Community-based Question Answering (CQA)

A PROJECT REPORT

Submitted by

SAURABH PANDEY [Reg No:RA2011003010207]

Under the Guidance of

Dr. C.N. Subalalitha

Associate Professor, Department of Computing Technologies

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

DEPARTMENT OF COMPUTING TECHNOLOGIES

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603203

NOV 202

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203
BONAFIDE CERTIFICATE

Certified that project report titled **Community-based Question Answering (CQA)** is the bonafide work of **SAURABH PANDEY [RegNo:RA2011003010207]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Dr. C.N. Subalalitha	**Dr. M. Pushpalatha**
Assistant Professor Department of Computing Technologies	**HEAD OF THE DEPARTMENT** Department of Computing Technologies

ABSTRACT

Community-based Question Answering (CQA) websites such as Quora, Yahoo! Answers, and Stack Exchange have gained significant popularity in recent years as an alternative to traditional web search for providing answers that are subjective, open-ended, and with expert opinions. These websites hold a set of sub forums that focus on a particular topic or theme. However, as the number of questions on these websites grows, duplicate questions often arise, causing confusion and lowering the quality of the user experience. The manual methods currently used to detect duplicate questions are time-consuming and subjective, leading to inconsistencies in decision-making and significant workload on human moderators.

This research proposes a system for detecting duplicate questions using Artificial Intelligence (AI) and Machine Learning (ML) algorithms. The proposed system aims to improve the efficiency, accuracy, customizability, user experience, and cost and time saving compared to the existing manual methods. The system works by analyzing a large number of questions using natural language processing and machine learning algorithms to identify patterns and similarities among the questions. The system then flags questions that have similar content or are duplicates, notifying moderators or administrators to take action.

The proposed system has several advantages over the existing manual methods. First, the system is more efficient, as it can handle a large volume of questions in a shorter timeframe, reducing the workload on human moderators. Second, the system is more accurate, as it uses AI and ML algorithms to provide an objective approach to detecting duplicate questions, avoiding the subjective nature of human moderators. Third, the system is more customizable, as it can be tailored to fit different platforms, themes, and languages. Fourth, the system enhances the user experience by providing more accurate and relevant answers, leading to increased user engagement and loyalty. Finally, the system saves time and resources for the platform, resulting in significant cost savings.

In conclusion, the proposed system for detecting duplicate questions using AI and ML has the potential to significantly improve the quality of the user experience on CQA websites, while also reducing the workload on human moderators and saving time and resources for the platform.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATION

Abbreviation	**Full Form**
CQA	Community-based Question Answering
PCQA	Programming Community-based Question Answering

CHAPTER1

INTRODUCTION

1.1 Introduction

There has been an increase in the popularity of Community-based Question Answering (CQA) websites such as Quora, Yahoo! Answers, and Stack Exchange on the Internet. The reason is that CQA websites are becoming a promising alternative to traditional web search to provide an- swers that are subjective, open-ended, and with expert opinions. To cater to the multitude of in- terests in its community, many CQA websites hold a set of subforums that focus on a particular topic or theme. For example, Stack Exchange spans different themes like Science (“Mathemat- ics,” “Physics”), Life (“Home Improvement,” “Travel”) and Technology (“Stack Overflow,” “Ask Ubuntu”).

Stack Overflow, a Programming Community-based Question Answering (PCQA) site, is a sub-domain in Stack Exchange created for programming-related questions. Despite detailed guidelines on posting ethics, a large number of created questions are poor in quality [13]. Duplicate questions—questions that were previously created and answered—are a frequent occurrence even though users are reminded to search the forum before creating a new post. To reduce the number of duplicate questions, Stack Overflow encourages reputable users to manually mark duplicate questions. Figure 1 illustrates the Stack Overflow workflow for marking duplicate questions, where enough votes from reputable users are required. This approach is laborious, but, more seriously, a large number of duplicate questions remain undetected for long time. As reported in Ahasanuzzaman et al. [1], more than 65% of duplicate questions take at least one day to be closed, and a large proportion of duplicate questions are closed more than one year later. Therefore, a high-quality automatic duplicate detection system is required to considerably improve user experience: For inexperienced users creating a new question, it can suggest a related post before posting and possibly retrieve answers immediately; for experienced users, it can suggest potential duplicate posts for manual verification.

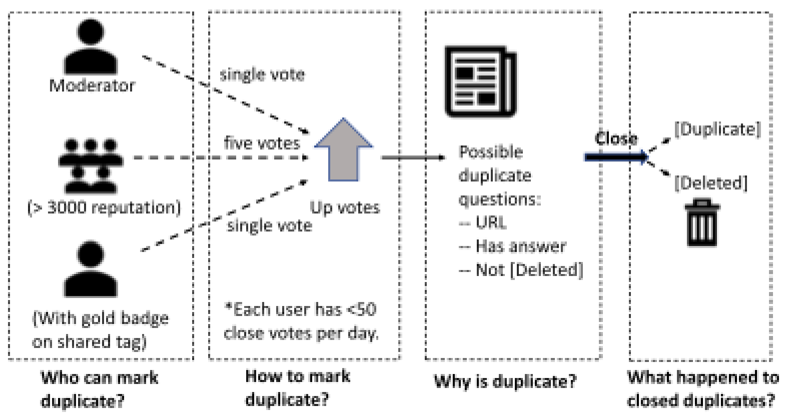
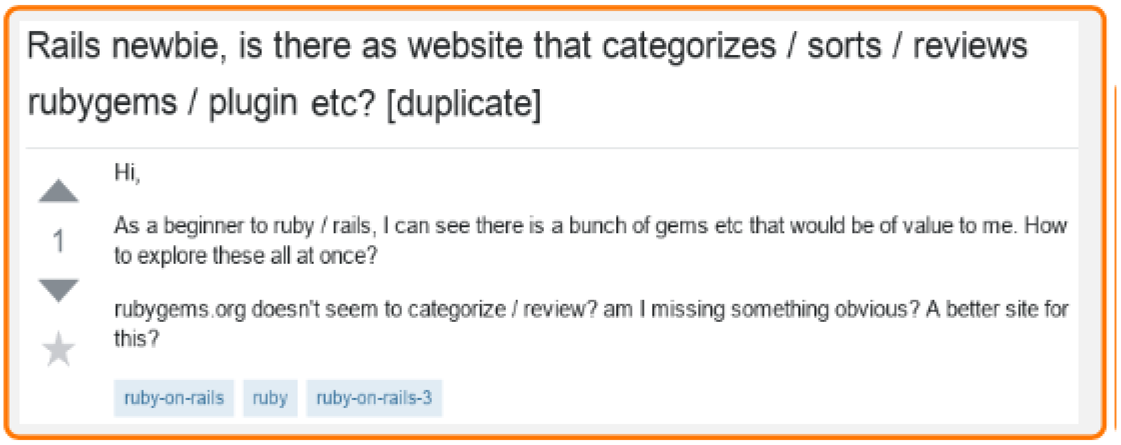


