IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 27, 2018, accepted October 12, 2018, date of publication October 31, 2018,
date of current version December 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2878523

A Smart Email Client Prototype for Effective Reuse of Past Replies

M. ASIF NAEEM[1], (Member, IEEE), I. WAYAN S. LINGGAWA[1], AFTAB A. MUGHAL[2], CHRISTOF LUTTEROTH[3], AND GERALD WEBER[4]

[1]School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland 1142, New Zealand
[2]AGI Education Limited, Auckland 1010, New Zealand
[3]Department of Computer Science, University of Bath, Bath BA2 7JX, U.K.
[4]Department of Computer Science, The University of Auckland, Auckland 1142, New Zealand

Corresponding author: M. Asif Naeem (mnaeem@aut.ac.nz)

This work was supported by the New Zealand ASEAN Scholarship (NZAS) Fund.

ABSTRACT Email communication is widely used in managing customer queries such as complaints and inquiries. With an increasing number of customer emails being received every day, better tools are needed to answer emails efficiently, reuse past efforts and prevent overwhelming and cluttered email backlog. Finding similar past cases is difficult since customer requests might use non-standard terminology. In this paper, we develop a smart email client prototype which helps in replying to an email by providing a list of replies gleaned from the past replied emails. The suggested replies are ranked according to the level of similarity. We have evaluated real-world email data in order to ascertain that reuse is possible, but requires careful retrieval mechanisms. We implement and evaluate a case-based reasoning approach as a methodology to solve the problem by reusing previously written solutions from the past replies stored in the case base. We build a retrieval algorithm that finds similar cases beyond the exact matching, by using text processing and semantics analysis techniques. To optimize retrieval, we apply and cross-evaluate several text analysis techniques such as lexical analysis and synonym expansion, and our evaluation shows that synonym expansion could improve the chance of retrieving a more relevant match even at lower ranks. We evaluate our prototype based on the quality of retrieval results, index size, and processing time elapsed.

INDEX TERMS Case-based reasoning, indexing, information retrieval, email client, synonym expansion, vector space model.

I. INTRODUCTION

Email is a preferred communication channels as reflected by the fact that in 2015, the total number of email accounts reached 4.3 billion, a figure that is expected to grow by six-7 percent in the next four years [1]. In the same period, there were 2.5 billion email users, an annual growth of three percent. This indicates a user may have at least one email account, with an average of 1.72 email accounts per user, and more than 112 billion email messages are sent and received daily [1].

While these overall statistics show the importance of email, it is also interesting to see the statistics of user activities. According to the statistics presented in [1], an average of 122 emails are sent and received by a user daily. Furthermore, 72% are received and 28% are sent emails. Assuming that the received email messages are queries, the user is expected to manage the workflow to respond to the query properly, or it may result in a personal information management issue called *email overload*.

Historically, managing email messages has been done through an application called *email client*, which originally had the main role of accessing the user's mailbox in a mail server. However, as email communication evolved and became more complex, an email client became capable of dealing with task management applications as well as personal archiving. One of the components of task management is related to how a user can receive a message (which may come in the form of a query), and how they respond to that message, (in other words, providing the solution in reply to the query). Due to the increase in daily incoming emails (or queries), the user needs a more efficient workflow to respond to the query.

An automatic email response can assist users in replying their emails. In particular it can be helpful for such

workplaces where we expect a large number of emails everyday while most of these emails are related to a few common issues. Examples of such workplaces are customer services of hotels, shopping centres, educational help desk, social events, etc. Since most of the time the emails are related to a few common issues, there replies are also similar. This creates an opportunity of using past replies for responding future emails. In other words, a user could benefit from reusing his/her own past replies if the query is similar. This may improve the experience of responding to an email, as the user does not need to start with a blank message.

Enhancements in retrieving documents based on the user query can be achieved by implementing additional knowledge related to that domain. Some studies suggest that providing a domain-specific ontology may improve the chance of finding a more relevant result [2]. However, this specific knowledge might not be useful, since the context of the terms might not be relevant to the general domain. Another approach adopts language-based knowledge to accommodate lexical semantics by understanding words such as synonyms [3], [4].

The problem-solving paradigm has also evolved over time, and recently Case-Based Reasoning (CBR) methodology has been introduced [5], [6]. CBR solves a problem by retrieving similar past problems from the collection (knowledge base) - referred to as the case base - and reusing its solutions for the new problem. This approach is commonly known as *lazy learners*; it performs the function of retrieving similar knowledge at the time the query is presented instead of first building a training model.

In this paper, we investigate the hypothesis that we can use CBR methodology to solve the email overload problem [7]. We have analysed real world email data sets in order to investigate the potential for applying CBR. Our analysis shows future questions could be mapped by utilising similar previous questions and reusing the answers given to them. This approach would still allow the user to have control of the task; further, it could reduce the time and energy required to compose from scratch or search through the whole mailbox to find similar queries.

Our proposed email client solution is a desktop application that has standard functionality for email communications and an enhanced capability to intelligently provide recommendations for reply. The application starts performing when a user initiates the action to reply to a new incoming message. It then attempts to retrieve similar past incoming emails (known as cases), reusing the answers, and treating them as a reply recommendation list. The process is finished when the user selects a result from the list and sends the email. The answered email is considered to be a problem solved, and is stored in the case base.

Contributions: The paper offers the following contributions:

- We develop a prototype of smart email client which helps in replying to an email by providing a list of replies gleaned from past replied emails.
- We apply the CBR methodology to the problem domain, the email reply behaviour.
- We build a retrieval algorithm to find similar cases beyond the exact matching using text processing and semantics analysis techniques [3], [4].
- We evaluate our prototype based on the quality of retrieval results, index size, and processing time elapsed.

The rest of the paper is organised as follows. Section II presents related work in detail. Section III describes background and preliminaries to understand the problem. Section IV presents our new smart email client prototype including its system overview and functionality design. Section V describes implementation of the system. Section VI provides evaluation of our prototype. Section VII presents discussion on results we obtained. Section VIII concludes the paper and indicates some future directions.

II. RELATED WORK

In this section, we examine current developments in responding to email, and identify the gap. A key challenge in email communication is related to defining a strategy on how to respond to the high volume of incoming email. This usually arises in customer-related domains such as help desks, event management, or customer service [8], [9]. Most of the time, users have already replied to a similar inquiry; however, they spend time browsing through their previously sent emails in order to obtain the respective reply (or solution). This is a time-consuming and frustrating process. Several studies have attempted to assist users in managing email responses, as summarised in subsections below.

A. AUTOMATIC EMAIL ANSWERING OR TEMPLATE GENERATION

One method of managing an email reply is to implement an automated email answer. The extensive review by Sneiders [10] shows that the main approaches for this method use machine learning techniques.

Kosseim *et al.* [11] conducted a study that arose from the typical task of answering an email inquiry, which included recognising the content (usually a problem) and generating the reply response. Since the domain is customer support, they identified the particular categories that are specifically available in that domain, such as a *how-to question*, *suggestions*, *problem reports*, and others. To analyse the message text, an information extraction process is undertaken to identify specific information and represent it in the structured format. Tokenisation and other lexical analyses are performed to group part-of-speech and phrases. The text then fills the prepared templates used in the discourse analysis. Response formulation is prepared afterwards and typically has a structured content, such as beginning with a salutation (and the name of the customer) and ending with a formal closing.

A similar approach [12] was presented for preparing predefined templates to answer customer queries. This idea arises from an understanding that in a set of query-response email pairs, the association between the question and answer can be identified. This association would be useful to map

similar future questions to their answer templates. The system is trained to initially identify a large number of email message pairs in the archive in order to learn to classify them into standard answers. In mapping new email queries to the old ones in the archive, this system employs WordNet language reference. The system performs at 61 percent capability compared to a human-performed task.

Another approach [13] was presented to focus on providing a set of reply templates based on the incoming email content. Concepts, as a collection of terms, were extracted from the body of the email and indexed. The main contribution of this approach was finding that indexing could make the search for the associated reply template faster. However, the knowledge base for a reply template needs to be defined before the system can associate them with an incoming email. In addition, the recommendation only considers a template calculation score, which came from the terms that appear more frequently. The best matched reply templates were then given to the users.

