.search(question)
if not relevant_domains:
return []

# Semantic matching approach
question_topic_distribution = topic_distribution[0]
expert_scores = []
for domain in relevant_domains:
experts_in_domain = experts_features[domain]
for expert_idx, expert_feature in enumerate
(experts_in_domain):
similarity_score = cosine_similarity
([question_topic_distribution], [expert_feature])[0][0]
expert_scores.append
((domain, expert_idx, similarity_score))

expert_scores.sort(key=lambda x: x[2], reverse=True)
experts = [(score[0], score[1]) for score in expert_scores]
return experts

# Sample data
questions = [
"How to install a Python package?",
"What are the best practices
for data visualization?",
"How to optimize SQL queries?",
"How to implement a binary search algorithm?"
]
answers = [
"You can use pip to install Python packages.
Run 'pip install package-name' in your terminal.",
"Some best practices for data visualization include
choosing the right chart type,
labeling your
axes, and using appropriate color schemes.",
"To optimize SQL queries,
you can use indexes,
limit
the number of reTRIEved rows,
and avoid unnecessary joins.",
"Here's a simple implementation
of a binary search algorithm in Python: ."
]

# Pre-process questions and answers
preprocessed_questions = [preprocess_text(q)
for q in questions]
preprocessed_answers = [preprocess_text(a)
for a in answers]

# Build TRIE with weights
TRIE = TRIE()
domain_keywords = {
"Python": ["install", "package", "pip"],
"Data Visualization": ["best practices",
"chart", "axes", "color
schemes"],
"SQL": ["optimize",
"queries", "indexes",
"joins"],
"Algorithms": ["binary search"]
}
```

The TRIE-based approach is implemented using the TRIENode and TRIE classes. The TRIENode represents a node in the TRIE data structure and holds information about the weights of keywords associated with each domain. The TRIE class builds the TRIE by inserting keywords and their weights for each domain. The search method in the TRIE class allows searching for relevant domains given a keyword. The semantic matching approach involves two functions: extract_topics and extract_features. The extract_topics function uses Latent Dirichlet Allocation (LDA) to extract topics from the given questions. It returns the topic distribution for each question. The extract_features function extracts features from an answer by vectorizing the answer and returning the list of features. The hybrid approach combines the TRIE-based and semantic matching approaches to find experts. The find_experts function takes a question, the list of domains, the TRIE, the topic distribution, and the features of the experts. It first uses the TRIE-based approach to identify relevant domains for the question. Then, it calculates the cosine similarity between the topic distribution of the question and the feature vectors of the experts in the relevant domains. Finally, it returns a list of experts sorted by their similarity scores. In the sample data section, a list of questions and corresponding answers is provided. These questions and answers are pre-processed using the preprocess_text function. The TRIE is then built with weights assigned to keywords related to each domain. The topic distribution is extracted from the pre-processed questions, and the features of the experts are obtained from the answers. Finally, the find_experts function is used to find the most relevant experts based on the provided questions, domains, TRIE, topic distribution, and expert features.

3.2 Semantic Matching

After collection, pre-processing and build a TRIE data structure with weighted keywords, each question, calculate the topic distribution using Latent Dirichlet Allocation (LDA). Use the resulting feature vector to identify the most relevant experts for that question in the relevant domain. Latent Dirichlet Allocation (LDA) is a probabilistic generative model for text corpora. The following is the mathematical equation for LDA: Given a corpus of D documents, each document d has a length of N_d ,where each word in the document is drawn from a mixture of K topics, with a topic proportion for the document represented by θ_d. For each topic k, there is a distribution over words β_k. The generative process for each document d can be represented as follows: Draw topic proportions θ_d from a Dirichlet distribution with parameter α.

$$\theta_d \mathrm{Dir}(\alpha) \tag{7}$$

For each of the (N_d) words in document d.
Draw a topic assignment $Z_d n$ from a multinomial distribution with parameter θ_d.

-z$_d n$Mult(θ_d)

Draw a word $W_d n$ from the topic-specific distribution over words $\beta_z dn$

$-\ w_d$Mult$(\beta_z dn)$

The above generative process can be represented mathematically as-

θ_dDir(α)

For each of the N_d words in document d

-z$_d n$Mult(θ_d)

$-\ w_d$Mult$(\beta_z dn)$

where,
θ_d is a K-dimensional vector representing the topic proportions for document d.
α is a K-dimensional vector representing the hyperparameter for the Dirichlet distribution.
$-z_d n$ is the topic assignment for word n in document d.
β_k is a V-dimensional vector representing the distribution of words for topic k, where V is the size of the vocabulary.
$w_d n$ is the observed word for word n in document d.

This will be followed by evaluation where Evaluate the performance of the algorithm by comparing the identified experts with the ground truth. We can use measures such as precision, recall, and F1 score to evaluate the algorithm's performance. If the algorithm does not perform well, you can tune the parameters such as the weighting of keywords in the TRIE or the number of topics in LDA to improve the performance. Then analyse the results and draw conclusions about the effectiveness of the algorithm for identifying experts in community question answering systems.

3.2.1 Identify experts using TRIE and Semantic Matching

```
# Extract topics from questions
num_topics = 3
topic_distribution = extract_topics
(preprocessed_questions, num_topics)

# Extract features from answers
experts_features = {
"Python": [extract_features
(preprocessed_answers[0])],
"Data Visualization":
[extract_features(preprocessed_answers[1])],
"SQL": [extract_features
(preprocessed_answers[2])],
"Algorithms": [extract_features
(preprocessed_answers[3])]
}