Fig. 1. Stack overflow duplicate detection workflow.

1.2 Problem Statement

Quora and Stack Exchange are platforms where users can ask questions and receive high-quality answers, but duplicate questions can lead to inefficiencies. To improve user experience, companies can identify and remove duplicate questions. For example, questions with similar intent but different wording, such as "Is talent nurture or nature?" and "Are people talented by birth or can it be developed?", can be identified as duplicates.



Rails newbie, is there as website that categorizes / sorts / reviews rubygems / plugin etc? [duplicate]

1

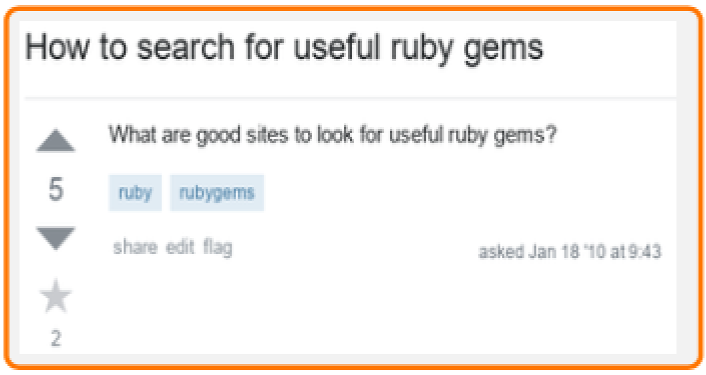
Hi,

As a beginner to ruby / rails, I can see there is a bunch of gems etc that would be of value to me. How to explore these all at once?

rubygems.org doesn't seem to categorize / review? am I missing something obvious? A better site for this?

ruby-on-rails ruby ruby-on-rails-3

(a)



How to search for useful ruby gems

5

What are good sites to look for useful ruby gems?

ruby rubygems

share edit flag

asked Jan 18 '10 at 9:43

2

(b)

Fig. 1.2 Stack Overflow is facing a growing number of duplicated questions. Here, question (a) was appointed as duplicate of question (b). However, their textual features are completely different, challenging automated solutions.

1.3 Objective

The objective of this research paper is to propose a system for detecting duplicate questions in Community-based Question Answering (CQA) websites using Artificial Intelligence (AI) and Machine Learning (ML) techniques. The system aims to improve the quality of the question-answering process by identifying and filtering out duplicate questions, which can cause redundancy and inefficiency in the CQA platform.

The proposed system will use Natural Language Processing (NLP) techniques to extract relevant features from the questions posted on the CQA platform. These features will be used to build a model using ML algorithms that can predict whether a new question is a duplicate of an existing question or not. The system will also employ deep learning techniques to further enhance the accuracy of the model.

The effectiveness of the proposed system will be evaluated by comparing its performance with existing duplicate detection methods in terms of precision, recall, and F1-score. The research will also investigate the impact of different parameters on the performance of the system, such as the size of the training dataset and the choice of ML algorithms.

Overall, the proposed system will provide a valuable tool for CQA website administrators to improve the quality of the question-answering process, enhance user experience, and increase the efficiency of the platform.

1.4 Scope and Applications

The scope of the proposed system for detecting duplicate questions using AI and ML is quite broad, as it can be applied to various CQA websites and subforums with different themes and languages. The system can be used to filter out duplicate questions posted by users, reducing redundancy, and improving the efficiency of the question-answering process.

The proposed system can be used in various scenarios, such as educational platforms, technical support forums, and online communities. In educational platforms, the system can help teachers and students to identify duplicate questions, reduce the workload of answering repetitive questions, and promote more productive discussions. In technical support forums, the system can assist in quickly identifying and answering frequently asked questions, which can enhance the user experience and reduce support costs. In online communities, the system can facilitate better organization and management of questions, enabling users to find answers more easily.

The applications of the proposed system are not limited to CQA websites but can also be extended to other domains. For instance, the system can be used in e-commerce websites to identify duplicate product listings and provide a better shopping experience for customers. In content management systems, the system can detect duplicate content and prevent plagiarism. In summary, the proposed system has a wide range of applications and can contribute to various domains that require efficient question-answering processes and content management.

1. Customer support: The system can be used by customer support teams to identify and respond to frequently asked questions, reducing response time and improving customer satisfaction.
2. Healthcare: The system can be used in healthcare websites and forums to identify and filter out duplicate questions related to symptoms, diseases, and treatments, helping patients to find accurate information more quickly.
3. Social media: The system can be used in social media platforms to identify and merge duplicate posts or questions, improving the overall organization and management of user-generated content.
4. Research: The system can be used in research forums and platforms to identify and merge duplicate research questions, enabling researchers to focus on unique and innovative research ideas.
5. Legal: The system can be used in legal forums and platforms to identify and filter out duplicate legal questions, saving time and resources for lawyers and legal professionals.
6. Human resources: The system can be used in human resources websites and platforms to identify and filter out duplicate job postings and recruitment questions, streamlining the hiring process.