Kannan *et al.* [14] developed a novel approach for Reply email messages using short response, Smart Reply. Given an incoming email message, the system predicts possible response suggestions that users can choose with their fingertips. These suggestions represent a simple *yes*, *ongoing*, and *no answer*, which were formulated from semantic clustering techniques. This approach has been implemented in *Inbox by Gmail* and at least 10 percent of the composed Replies on that system were assisted using this Smart Reply.

Van Gysel *et al.* [15] argue that being proactive in assisting email Reply composition can be done by suggesting email attachments too. The idea arises from a context in which the request for file may or may not be explicit in the message; however, it may still be appropriate to attach relevant files. The developed system identifies the terms in the subject and body of the email message and suggests the appropriate file attachment in a ranked manner. The evaluation result shows that the system's ability to extract a request query can be affected by the amount of noise in the subject and body field of the message.

The work presented in [16] identified the importance of intent and tasks in an email message to improve automated classification process. In their study, a new taxonomy of intent (e.g. implicit reason of why someone sends a message) and task (e.g. the task subject) was proposed to assist annotation process of two email datasets, Enron and Avocado. The evaluation results on classifying number of tasks in an email message shows that the annotated data allows the automated classification process to achieve 71% accuracy.

Extensive observation presented in [17] suggested that an inquiry in an incoming email can be separated into two parts: the context or description of the problem, and a request to resolve. These two elements are important in delivering the most appropriate reply generation. In this study, instead of generating answer templates, the authors designed their own reply pattern, which was manually designed and consists of syntax and regular expression.

While providing an intensive review of automatic email answering, Sneiders [10] also argued that business nowadays does not commonly practice automated email answering. This happens because of the fear they will lose contact with their clients, and lead to lost opportunities. This perspective aligns with our study; we realise users still want control over the actions they perform; therefore, the solution we propose should be able to put user control one step above the automated tasks.

In addition, we can see from the recent studies that most attempted to develop solutions based on the rule-based approach, which has the advantage of utilising a set of predefined rules to build a strong knowledge base for performing the automation process. However, the main drawback of the rule-based approach is that it requires an extensive knowledge base before it generates results. In some newly established recommendations or automation systems, a robust knowledge base might not be present. Thus, the solution does not work well when the knowledge base has less or minimal information.

B. PREDICTING REPLY ACTION

Automatic template generation is useful for an efficient reply to a homogeneous query. Often, the user needs to be assisted in a way that gives them more control over the Reply process. In this case, some of the studies presented below suggest that by identifying the structure and content of an email message, a prediction of whether a message needs a reply can be made.

Ayodele and Zhou [18] conducted a study on generating reply prediction using the machine learning technique. This study was part of the email management system the author previously used [19]. The main idea is to generate a prediction of whether or not an incoming email message requires a reply and represent it in a form of labelling. The prediction will appear as the result of a scoring mechanism calculation [20] based on the email subject and content extraction, which is examined based on indicators such as interrogative words, question marks, domain recognition of the sender's email address, attachments, and most-used phrases. Each email is given a label of *definitely needs reply* or *needs no reply* and is shown to the user according to the score.

The work presented in [18] showed this solution could assist the users to determine which email is prioritised to be replied to. However, it is not clear in their study how they perform the text processing, other than finding the interrogative words. In addition, the authors did not explicitly state whether they considered the body content of the email.

Dredze *et al.* [21] developed a prototype for predicting whether a reply is necessary for an email message. This work differs from that of [18], since it includes a prediction for the attachment too. The reply prediction indicates that a message requires a reply by establishing a rule-based system using reply predictors. The predictors include the user profile (e.g send/received statistics and address book), as well as the presence of the words *reply* or *urgent*, and of question marks in a message. Afterwards, the attachment predictors

benefit from the discovery of the words *attach*, *attaching*, *attachment*, or *attached*.

A study [22] for predicting email reply behaviour was performed based on two approaches: will the message be replied to, and when will it be replied to? The study particularly focused on email communication in the enterprise domain. It investigated various email metadata that are related to behaviour. The system showed prediction of probability of the message to be replied to and the reply time latency. The evaluation result from the feature selection process showed temporal (message time-stamp), historical interaction (conversational thread), and message content were important email features in the prediction of reply behaviour.

While previous studies have demonstrated the usefulness of features for user control, as in providing labels and alerts, the system employed a rule-based approach under the hood. Therefore, to optimise the classification process, collecting sufficient training data to build the knowledge base for classifying email is necessary.

C. REUSING PREVIOUSLY AUTHORED REPLY MESSAGE

Some of the studies presented below extend beyond automatic reply, template generation, and predicting the reply. They know that a new incoming email forming a query (or problem) may have been replied to in the past. Therefore, the exploration and utilisation of past messages is identified, based on the notion that similar problems may have similar answers.

Lamontagne *et al.* [2] developed a solution to deal with customer email messages sent to a company. The goal here is to adapt previous email messages in order to reply to a new request. To achieve this, the system compares the new request message with a collection of past messages in order to find the most similar one. When the most appropriate past message is found, the system attempts to reuse the corresponding answer message to reply to the request.

The study conducted by [2] was also part of the Mercure project [23]. This study is domain-specific, notably in terms of investor-relation messages. Thus, the authors explain that no domain specific resources are present. They did not benefit from the existing linguistic resources such as WordNet; however, by expanding the query into term co-occurrences, they benefited from a slight increase in the result. Word embeddings e.g. word2vec [24] can be another option to retrieve the similar replies from the past emails [25]–[35] but that is not included in the scope of this paper.

Although providing reply predictions and templates could minimise the users' efforts to reply to similar emails, an extensive training model with a high volume of samples could be required to achieve better performances. This investigation, however, suggests user intervention could help to provide a better recommendation process by giving feedback on the most relevant results [36]. Feedback was given after the system performed a search for the most similar sent emails. A higher score could be achieved when the users vote a particular result to be the most appropriate reply. Hence, the system also learns about user preference.

III. BACKGROUND AND PRELIMINARIES

The problem-solving paradigm has also evolved over time, and recently Case-Based Reasoning (CBR) methodology has been introduced. CBR solves a problem by retrieving similar past problems from the collection (knowledge base), referred to as the case base, and reusing their solutions for the new problem. This approach is commonly known as *lazy learners* as it performs the function of retrieving similar knowledge at the time the query is presented rather than building the training model first.

A. CBR CYCLE

Case-based reasoning as a framework for problem solving has two main components: the modelling process translated as a CBR cycle, and the task-method structure associated with each process in the model. The diagram in Figure 1 represents the high-level process of adapting the CBR approach. It is widely recognised as the 4R's: Retrieve, Reuse, Revise and Retain.

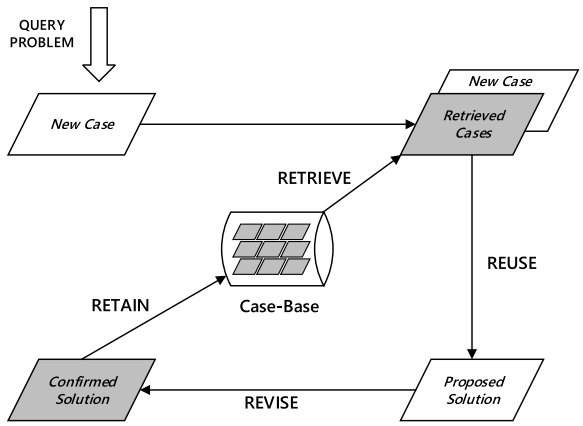


FIGURE 1. The CBR Cycle adapted from [37].

A case typically consists of a problem description and its corresponding solution. When a query problem arrives, it is treated as a new case with a defined problem and no solution yet; then, an attempt is made to retrieve past cases from the memory (or case base) with a similar problem description. Next, the solution to these past cases is obtained to be reused as a proposed solution. The system either automatically, or through user intervention, attempts to revise the solution by considering the differences to the current problem. The proposed solution is evaluated by applying it to the initial problem, or assessed by domain experts on its respective problem domain. Finally, the confirmed solution is retained in the case base to be considered for solving similar problems in the future.