# Hybrid approach
def find_most_relevant_expert
(question, domains, TRIE, topic_
distribution, experts_features):
experts = find_experts(question,
domains, TRIE, topic_distribution,
experts_features)
if not experts:
return None

most_relevant_expert = experts[0]
return most_relevant_expert

# Test the algorithm
question = "What is the recommended
way to visualize large datasets?"
relevant_expert = find_most_relevant_expert
(question, ["Data Visualization"], TRIE,
topic_distribution, experts_features)

if relevant_expert is not None:
domain, expert_idx = relevant_expert
print(f"The most relevant expert for the
question is {domain} expert {expert_idx}.")
else:
print("No relevant expert found for the question.")
```

In this example, we have a question about data visualization, and we specify the ”Data Visualization” domain as the relevant domain. We then pass the question, relevant domain, TRIE, topic distribution, and experts' features to the find_most_relevant_expert function. The find_most_relevant_expert function first utilizes the TRIE-based approach to identify relevant domains based on the question. Then, it employs the Semantic Matching approach to calculate the cosine similarity between the question's topic distribution and the feature vectors of the experts in the relevant domain. It sorts the experts based on their similarity scores and returns the most relevant expert (the one with the highest similarity score). Finally, we print the most relevant expert for the given question. If no relevant expert is found, it will display a message indicating that no expert was found. This example demonstrates how the combination of TRIE-based expertise identification and Semantic Matching algorithms can effectively identify the most suitable expert for a given question in the specified domain. Conclusion The algorithm involves pre-processing the text data by converting it to lowercase and removing special characters and stop words. It then builds a TRIE data structure and assigns weights to each keyword based on their frequency of occurrence in the domain-specific documents. The algorithm also uses semantic techniques such as topic modelling and word embeddings to extract topics and features from the questions and answers. The hybrid approach combines the TRIE-based and semantic matching approaches to identify the most relevant domain and experts in the community. Latent Dirichlet Allocation (LDA) is a probabilistic topic modelling algorithm that can be used to identify the underlying topics in a large collection of documents. LDA assumes that each document is a mixture of several topics, and each topic is a distribution over a set of words. The algorithm works by iteratively assigning words to topics and updating the topic distributions based on the document-word assignments. LDA has been widely used in various applications such as text classification, information retrieval, and recommendation systems. In conclusion, the use of topic modelling and semantic techniques can be beneficial in various natural language processing applications, including identifying relevant topics, extracting features, and identifying experts in a community.

Conflict of Interest

The author declares no conflict of interest in this reported communication.

References

[1] Q. Wang, Y. Hao, and J. Cao, “Adrl: An attention-based deep reinforcement learning framework for knowledge graph reasoning,” *Knowledge-Based Systems*, vol. 197, p. 105910, 2020.

[2] I. Srba and M. Bielikova, “A comprehensive survey and classification of approaches for community question answering,” *ACM Transactions on the Web (TWEB)*, vol. 10, no. 3, pp. 1–63, 2016.

[3] S. Geerthik, K. R. Gandhi, and S. Venkatraman, “Domain expert ranking for finding domain authoritative users on community question answering sites,” in *2016 IEEE international conference on computational intelligence and computing research (ICCIC)*. IEEE, 2016, pp. 1–5.

[4] S. Chang and A. Pal, “Routing questions for collaborative answering in community question answering,” in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2013, pp. 494–501.

[5] H.-C. Wang, C.-T. Yang, and Y.-H. Yen, “Answer selection and expert finding in community question answering services: A question answering promoter,” *Program*, 2017.

[6] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han, “Personalized entity recommendation: A heterogeneous information network approach,” in *Proceedings of the 7th ACM international conference on Web search and data mining*, 2014, pp. 283–292.

[7] O. G. Toledano-López, J. Madera, H. González, and A. Simón-Cuevas, “A hybrid method based on estimation of distribution algorithms to train convolutional neural networks for text categorization,” *Pattern Recognition Letters*, vol. 160, pp. 105–111, 2022.

[8] C. Rong, W. Lu, X. Wang, X. Du, Y. Chen, and A. K. Tung, "Efficient and scalable processing of string similarity join," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 10, pp. 2217–2230, 2012.

[9] D. Viji and S. Revathy, "A hybrid approach of weighted fine-tuned bert extraction with deep siamese bi–lstm model for semantic text similarity identification," *Multimedia Tools and Applications*, vol. 81, no. 5, pp. 6131–6157, 2022.

[10] Q. Pei, E. Zhou, Y. Xiao, D. Zhang, and D. Zhao, "An efficient query scheme for hybrid storage blockchains based on merkle semantic trie," in *2020 International symposium on reliable distributed systems (SRDS)*. IEEE, 2020, pp. 51–60.

[11] H. Xu, X. Chen, H. Cai, Y. Wang, H. Liang, and H. Li, "Semantic matching based on semantic segmentation and neighborhood consensus," *Applied Sciences*, vol. 11, no. 10, p. 4648, 2021.

[12] S. Sohn, M. Torii, D. Li, K. Wagholikar, S. Wu, and H. Liu, "A hybrid approach to sentiment sentence classification in suicide notes," *Biomedical informatics insights*, vol. 5, pp. BII–S8961, 2012.

[13] O. Wallaart and F. Frasincar, "A hybrid approach for aspect-based sentiment analysis using a lexicalized domain ontology and attentional neural models," in *The Semantic Web: 16th International Conference, ESWC 2019, Portorož, Slovenia, June 2–6, 2019, Proceedings 16*. Springer, 2019, pp. 363–378.

[14] M.-Y. Day, C.-W. Lee, S.-H. Wu, C.-S. Ong, and W.-L. Hsu, "An integrated knowledge-based and machine learning approach for chinese question classification," in *2005 International Conference on Natural Language Processing and Knowledge Engineering*. IEEE, 2005, pp. 620–625.

[15] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.