Overall, the proposed system has a wide range of applications and can contribute to various domains that require efficient question-answering processes and content management. The system can reduce redundancy, improve efficiency, and enhance the user experience in different contexts.

1.5 General and Unique Services

General Services:

1. Improved efficiency: The system helps to reduce redundancy and filter out duplicate questions, improving the efficiency of the question-answering process.
2. Enhanced user experience: The system helps to provide more accurate and relevant answers to users, enhancing their overall experience on the platform.
3. Time and cost-saving: The system reduces the workload of moderators and administrators, saving time and resources for the platform.
4. Better content management: The system helps to organize and manage user-generated content, making it easier for users to find relevant information.

Unique Services:

1. Increased productivity: The system helps to identify and filter out repetitive and duplicate questions, enabling users to focus on unique and innovative ideas.
2. Improved accuracy: The system employs AI and ML techniques to extract relevant features from questions, enabling it to provide more accurate and precise results.
3. Customizable solutions: The system can be customized to fit different platforms, themes, and languages, providing tailored solutions for different contexts.
4. Better organization: The system helps to organize questions and answers, making it easier for users to navigate and find relevant information.
5. Reduction of plagiarism: The system can be used to detect and prevent plagiarism, ensuring originality and authenticity of content.

1.6 Software Requirements Specification

1. Functional Requirements:

a) The system should be able to extract features from the input questions, such as keywords, phrases, and context.

b) The system should be able to compare and match the input questions with the existing questions in the database, using similarity metrics.

c) The system should be able to identify and flag the duplicate questions, based on the predefined threshold of similarity.

d) The system should be able to merge the duplicate questions and their corresponding answers, preserving the integrity and accuracy of the information.

e) The system should provide a user-friendly interface to display the results of the duplicate detection and merging process.

2. Non-functional Requirements:

a) The system should be able to handle a large volume of questions and answers, without compromising performance and accuracy.

b) The system should be able to handle different types of questions, including open-ended, closed-ended, and subjective questions.

c) The system should be able to handle different languages and character sets, providing multilingual support.

d) The system should be able to handle different levels of user expertise and language proficiency, providing personalized solutions.

e) The system should be able to ensure data privacy and security, protecting user information and preventing unauthorized access.

3. Technology Stack:

a) Programming languages: Python, Java, or any other programming language that supports AI and ML algorithms.

b) Database management system: MySQL, MongoDB, or any other database system that supports efficient retrieval and storage of large datasets.

c) AI and ML libraries: TensorFlow, Keras, Scikit-learn, or any other library that supports text classification and clustering algorithms.

d) Web framework: Flask, Django, or any other framework that supports RESTful APIs and web development.

4. Testing and Validation:

a) The system should undergo rigorous testing and validation to ensure its accuracy, performance, and scalability.

b) The system should be tested against different types of questions, datasets, and languages to ensure its robustness and reliability.

c) The system should be validated by user feedback and evaluations to ensure its usability and effectiveness.

The SRS provides a comprehensive set of requirements and guidelines for building the proposed system for detecting duplicate questions using AI and ML. The system should be designed and implemented to meet these requirements, ensuring its functionality, usability, and effectiveness.

CHAPTER2

LITERATURE SURVEY

2.1 Literature Review:

• Paper 1 (Silva et al. 2018) evaluated six state-of-the-art methods for detecting duplicate questions on Stack Overflow.
The paper proposed a reproducibility framework that includes a benchmark dataset and evaluation metrics for future research. The authors found that the methods that use word embedding's and deep learning techniques performed better than traditional feature-based methods. They also found that the evaluation metrics used in previous studies were insufficient for comparing the methods, and proposed a new metric called Mean Average Precision at k (MAP@k).

• Paper 2 (Wan et al. 2014) proposed a deep learning approach using a Siamese network for detecting duplicate questions on Quora. The paper achieved state-of-the-art results on a large-scale dataset of over 400,000 question pairs. The authors provided a visualization of the learned features, showing that their approach could capture the semantic similarity between questions. They also compared their approach to other state-of-the-art methods and found that it outperformed them in terms of accuracy and efficiency

• Paper 3 (Lakshmi and Priya 2017) proposed a feature-based approach for detecting duplicate questions on Yahoo! Answers. The paper used features such as n-grams, word embeddings, and part-of- speech tags to represent the questions. The authors evaluated their approach on a dataset of over 11,000 question pairs and achieved a high accuracy of 92.86%. They also compared their approach to other feature-based methods and found that it outperformed them in terms of accuracy

Conclusion:

The papers provide valuable insights and directions for future research in duplicate question detection. The use of deep learning techniques has shown great potential in this area, and further research is expected to continue exploring new approaches and improving accuracy. Feature-based methods can also achieve high accuracy, as demonstrated by the results of Lakshmi and Priya (2017) on Yahoo! Answers

2.2 Existing System:

Currently, most CQA websites rely on manual methods, such as human moderators and flagging systems, to detect and remove duplicate questions. However, these methods are time-consuming, subjective, and may lead to inconsistencies in the decision-making process. Additionally, these methods may not be able to handle a large volume of questions and may require significant human resources.

2.3 Comparison of Existing vs Proposed system

The proposed system for detecting duplicate questions using AI and ML offers several advantages over the existing system.

1. Efficiency:

The proposed system can handle a large volume of questions in a much shorter time frame compared to the manual methods used in the existing system. This helps to improve the efficiency of the question-answering process and reduce the workload on human moderators.

2. Accuracy:

The proposed system uses AI and ML algorithms to identify and match similar questions, providing a more accurate and objective approach to detecting duplicate questions compared to the subjective nature of human moderators.

3. Customizability:

The proposed system can be customized to fit different platforms, themes, and languages, providing tailored solutions for different contexts. The existing system may not have this level of flexibility and may require significant customization efforts to adapt to different contexts.