B. CATEGORISATION OF THE CBR SYSTEM

There has been a significant development in various CBR systems. In [6] the differences are categorised based on four criteria (or dimensions): the source of knowledge, function,

organisation, and distribution. Although it is not an exhaustive list of every CBR system development, we found it useful to see the positioning of our CBR system within real world implementation, so we attempted to examine our email client system based on the criteria previously mentioned.

The first dimension identifies what the form of the case is. Some CBR systems have their source in the conversation between the user and the system, the interpretation of images, or even in the information from temporal relationships such as user action history in a game. Furthermore, this information can be stored in predefined variables for a specific domain such as the medical field. Our system relies extensively on the presence of text in the email message, and this textual collection is easily obtained.

The second dimension groups the CBR system based on the function of its development. It is common to see a CBR system mainly utilised for a classification task where it can predict labels or classes in a binary (positive or negative) or discrete (multiple) manner. A CBR system also assists in the processes of a development life cycle such as knowledge management, planning, and monitoring the deviation in the behaviour of a system. In some cases, it can also provide recommendations according to user preference and interaction in the system. Our system attempts to provide this recommendation based on information acquired from the user's past responses in replying to messages.

A CBR system can be combined with other knowledge-based systems or even another CBR system that has a different functionality to solve a problem. This combination may involve a multilevel organisation used by several CBR systems, hybridised with other problem-solving methodologies, or utilise meta information acquired by another CBR system to reason the best method to apply at every CBR stage. On the other hand, it is also common to use a single CBR system for problem solving. In our case, it is sufficient to use only a single CBR system, as the nature of our task is relatively simple.

Finally, a CBR system can be categorised by either the number of the case-bases (or memory) or the type of processing distribution. It can have one case-base or many, and the processing can be done by either single or multiple agents in a system. Multiple distributions may be used for building case-bases to solve complex problems, or for sharing processes that are computationally expensive. Our system focuses on a relatively simple process, which involves finding and recommending email messages from our case-base. In addition, it is not necessary to build multiple case-bases since a single case-base already represents information storage in an email account.

C. TEXTUAL CBR FOR TEXT-BASED PROBLEMS

In recent years, an interest in exploring CBR that specifically deals with text problems has emerged. This is possibly because CBR itself is a methodology [37]; thus, it is open to new problem domain discoveries, such as text. Experience and information from past problem solving processes is retained and explicitly reused to deal with new tasks or problems. This extension is commonly referred as *textual CBR*.

One of the typical areas that implements textual CBR is the customer support domain. Normally, this service relies heavily on text-based documents such as error reports, technical documentation, or frequently asked questions (FAQs) to solve customer problems. The problem itself may not necessarily be solved within seconds, but the support staff needs a system to assist them in finding the documents that are relevant to a customer's query. In addition, a specific technique to retrieve information based on natural language text in a customer query is mandatory. Further explanation of this technique is presented in the next section.

Since CBR is a knowledge-based approach, it is important to note that a textual CBR system needs knowledge (or case) representations in any or some of the following categories: case collection, index vocabulary for each case, similarity measurement, and knowledge adaptation [38]. First, text-based documents can be utilised as the source of the case base. Then, the index is constructed around the terms used in the document. Through considering the various terms in the index, the similarity measurement is performed between the query and the document collections. In addition, this measurement may go beyond the statistical term weighting by utilising semantics components such as a thesaurus or ontology from a particular domain.

The domain that we are dealing with, email communication, has the characteristics that make it suitable for textual CBR implementation. The document collection used as the basis of the case base has a text-based form. Even though an email message has a clear structure consisting of sender, recipient, date, subject, and body, the greatest source of information is located in the subject and body. Moreover, these fields may form a semi-structured or even unstructured text, making it a challenge to extract useful knowledge from it. Therefore, a textual CBR system relies heavily on a technique that identifies the natural language text in an email message.

Our work is closely related to that of [2], [36] as we endeavoured to utilise past email replies to answer an incoming email inquiry. However, only one of the studies implemented Case-Based Reasoning as the underlying methodology [2]. Contrary to our work, the authors have not provided experimental evaluation of their approach. Also, we believe future questions could be mapped by utilising similar previous questions, and then reusing the answers given to them. This approach would still allow the user to have control of the task. Further, it could reduce the time and energy required to compose from scratch or search through the whole mailbox to find similar queries.

IV. SMART EMAIL CLIENT

Our email client solution is a desktop application that has standard functionality for email communications and an enhanced capability to intelligently provide recommendations for a reply. The application starts performing when

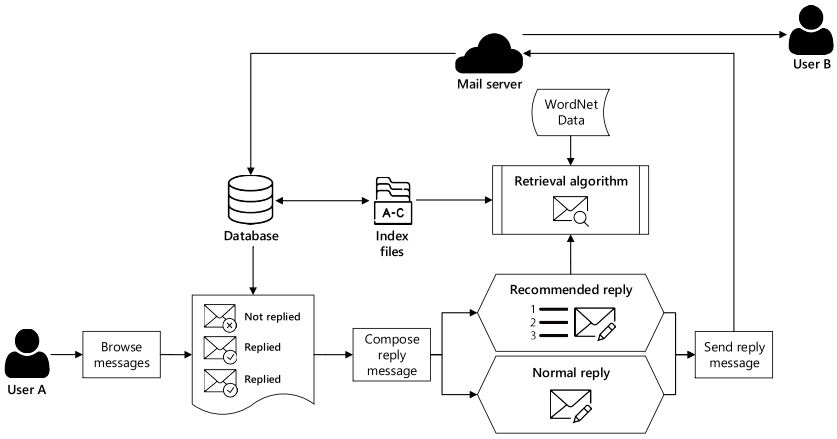


FIGURE 2. System overview.

a user initiates the reply action for a new, incoming message. It then attempts to retrieve similar past inbound emails (known as cases), reusing the answers, and then using them as a basis for a reply recommendation list. The process is finished when the user selects a result from the list and sends the email. The answered email and its reply is considered to be a solved problem and is stored in the case base for the future.

A. SYSTEM OVERVIEW

Figure 2 provides an overview of our smart email client. The user starts by browsing through their inbox messages. These messages are stored locally in the database after having been retrieved from the mail server. Two types of message are available in the inbox: those replied to and those that have not been replied to. When the user selects a message to reply to, the system provides two options for creating a reply. First is the normal reply option that implements a basic reply procedure – a blank text box with the original message attached at the end of the box. Second, is the recommended reply option that provides a ranked list of replies obtained from the user's own sent items. When the message composition is complete, the user sends the message, the system delivers it to the mail server, and then stores it in the database for future retrieval.

1) NORMALISING MESSAGES WITH TEXT ANALYSIS

In this application, we dealt with text-based email messages; therefore, it is important to perform several techniques to analyse the content of the message in order to remove the noise and identify meaningful context. First, we performed lexical analysis to normalise the text, including, but not limited to, the treatment of punctuation marks, digits, and hyphens, as well as upper and lower cases. This process is continued by converting a stream of text in a document, commonly separated by a whitespace, into a stream of candidate words to be indexed. Once the stream of potential words for the use of index terms has been generated through lexical analysis, some may be shown to have less influence on the context of the document. This happens when they appear too frequently in a document, and can be useless in the retrieval task. Thus, we removed those words commonly known as stopwords.

In addition, we derived a word by its root. This method is commonly applied to reduce the variability of words; thus, more matching can be achieved. For example, a string such as *buy*, *buys*, and *buying* can be reduced to its root, buy. This can reduce the number of index terms so a lower number of distinct words (word roots) are stored.

Finally, we performed term frequency - inverse document frequency (TF-IDF) – the weighting of a word to measure its importance to a document within a corpus or collection. This weighting considers the scaling mechanism: frequent terms are scaled down and rare terms are scaled up. For example, a term that appears 10 times more than another may not necessarily be more important.

2) MEASURING SIMILARITY IN THE VECTOR SPACE MODEL

The Vector Space Model (VSM) transforms a collection of words in a document into a high dimensional vector. The weight of the vector is represented by a set of document terms that have been weighted (in this case using TF-IDF). The distance between two vectors is measured using the cosine angle between them, which is usually referred to as the Euclidean distance. Thus, it appears that the shorter the distance between two vectors, the more similar the documents are.

3) EXPANDING SYNONYMS USING THESAURUS

During the process of the semantic identification of words in a document, a common strategy is to find a set of words that closely relate to a given word. In linguistics, a thesaurus is used to find these words known as synonyms.

The use of the thesaurus was motivated by a desire to achieve advantages such as a reduction in noise, and retrieval based on concepts (semantic matching) rather than on words (exact matching). It is useful when the domain has an

extensive and specific use, such as in the medical field. In the general domain, a popular lexical database using a thesaurus in the English language, known as WordNet.

B. FUNCTIONALITY DESIGN

This section explains three main functionalities of our prototype: the email communication, information retrieval, and user interface as shown in Figure 3.

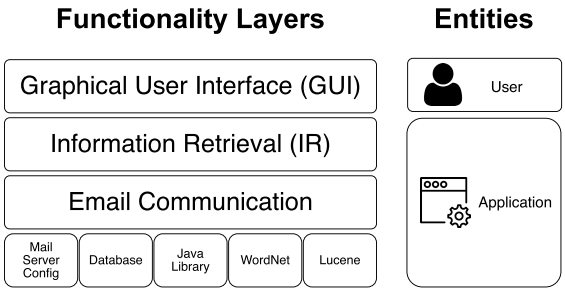


FIGURE 3. Proposed functionality layers in the application.

1) EMAIL COMMUNICATION

In order to fetch and send email, we developed an email communication component which was able to perform the following: (1) establish a connection to the mail server using given email account settings; (2) fetch all email messages in the first run and fetch only new messages on subsequent runs, and (3) use mail server settings to send outgoing emails. The system utilises Internet Message Access Protocol (IMAP), thus it is possible to fetch from the inbox and send messages from its corresponding folder. We developed the sending capability using the Simple Mail Transfer Protocol (SMTP) settings of the mail server. All these tasks were achieved by implementing JavaMail API.

The email message that has been fetched needs to be further processed, since it mostly comes with embedded Hypertext Markup Language (HTML) tags. This process is known as *parsing* the content in the body part of the message. Parsing is performed to ensure the content of the message does not contain significant noise. While we carefully dealt with the quality of the content, we left the parsing for multiple email recipients to future development. At this stage, therefore, our system focuses on one-to-one email communication between a sender and a recipient. After building the communication capability and parsing the content, the messages are stored in the MySQL database.

2) INFORMATION RETRIEVAL

The next component that we developed is related to the IR capability. This is the core component in the architecture as it enables the ad-hoc retrieval of documents (in this case the email message), according to the user query. It also performs the following tasks: (1) the indexing of message entries in the database, (2) the querying of the index to find the most similar document, and (3) the ranking of the most relevant documents similar to the user query.

We used Apache Lucene[1] to perform the above tasks. There are several reasons for that. First, Lucerne is an open source library and available for Java-based application development, which aligns with our approach. Being open source, Lucene is frequently updated and maintained by the community. Second, it is capable of performing text analysis, such as tokenisation, word stemming, and stop words removal [39]. In addition, with further alteration, it is able to parse the WordNet database to apply synonym expansion during text analysis. Lastly, by default, it builds its term indexes using *TF-IDF* weighting [39], which simplifies our work in using term weighting for calculating the similarity between query document and documents in the corpus.

3) USER INTERFACE

We adapted the interface layout guidelines from several widely known email client applications, such as Mozilla Thunderbird[2] and Opera Mail.[3] Figure 4 shows that the system returns the top ranked matched messages from our own past replies as a starting template to reply to the query message.

C. ADAPTING THE CBR IN SMART EMAIL CLIENT

This section explains how we adapted the CBR approach into our Smart Email Client prototype. Figure 5 illustrates an overall process workflow of our system after adapting CBR approach, while each phase of the CBR cycle is explained below.

1) CASE RETRIEVAL

The retrieval of similar cases starts whenever a user chooses to reply using the recommended reply option. We developed a retrieval algorithm that measures the angle of two document vectors in the multi-dimensional space.

As can be seen in Algorithm 1, the process starts when it takes on the input of a new case, which is a selected email that is about to be replied to. Lucene's native functionality to read the index file is then defined (lines 1-3). A document object, which contains the vector of terms in that particular document, is collected and normalised (lines 4-8) according to the cosine similarity formula. The similarity score of the two documents (the query document and the document in the case base) is obtained by calculating the dot product of both vectors (lines 10-14). At the end of the process, the algorithm generates a ranked list of usable past replies (lines 15 and 16). When the similarity measurement has been calculated in every case, it is ranked according to the similarity score. The higher the score, the more similar the new case is to the retrieved past cases. The next step, reusing the solution from the past case, is triggered by the user when they select a result from the recommendations list.

[1] http://lucene.apache.org/core/
[2] https://www.mozilla.org/en-US/thunderbird/
[3] http://www.opera.com/computer/mail

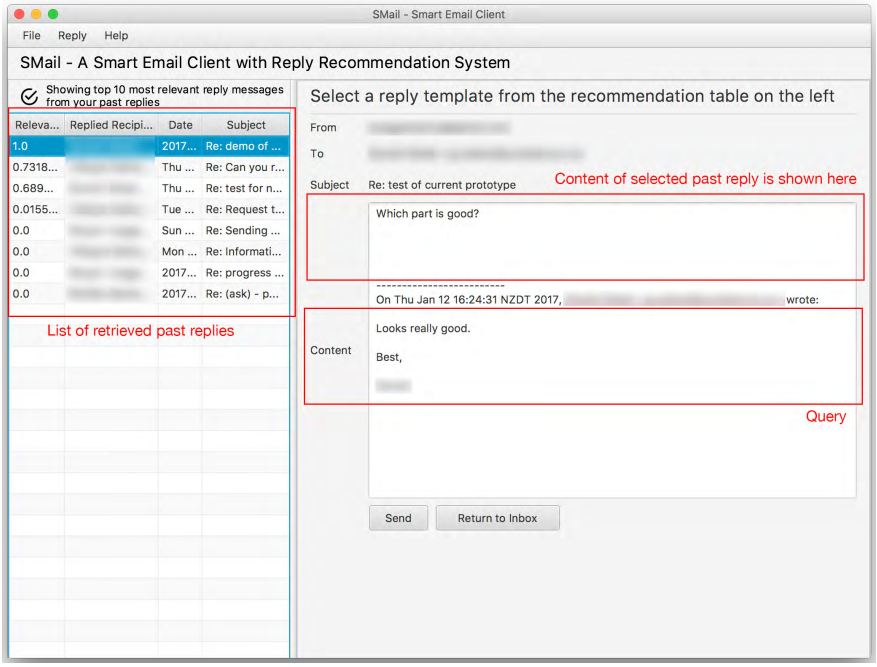


FIGURE 4. Ranked reply recommendation to be used as a starting template.

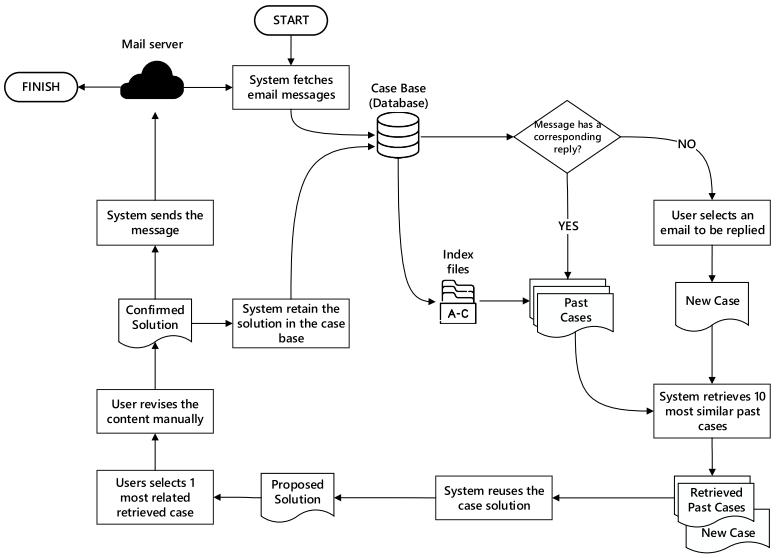


FIGURE 5. Process workflow adapting CBR methodology.