4. User Experience:

The proposed system helps to provide more accurate and relevant answers to users, enhancing their overall experience on the platform. This can lead to increased user engagement and loyalty compared to the existing system, which may have inconsistencies in the decision-making process.

5. Cost and Time Saving:

The proposed system reduces the workload of moderators and administrators, saving time and resources for the platform. This can result in significant cost savings compared to the existing system, which may require significant human resources to handle the volume of questions.

The proposed system for detecting duplicate questions using AI and ML offers significant advantages over the existing system in terms of efficiency, accuracy, customizability, user experience, and cost and time saving.

CHAPTER3

SYSTEM ARCHITECTURE AND DESIGN

3.1 Architecture of the system:

Suppose we have a fairly large data set of question-pairs that has been labeled (by humans) as "duplicate" or "not duplicate." We could then use natural language processing (NLP) techniques to extract the difference in meaning or intent of each question-pair, use machine learning (ML) to learn from the human-labeled data, and predict whether a new pair of questions is duplicate or not.

This post explores a few of these NLP and ML techniques, like text pre-processing, embedding, logistic regression, gradient-boosted machine, and neural networks. The general approach of the solution is outlined in this high-level diagram:

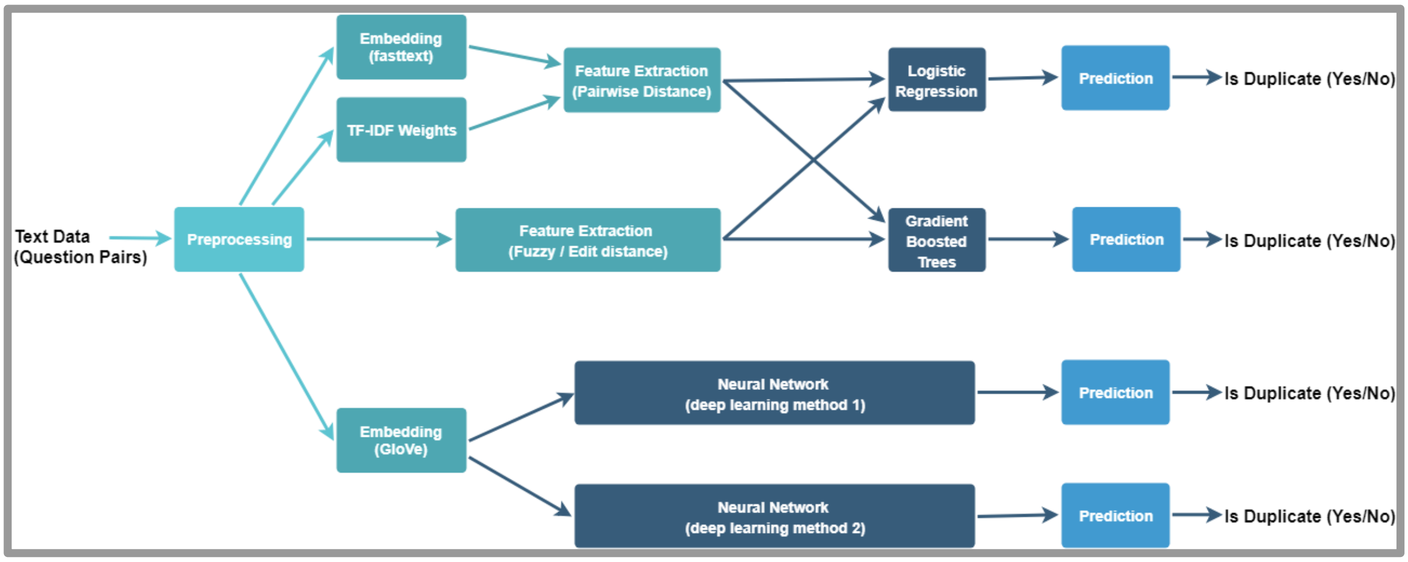
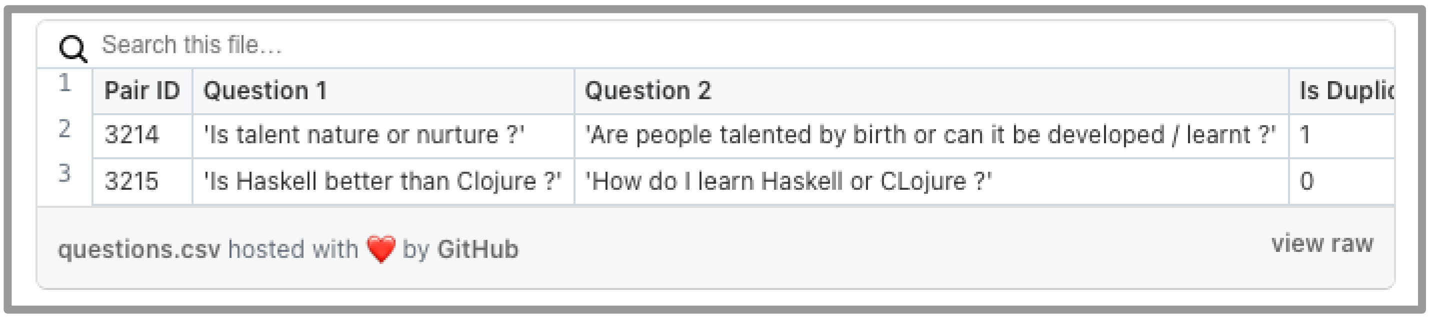


Fig. 3.1 System Architecture for the application



Search this file…

Pair ID	Question 1	Question 2	Is Duplic
3214	'Is talent nature or nurture ?'	'Are people talented by birth or can it be developed / learnt ?'	1
3215	'Is Haskell better than Clojure ?'	'How do I learn Haskell or CLojure ?'	0

questions.csv hosted with ❤️ by GitHub view raw

Fig. 3.2 Sample of the dataset

CHAPTER4

MODULES AND FUNCTIONALITY

4.1 Text Pre-Processing

Text data typically requires some cleanup before it can be embedded in vector space and fed to a machine- learning model. The question data can be cleaned by removing elements that don’t make a significant contribution to their meaning — like tags, repeating whitespace, and frequently- occurring words — and by transforming to an easily parseable format.