2) CASE REUSE

The process of reusing past retrieved cases comprises two aspects. First is the difference between the new and past cases. The second regards the part of the retrieved cases that can be used in the new case. In our email reply problem, consideration of these differences was partially made during the retrieval process by calculating the similarity of our case feature; the system then identifies the corresponding email reply and reuses it.

3) CASE REVISION

Our system implements case revision, which involves user intervention. The reasoning behind this is the user should have the role of direct evaluator. While partly subjective, this is the best method of matching her or his personal preference. If the user is pleased with the result, they are able to use it as is. However, if they are not, our system provides an editing feature. In this case, the user is able to revise the predefined reply message before sending the message.

Algorithm 1 Retrieving Similar Past Email (Past Cases)

Require: newCase C_n & pastCases C_p in case base Ix
Ensure: n most similar pastCases C_{pn}

initialise *indexReader*()
read Ix using *indexReader*()
initialise *caseVector*[] as **count** total C_{pn} in Ix *caseIx*
for all C_n, C_p in Ix **do**
    get term vector *caseTermVec* in Ix
    caseVector[] ← **count** *caseTermVec*
    | *caseVector*[] | ▷ Normalisation of vector
end for each
$i = 0$
for all *caseVector*[] **do**
    $cosineSim \leftarrow caseVector[C_n] \cdot caseVector[C_p]$ ▷ Dot product
    $mapRetrievedCases \leftarrow map(i, cosineSim)$
    $i++$
end for each
sort *mapRetrievedCases*
use top n $mapRetrievedCases(C_{pn})$

4) CASE RETENTION

This process is concerned with what information to store and how the solved case can be indexed for a future retrieval process. In this system, when the user sends the message, it performs two actions. First, the message is delivered to the mail server to be processed further for the recipient; then, the sent message is automatically paired with its incoming message and stored in the database. It can then be considered a solved case, so that when the retrieval phase is triggered in the future, the solved case, if it is similar to the new problem, can be indexed and retrieved.

V. IMPLEMENTATION

This section provides several screen captures of the smart email client application. We briefly explain what the user can do on the corresponding screen.

A. UNREPLIED MESSAGE SCREEN

When the user starts the application, the list of their inbox messages is shown on the left-hand side of the screen, as shown in Figure 6a. The red colour represents the unreplied messages, while the white is for replied messages. The details of the messages are shown in a split screen, where the top is for the incoming message and the bottom is for its corresponding reply. Finally, two buttons – labeled with Normal Reply or a Recommended Reply - provide options for replying.

B. NORMAL REPLY SCREEN

Figure 6b shows the screen using Normal Reply. It appears when the user clicks on the normal Reply button. In this screen, a pop-up window was built as the underlying container for other components. It has a heading, which acts as an instruction. Text labels map information about the email that is to be replied to. The editable text area is provided for typing the body message.

C. RECOMMENDED REPLY SCREEN

In addition to normal reply, the system also provides an assisted reply via the recommended reply option. When the user clicks on that option, the system starts the retrieval of similar past cases and reuses its past replies so they are presented in a ranked list, as shown in Figure 6c. The relevance column shows the similarity of the corresponding past reply; the higher the score, the more relevant it is. Whenever the user selects a result from the recommendation table, the content is prepopulated automatically in the reply box. This allows the user to see the content of their past replies, as illustrated in Figure 6d. If the user is satisfied with the content, they can send the message directly. Furthermore, every change in selection automatically changes the populated content, while the details from the origin email that is about to be replied to is still preserved. However, it is also possible to add more or to revise the content according to the user's preference. The editing process is shown in Figure 6e.

D. REPLIED MESSAGE SCREEN

At the end of the process, after the user has clicked the send button, the system delivers the message to the mail server. In addition, the system also retains the reply message as a solution, making the case solved as shown in Figure 6f.

VI. EVALUATION

The first part of this section provides our experimental design for evaluating our approach. This includes evaluation method, evaluation metric, dataset description, and hardware specifications, while the second part presents our results with their illustration.

A. EXPERIMENTAL DESIGN

1) HARDWARE SPECIFICATIONS

The experiments were performed on a machine with the following specifications: 4.6GHz Intel Core i5 with 16GB RAM memory, 500GB solid-state drive (SSD), and running Windows 10. The evaluation module was implemented in Java SE version 8.

2) DATASET DESCRIPTION

In order to evaluate our system in conditions closer to a real-world scenario, we considered the following components to find the most appropriate dataset. First, since we are dealing with email problems, it is best to use a dataset of email messages. However, due to privacy issues around the nature of email communication, it was challenging to find a publicly available email dataset. Until recently, there was only one realistic email dataset available and that has been widely used for research purposes. The dataset was collected from Enron, a company that went bankrupt. Originally, this dataset was

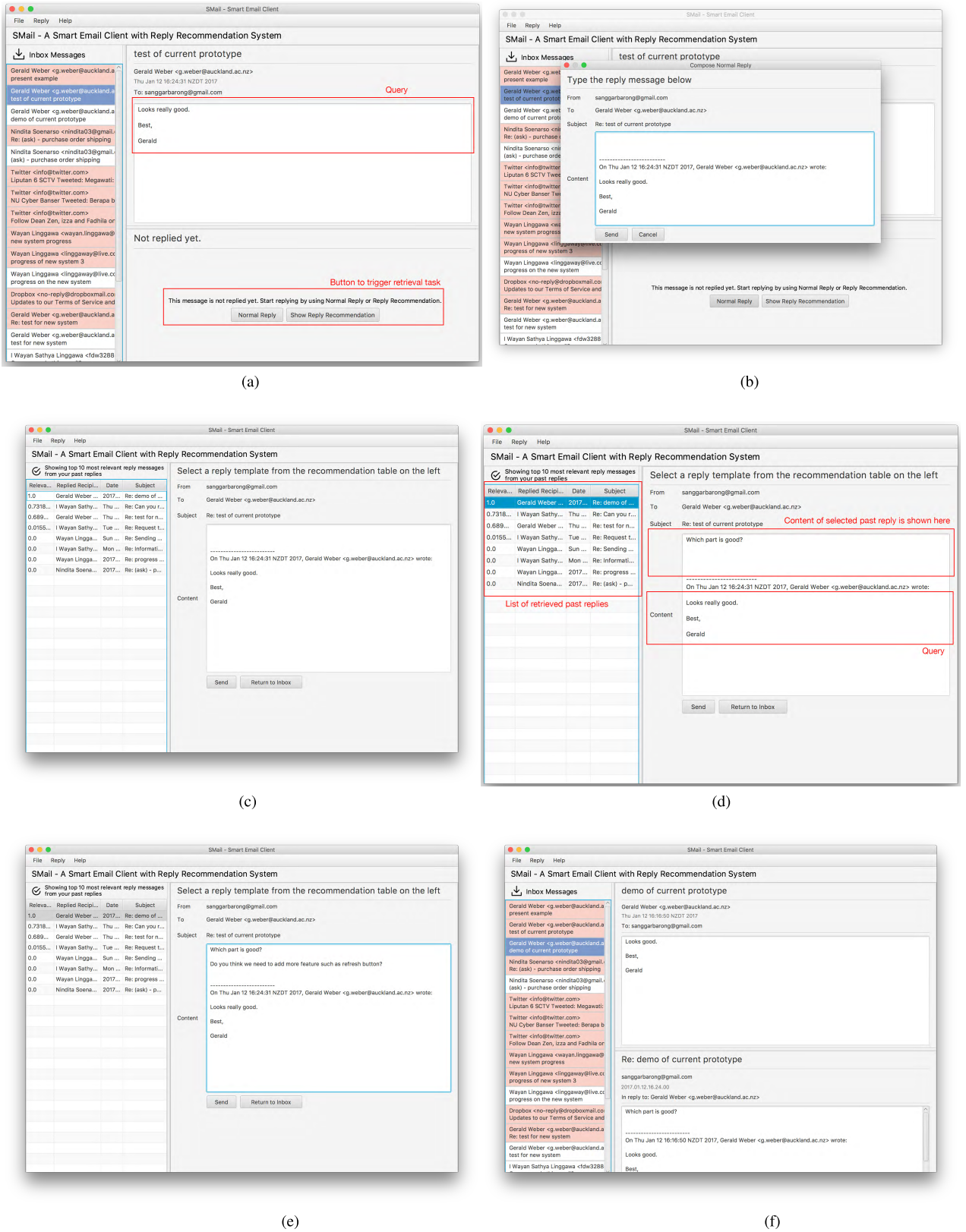


FIGURE 6. System screens. (a) Unreplied message screen. (b) normal reply screen. (c) recommended reply screen. (d) recommended reply selected. (e) reply revision. (f) solved query.

published during an investigation into the bankruptcy; after the data was compiled and cleaned up, it was made available[4] to be used in research [40].

Interestingly, a study has been done that used a subset of the Enron dataset and annotated it [41]. Initially, the authors used this dataset for person name disambiguation and an intelligent email-threading task. While they examined both email headers (e.g. recipients, date, and subject) and email content, we only used the subject and the content in our evaluation. Mailbox from two Enron employees was prepared: `germany-c` and `farmer-d`. Further details about the datasets are presented in Table 1.

TABLE 1. Datasets descriptions.

Description	germany-c	farmer-d
Count of email messages	2651	2642
Count of annotated email pairs	81	127
Average word count per message	48.36	53.38
(Field: Subject) Blank messages	91 (3.4%)	41 (1.6%)
(Field: Subject) Total word count	10018	11916
(Field: Body) Blank messages	276 (10.4%)	289 (10.9%)
(Field: Body) Total word count	113778	121514

3) EVALUATION METHODS

Retrieval phase is the first step in the CBR cycle; an effective retrieval process ensures the system generates the most similar past cases with respect to the given problem. We identified three objectives in evaluating the retrieval system. First, it is important that our system returns only the most relevant results to satisfy the user needs, according to the user query. This is known as retrieval effectiveness [42]. Second, the more cases stored in the case base, the bigger the index size becomes. Since cases are indexed using the terms, it seems important to maintain the size of the index by only storing the most representative terms. Therefore, we also measure the index size of the system. Third, it is also essential to provide a seamless response, as the user triggers an ad-hoc retrieval using a query. Thus, we record the time required to process a query. While this measurement may vary according to the hardware specifications, the results generated can still provide a general notion of the system speeds. This is further considered as retrieval efficiency [42].

4) EVALUATION METRICS

To measure the effectiveness of the retrieval process, we used Reciprocal Rank (RR). This is a relative score that calculates the average or mean of the inverse of the ranks at which the first relevant document was retrieved for a set of queries. For instance, if a search for a specific query returns a relevant document at the 1^{st} position, its relative rank or RR is 1. If the relevant document is at position 2, then the score is 0.5 and so on. If there are no relevant documents, the score is 0. When averaged across the set of queries, this measure is called the Mean Reciprocal Rank (MRR). It is associated with the use case where the user wishes to see only one relevant document for a search, subsequently assuming the user will keep scrolling down on the search results until the first relevant document is found. Thus, the document is found at rank n, and the quality of the retrieval is measured by the reciprocal of the rank (i.e. $1/n$).

We observed that each user query has one correct answer and the assumption is that the user will stop searching once the correct document, based on his/her preference, is found. Alternatively, the time required by the user to find the relevant document corresponding to the search is inversely proportional to the rank. In this case, the better the rank, the lower the time taken by the user to get to the relevant document. For instance, an MRR score of 0.8 indicates that the information retrieval system is 80% relevant.

In addition, Mean Average Precision (MAP) is also a popular scoring method used in measuring the effectiveness of an information retrieval system [43]. It is used mostly when more than one possible relevant result is expected for their search query [43]. However, in our system, since we assume the user chooses only the most relevant answer from the provided recommendation list, we excluded this metric in our evaluation.

B. RESULTS

In this section, we presented the results of our experimentation. As mentioned in the previous section, we described the results based on three indicators we measured in our evaluation: the quality of retrieval results, index size, and processing time elapsed. However, before starting to present our experimental results, we performed an initial analysis to see the distribution of the terms in our dataset. To do this, the email messages were tokenised to obtain the collection of words, which were stored as index terms, and then counted and ranked with respect to the frequency of occurrence in each document in the corpus. We only kept terms that had a frequency of at least two in the corpus. We considered trimming this to provide better distribution visualisation.

The term *frequency calculation* was obtained using Luke,[5] an open source GUI tool that examines the Lucene index files. The interface of the tool is shown in Figure 7. We can select the feature indexed, either subject or body, and copy the results from the table on the right-hand side of the pane. The term frequeTF-cy obtained is meaningful for two reasons: first, we saw that some of the most frequent terms that occur in the corpus are stop words, so we considered examining the stop words removal influence in our experiment. Second, we compared our term distribution with Zipfian Law [44], which is also commonly referred to as Zipf's distribution. In the implementation of this distribution in the linguistics domain, the frequency of a word is inversely proportional to its rank in the frequency count. A study by [45] demonstrated Zipf's distribution in a corpus using a graph. We also

[4]http://www.cs.cmu.edu/ einat/datasets.html

[5]https://github.com/DmitryKey/luke

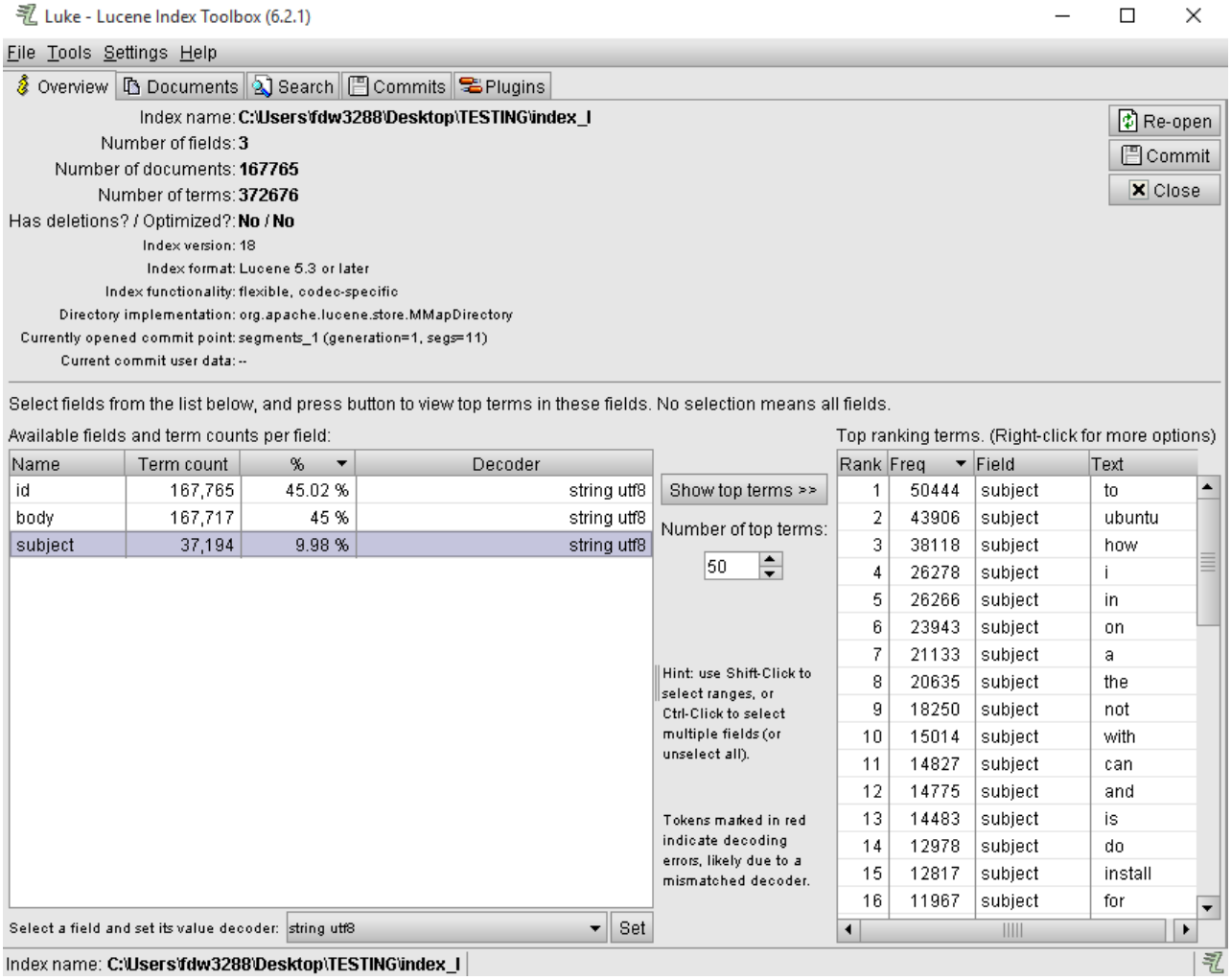


FIGURE 7. Luke: An open source tool to read Lucene index files.