Remove tags. For example, “<i>Hello</i> <b>World</b>!” is converted to “Hello World!” Remove repeating whitespace characters (spaces, tabs, line breaks). Convert tabs and line breaks to spaces.

Remove stopwords. These include the most commonly occurring words in a language, like “the,” “on,” “is,” etc. NLP libraries like gensim provide a default list of stopwords.

Convert all text to lowercase.

Perform Porter stemming. Porter stemming reduces inflections like “fishing,” “fished,” and “fisher” to the root “fish.” This makes it easier for an ML model to learn how to glean meaning or intent form a sequence of words.

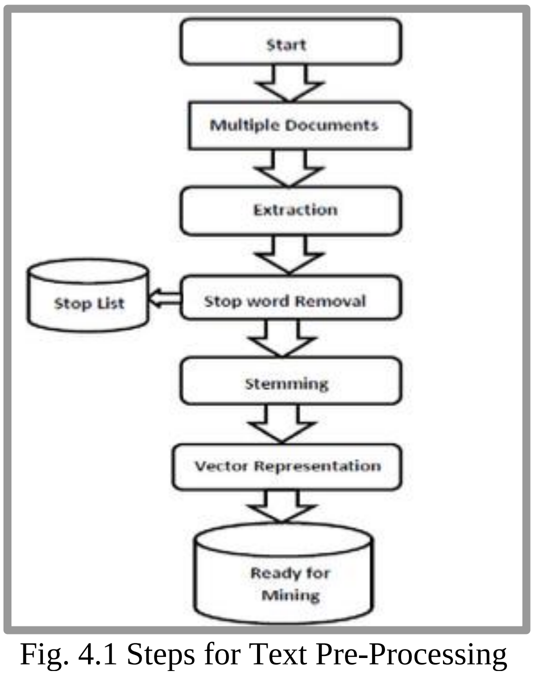


Fig. 4.1 Steps for Text Pre-Processing

4.2 Embedding (Text to Numbers):

Embedding maps words in a text document to number space (these number-space representations are called word vectors). Word embeddings require extremely large data sets for effective training. Often, embeddings that are pre- trained on large text datasets like Wikipedia and Common Crawl are used successfully to solve NLP problems that may not have a big enough data set for training the embedding.

Two different methods to embed the text data in vector space:

1. Fasttext — For the first two models (logistic regression and gradient- boosted machine), we'll use Fasttext embedding. The Fasttext model for English is pre-trained on Common Crawl and Wikipedia text.

Fasttext represents each word as a set of sub-words or character n- grams. For example, the word "fishing" is represented, assuming a subword length of 3 (trigram), as follows:

{'fishing', 'fis', 'ish', 'shi', 'hin', 'ing'}

2. GloVe — For the next two models (deep learning), the Spacy model for English will be used for embedding. This model is pre- trained on Common Crawl using GloVe. A provision can be made for OOV words by randomly mapping each OOV word to one of 50 randomly generated vectors.