attempted to observe the data according to this distribution by plotting it on a log-log graph, where each log represents log (rank order) in the y-axis and log (frequency) in the x-axis.

It can be seen from Figure 8 that the term distribution is a near fit to Zipf's Law. The deviation seems to appear in the most frequent words since the plot merges with the trend line at half way. Both datasets show similar behaviour, while slight differences are only present between the log (rank) of 2 and 4.

1) RETRIEVAL EFFECTIVENESS

The retrieval effectiveness of our algorithm was measured by comparing the retrieved results with the relevance judgments given for each corresponding query, as annotated by [41]. The MRR score reflects the retrieval effectiveness according to the score. The higher the score, the more relevant the results retrieved by our algorithm.

Furthermore, Table 2 presents the detailed MRR score from 18 different trials. Overall, the highest MRR score was achieved by applying the `SA` method to the case feature subject. Then, combining the features subject and body as all during retrieval still yields a higher MRR score when compared to that in the body only. It is interesting to see the `SA_NOSTM` method gives a better score when implemented in all case features.

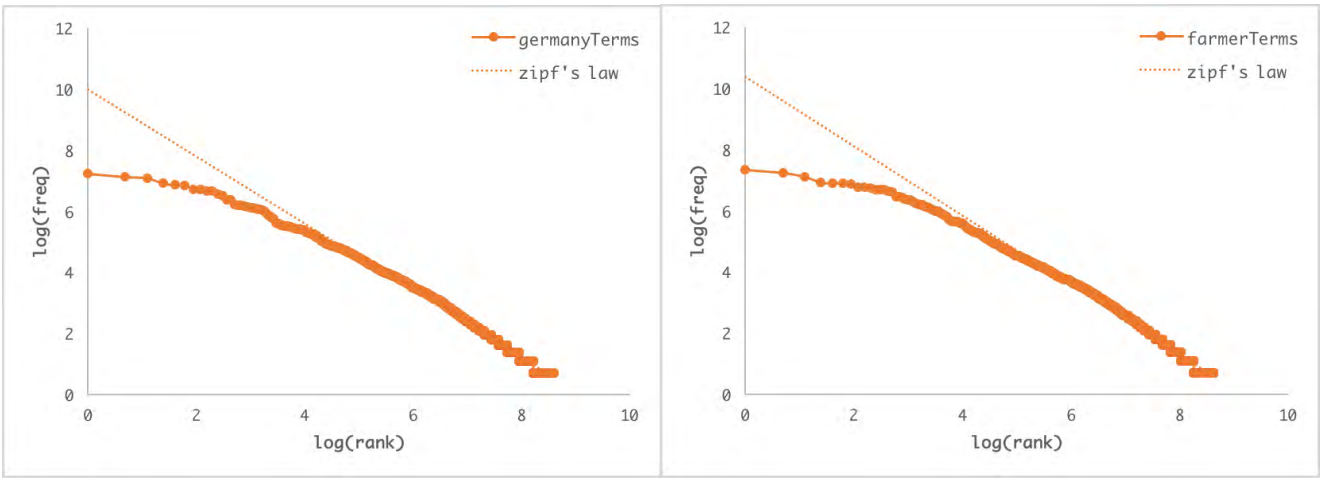
In order to provide better visualisation for comparison, Figure 9 depicts the summary of the results in a chart. It is apparent from this chart that the MRR score shows a decreasing trend when the `SE` method is applied in the trials, in spite of the stemming and stop words removal techniques. Moreover, the results also affirm the case feature subject returns more relevant emails that match the relevance judgments. This is followed by all and the body.

While the MRR score was used as the primary metrics for measuring retrieval effectiveness, we also examined the cosine similarity score generated from matching a query with documents in the corpus. This examination was performed for both datasets, `farmer-d` and `germany-c`, to determine

TABLE 2. MRR score from experiments in both dataset.

MRR score	SA[1]	SE[2]	SA_ NOSTM[3]	SE_ NOSTM[4]	SA_ NOSWR[5]	SE_ NOSWR[6]
Lexical analysis	✓	✓	✓	✓	✓	✓
Stopwords removal	✓	✓	✓	✓		
Stemmer	✓	✓			✓	✓
Synonym expansion		✓		✓		✓
Germany-C						
All	0.2970	0.1026	**0.3326**	0.1287	0.2126	0.1177
Body	**0.1126**	0.0592	0.1146	0.0680	0.0416	0.0491
Subject	**0.7881**	0.5741	0.7757	0.6008	0.7716	0.5936
Farmer-D						
All	0.1610	0.0608	**0.2627**	0.0719	0.1591	0.0618
Body	0.0177	0.0234	**0.0457**	0.0195	0.0245	0.0209
Subject	**0.8543**	0.6647	0.8340	0.6890	0.7940	0.6604

[1]Standard analysis, [2]Synonym expansion
[3]Standard analysis without stemmer, [4]Synonym expansion without stemmer
[5]Standard analysis without stopwords removal, [6]Synonym expansion without stopwords removal



(a) (b)

FIGURE 8. Dataset distribution according to Zipf's Law distribution. (a) germany-c. (b) farmer-d.

the influence of synonym expansion. Thus, SA represents results from standard analysis techniques, while SE enhances standard analysis SA with synonym expansion. We particularly observed the trends from the top 10 retrieved results with the highest cosine similarity score on two case features: body and subject.

The observations from Figure 10 show both datasets provided a similar decreasing trend in their respective body and subject features as the ranking goes lower from 1 to 10. This decreasing trend seems linear in the case feature body. However, the score in the case feature subject, as depicted in Figures 10b and 10d, shows gradually higher disparities between SA and SE compared to the results in Figures 10a and 10c. Thus, we could conclude that synonym expansion SE outperforms the standard analysis SA techniques in terms of the cosine similarity score. As a result, it could generate a higher chance of relevant matching, as shown in the higher score when compared to SA.

According to the results presented previously, it can be seen that the decreasing trends occur over the higher rank, despite the case features. This is possibly due to the sorting of the retrieval results according to the highest similarity score. However, it also seems interesting to examine the percentage of increase before and after applying synonym expansion.