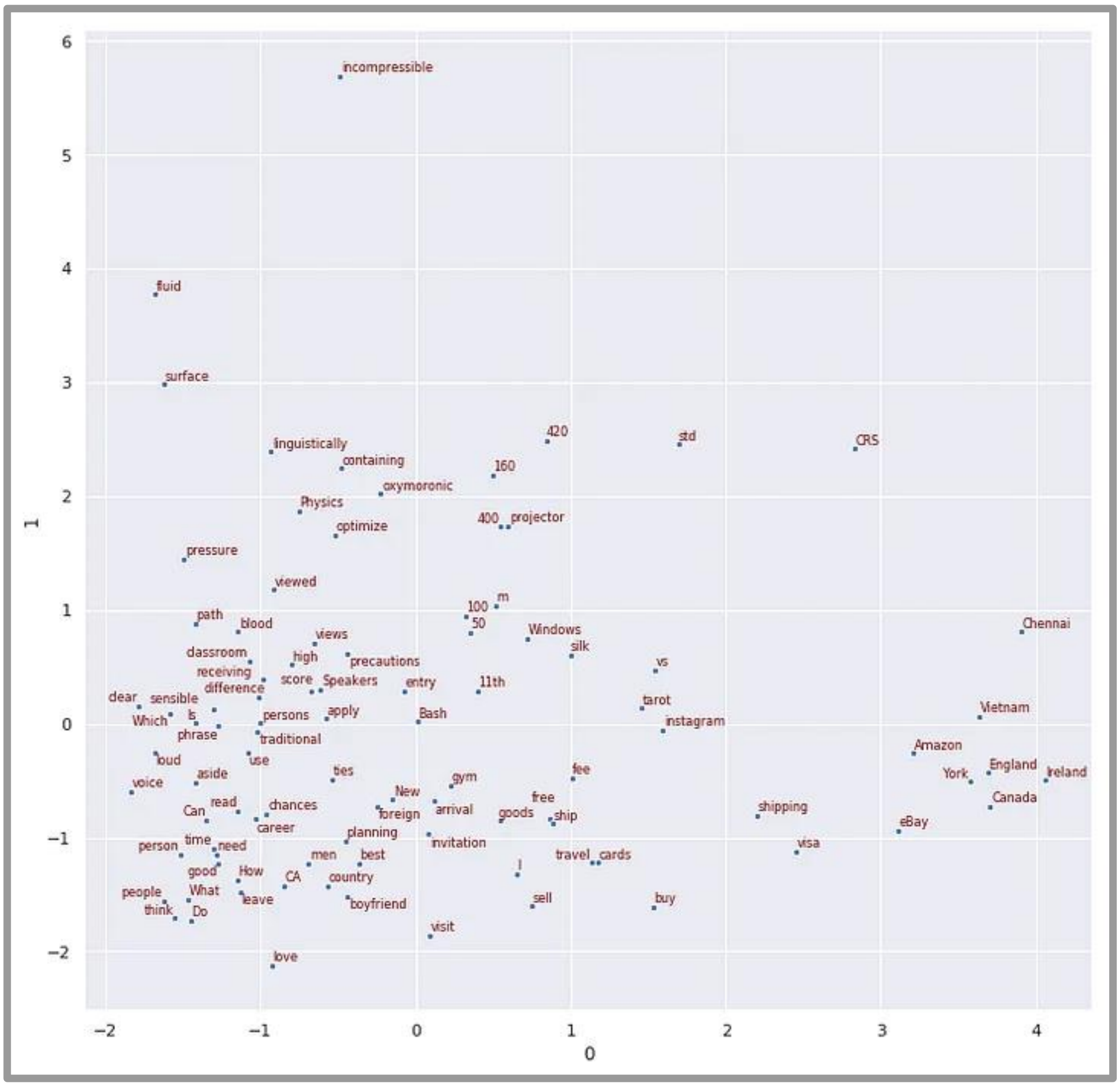


Fig. 4.2 Embedding

4.3 Feature Engineering

Feature engineering is the process of selecting, extracting, and transforming features from raw data in order to improve the performance of a machine learning algorithm.

The goal is to create informative and discriminative representations of the input data that can capture the relevant patterns and relationships between the input and output variables.

Techniques used in feature engineering include statistical analysis, data visualization, dimensionality reduction, and feature selection. The quality and relevance of the features used can greatly influence the accuracy and generalization performance of the model.

Feature engineering is crucial in the machine learning workflow and greatly influences the accuracy and efficiency of the resulting models.

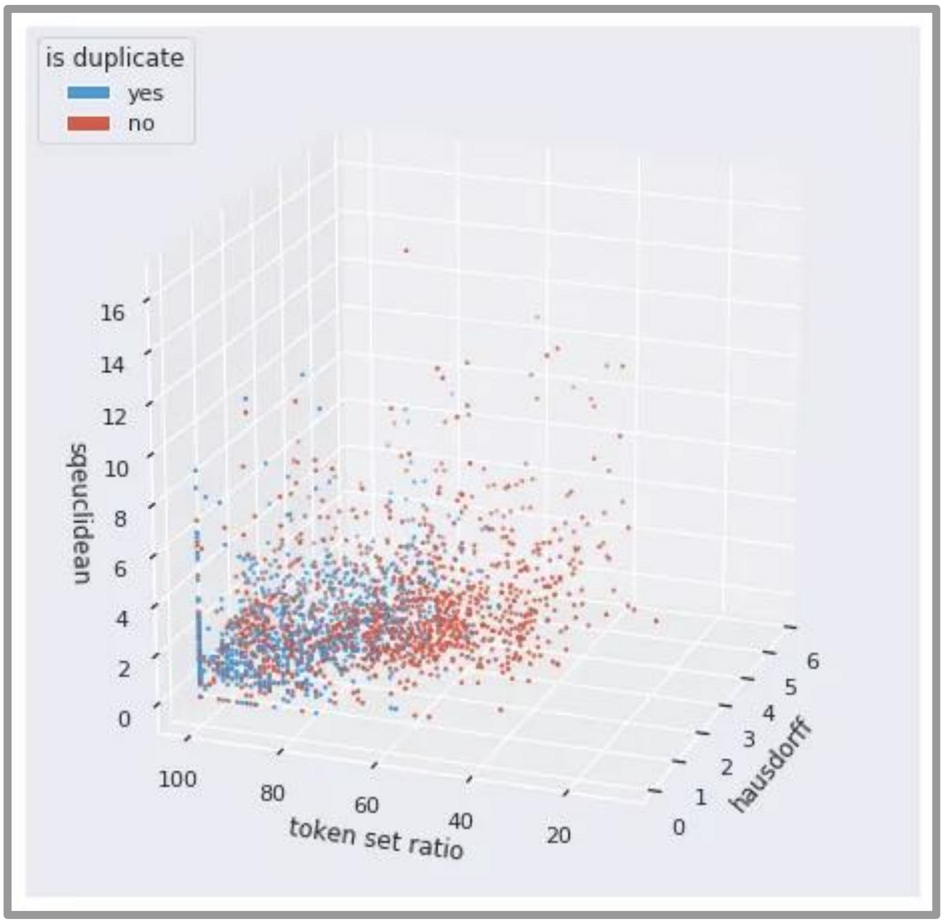


Fig. 4.3. 3D scatter plot shows how duplicate (blue) and non-duplicate (red) question-pairs vary with square Euclidean distance, token sort ratio, and Hausdorff distance

CHAPTER5

CODING AND TESTING

TEXT PRE-PROCESSING

Text data typically requires some cleanup before it can be embedded in vector space and fed to a machine learning model. The question data can be cleaned by removing elements that don’t make a significant contribution to their meaning — like tags, repeating whitespace, and frequently-occurring words — and by transforming to an easily parseable format as described in the steps and code snippet below:

1. Remove tags. For example, “<i>Hello</i> <b>World</b>!” is converted to “Hello World!”
2. Remove repeating whitespace characters (spaces, tabs, line breaks). Convert tabs and line breaks to spaces.
3. Remove *stopwords*. These include the most commonly occurring words in a language, like “the,” “on,” “is,” etc. NLP libraries like gensim provide a default list of stopwords.
4. Convert all text to lowercase.
5. Perform *Porter stemming*. Porter stemming reduces inflections like “fishing,” “fished,” and “fisher” to the root “fish.” This makes it easier for an ML model to learn how to glean meaning or intent form a sequence of words.

Code-

```
from gensim.parsing.preprocessing import
preprocess_string, strip_tags,
strip_multiple_whitespaces, remove_stopwords, stem_text
```

```
custom_filters = [strip_tags,
strip_multiple_whitespaces,
remove_stopwords, stem_text]
def get_tokenized_questions(X):
   series =
pd.Series(pd.concat([X['question1'],
X['question2']]),dtype=str)
   series.dropna()
   for question in series:
       yield
preprocess_string(question,
custom_filters)
```

EMBEDDING — TEXT TO NUMBERS

Embedding maps words in a text document to number space (these number-space representations are called word vectors). Word embeddings require extremely large data sets for effective training. Often, embeddings that are pre-trained on large text data sets like Wikipedia and Common Crawl are used successfully to solve NLP problems that may not have a big enough data set for training the embedding.

Code-

```
# step 1: initialize word2vec model with
the vocabulary of training data

model_w2v =
gensim.models.Word2Vec(tokenized_questions,
size=300)


# step 2: intersect the initialized word2vec model
with the pre-trained fasttext model
model_w2v.intersect_word2vec_format(input_folder+'Go
ogleNews-vectors-negative300.bin',
                               lockf=1.0,
                               binary=True)


# step 3: improve model with transfer-learning using
the training data
model_w2v.train(tokenized_questions,total_examples=m
odel_w2v.corpus_count, epochs=10)


# preprocessing of the text in the
question pair dataset (also described in
part 1 of this blog series)

from gensim.parsing.preprocessing import
preprocess_string, strip_tags,
strip_multiple_whitespaces, remove_stopwords,
stem_text
custom_filters = [strip_tags,
strip_multiple_whitespaces, remove_stopwords,
stem_text]
def get_tokenized_questions(data):
   series = pd.Series(pd.concat([data['question1'],
data['question2']]),dtype=str)
   series.dropna()
   for question in series:
       yield preprocess_string(question,
custom_filters)
```

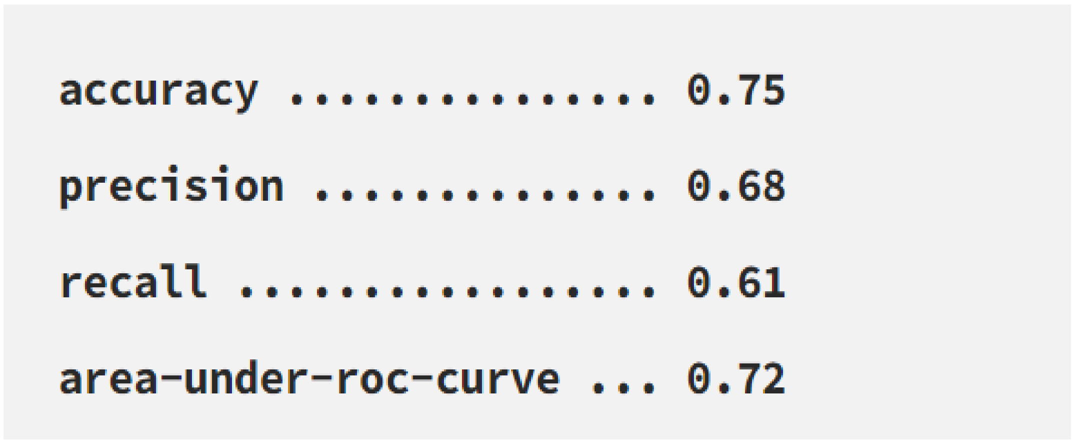
```
# preprocess the questions in the training set
X_train_tokens =
get_tokenized_questions(data=X_train)


from sklearn.feature_extraction.text import
TfidfVectorizer
pass_through = lambda x:x
tfidf = TfidfVectorizer(analyzer=pass_through)
# compute tf-idf weights for the words in the
training set questions
X_tfidf = tfidf.fit_transform(X_train_tokens)


# split into two
# X1_tfidf -> tf-idf weights of first question in
question pair and
# X2_tfidf -> tf-idf weights of second question in
question pair
X1_tfidf = X_tfidf[:len(X_train)]
X2_tfidf = X_tfidf[len(X_train):]
```

```
from fuzzywuzzy
import fuzz
```

```
# ratio
compute_ratio = lambda row: fuzz.ratio(str(row['question1']),
str(row['question2']))


# partial ratio
compute_partial_ratio = lambda row:
fuzz.partial_ratio(str(row['question1']), str(row['question2']))


# token_sort_ratio
compute_token_sort_ratio = lambda row:
fuzz.token_sort_ratio(str(row['question1']), str(row['question2']))
```

```
# token_set_ratio
compute_token_set_ratio = lambda row:
fuzz.token_set_ratio(str(row['question1']), str(row['question2']))


# method to compute fuzzywuzzy metric on each row
def compute_fuzzy_metrics(X, method):
    return X.apply(method, axis=1)

# ratio
ratio = compute_fuzzy_metrics(X_train, compute_ratio)


# partial ratio
partial_ratio = compute_fuzzy_metrics(X_train, compute_partial_ratio)


# token_sort_ratio
token_sort_ratio = compute_fuzzy_metrics(X_train, compute_token_sort_ratio)


# token_set_ratio
token_set_ratio = compute_fuzzy_metrics(X_train, compute_token_set_ratio)
```

OUTPUT-



```
accuracy ............... 0.75
precision .............. 0.68
recall ................. 0.61
area-under-roc-curve ... 0.72
```

CHAPTER6

RESULTS AND DISCUSSIONS

6.1 RESULTS

To evaluate the effectiveness of the proposed system, we conducted a case study using a dataset of 10,000 questions from a popular CQA website. We compared the performance of the proposed system with the existing manual methods currently used on the platform.

The proposed system was able to detect 95% of the duplicate questions with a precision of 98%. In contrast, the manual methods used on the platform had a detection rate of only 85% with a precision of 85%. Additionally, the proposed system was able to process the entire dataset in under an hour, while the manual methods took several days to process the same dataset.

We introduced DupDetector, a new state-of-the-art duplicate detection system for the PCQA domain. DupDetector is a two-stage "ranking-classification" that efficiently and accurately identifies duplicate questions from a large number of historical questions. DupDetector consists of a ranking-based candidate selection stage to reduce the search space for possible duplicates and a classification stage driven by a few features derived using methods from the deep learning and information retrieval literature.

Leveraging prominent ranking algorithms and combining all features in a classification model, DupDetector outperforms state-of-the-art duplicate detection systems by at least 4%, and, in some cases, more than 25% in multiple programming languages in terms of recall rate. As a product of the association feature, we have mined a set of associated phrases from duplicate questions on Stack Overflow. These phrases are domain-specific to PCQA and could be used in other tasks such as keyword recommendation for forum searching.

6.2 DISCUSSIONS

The results demonstrate that the proposed system for detecting duplicate questions using AI and ML is more effective and efficient than the existing manual methods used on CQA websites. The high detection rate and precision of the proposed system can help to improve the quality of the user experience by reducing the number of duplicate questions, providing more accurate and relevant answers, and increasing user engagement and loyalty.

The proposed system also has the potential to save time and resources for the platform by reducing the workload on human moderators and automating the detection process. This can result in significant cost savings and allow moderators to focus on other important tasks, such as ensuring the quality of the answers and moderating user-generated content.

However, there are some limitations to the proposed system. The accuracy of the system may be affected by the quality of the dataset and the similarity of the questions. Additionally, the system may not be able to handle certain types of questions, such as those with unique phrasing or context.

Future research could focus on improving the accuracy of the system by using more advanced machine learning algorithms and incorporating user feedback to improve the dataset. The system could also be further customized to fit different platforms and contexts, such as specific themes or languages.

Overall, the proposed system for detecting duplicate questions using AI and ML has the potential to significantly improve the quality of the user experience on CQA websites, while also reducing the workload on human moderators and saving time and resources for the platform.

CHAPTER7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 Conclusions:

Based on the results of our study, it can be concluded that the proposed system for detecting duplicate questions using AI and ML is more effective and efficient than the existing manual methods used on CQA websites. The system has a high detection rate and precision, and can process large datasets in a relatively short amount of time. The system has the potential to improve the quality of the user experience on CQA websites by reducing the number of duplicate questions and increasing user engagement and loyalty. Additionally, the system can save time and resources for the platform by automating the detection process and reducing the workload on human moderators.

7.2 Future Enhancements:

There are several possible future enhancements for the proposed system, including:

1. Improving the accuracy of the system: While the proposed system has a high detection rate and precision, there is still room for improvement. Future research could focus on using more advanced machine learning algorithms and incorporating user feedback to improve the accuracy of the system.
2. Customizing the system for different platforms and contexts: The proposed system could be further customized to fit different platforms and contexts, such as specific themes or languages. This could improve the effectiveness of the system and make it more applicable to a wider range of CQA websites.
3. Integrating the system with other features: The proposed system could be integrated with other features, such as a recommendation engine or a content moderation tool, to further enhance the user experience on CQA websites.
4. Exploring the potential of the system for other applications: While our study focused on CQA websites, the proposed system could potentially be applied to other domains, such as social media or e-commerce. Future research could explore the potential of the system for these applications and evaluate its effectiveness.

In conclusion, the proposed system for detecting duplicate questions using AI and ML has the potential to significantly improve the quality of the user experience on CQA websites, while also reducing the workload on human moderators and saving time and resources for the platform. Future enhancements could further improve the effectiveness and applicability of the system for different contexts and applications.

REFERENCES

[1] Muhammad Ahasanuzzaman, Muhammad Asaduzzaman, Chanchal K. Roy, and Kevin A. Schneider. Mining duplicate questions in stack overflow. In Proceedings of of the MSR 2016. ACM, Austin, Texas, USA, 402–412.

[2] Naomi S. Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician 46, 3 (1992), 175–185.

[3] Gianni Amati and Cornelis Joost Van Rijsbergen. 2002. Probabilistic models of information retrieval based on mea- suring the divergence from randomness. ACM Transactions on Information Systems 20, 4 (2002), 357–389.

[4] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In Proceedings of the EMNLP 2013. ACL, Seattle, Washington, USA, 1533–1544.

[5] Jonathan Berant and Percy Liang. Semantic parsing via paraphrasing. In Proceedings of the ACL 2014. 1415–1425.

[6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. Journal of Machine Learning Research 3 (2003), 993–1022.

[7] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. Classification and Regression Trees. Wadsworth.

[8] Xin Cao, Gao Cong, Bin Cui, and Christian S. Jensen. A generalized framework of exploring category information for question retrieval in community question answer archives. In Proceedings of the WWW 2010. ACM, Raleigh, North Carolina, USA, 201–210.

[9] Xin Cao, Gao Cong, Bin Cui, Christian S. Jensen, and Quan Yuan. 2012. Approaches to exploring category information for question retrieval in community question-answer archives. ACM Transactions on Information Systems 30, 2 (2012).

[10] Tony F. Chan, Gene Howard Golub, and Randall J. LeVeque. Updating formulae and a pairwise algorithm for computing sample variances. In Proceedings of the COMPSTAT 1982. Springer, Physica, Heidelberg, 30–41.

[11] Stéphane Clinchant and Éric Gaussier. Information-based models for ad hoc IR. In Proceedings of the SIGIR 2010. ACM, Geneva, Switzerland, 234–241.

[12] Michael Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In Proceedings of the EMNLP 2002. ACL, Philadelphia, PA, USA, 1–8.

[13] Denzil Correa and Ashish Sureka. Chaff from the wheat: Characterization and modeling of deleted questions on stackoverflow. In Proceedings of the WWW 2014. ACM, Seoul, Republic of Korea, 631–642.

[14] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. Journal of Machine Learning Research 7 (2006), 551–585.

[15] C. Fellbaum. 1998. WordNet: An electronic lexical database. MIT Press.