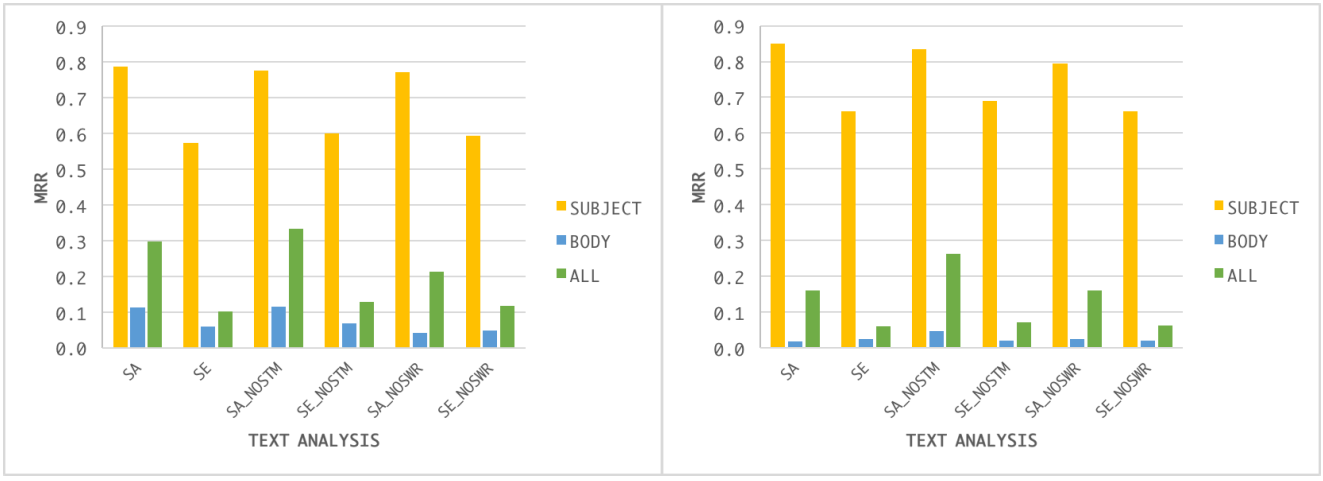


FIGURE 9. Retrieval effectiveness as represented by MRR score. (a) germany-c. (b) farmer-d.

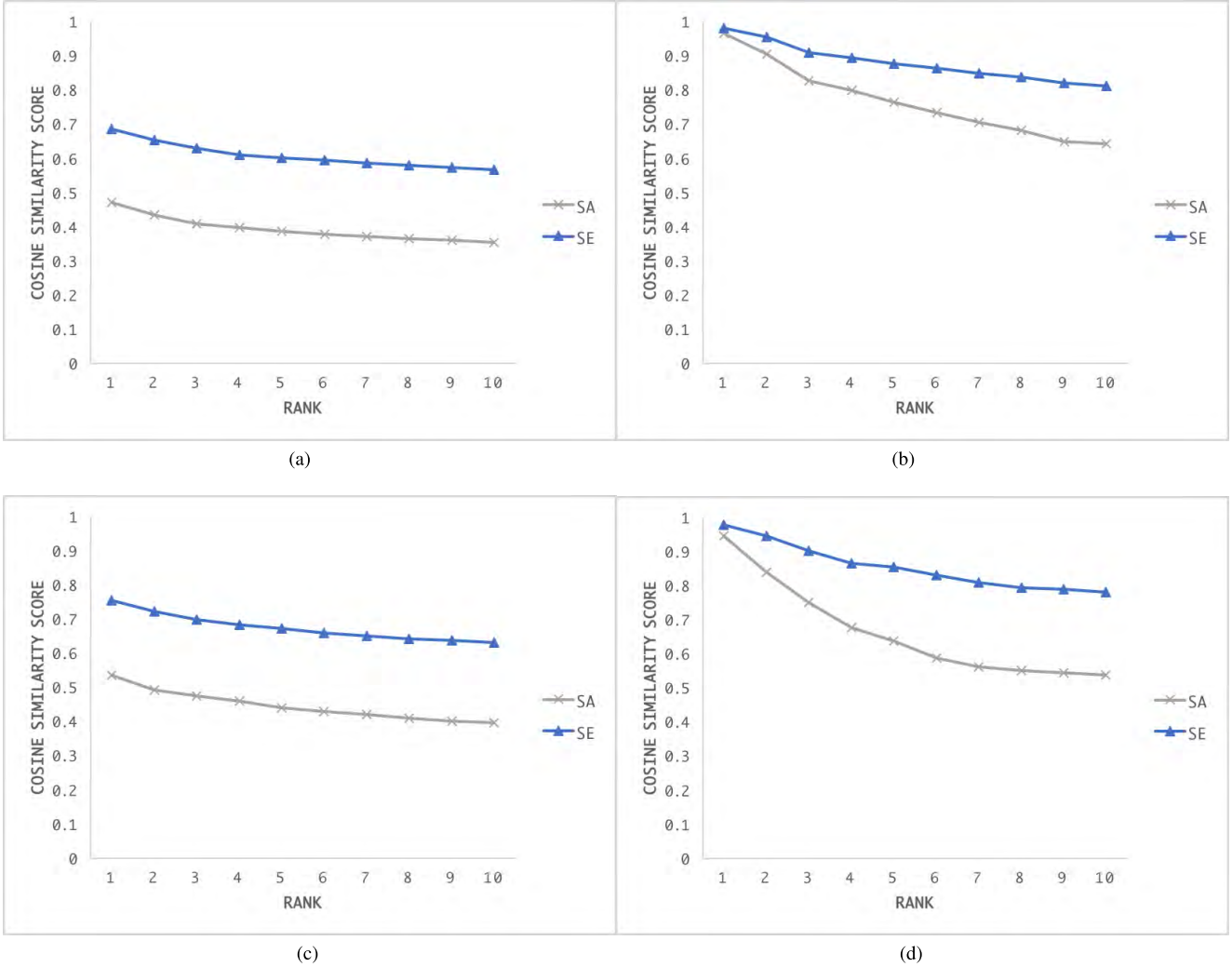


FIGURE 10. Average similarity score over top 10 results in both germany-c and farmer-d datasets. (a) germany-c body. (b) germany-c subject. (c) farmer-d body. (d) farmer-d subject.

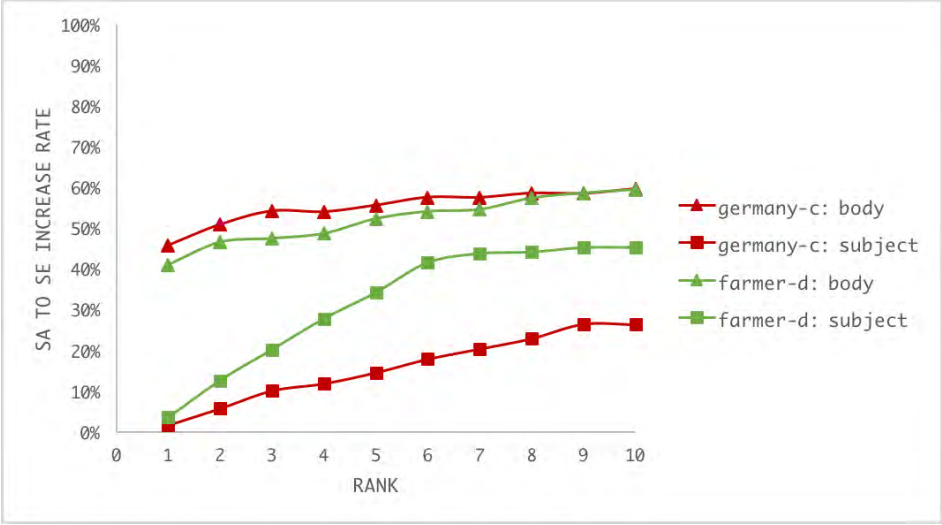


FIGURE 11. Similarity score increase rate before (SA) and after (SE) applying synonym expansion with respect to case features body and subject.

This increase is calculated from cosine similarity score of SA and SE. Given the same query, when the score of SA and SE is 0.5 and 0.75 respectively, the rate of increase 50%.

In contrast to the decreasing trend over the higher ranks in the cosine similarity score, Figure 11 depicts that nearly 40-60 percent of the increasing rate appears in the body feature, with a relatively low disparity between both datasets. However, this disparity gradually increases in the subject feature. This validates our previous findings that synonym expansion could improve the chance of retrieving more relevant matching, even in the lower ranks, by increasing the cosine similarity score up to 30 percent in the subject and up to 60 percent in the body.

Although SE could increase the cosine similarity score, its MRR score shows the opposite. This might be related to the relevance judgments provided by [41]. Since it is highly dependent on human annotation, it might affect the MRR score. This is further explained in the discussion section.

2) INDEX SIZE

In addition to measuring the retrieval effectiveness, we also examined the influence of text analysis and case feature selection on the size of the index. According to Table 3, the SE_NOSTM method generated the highest number of terms over all case features. In addition, the case feature *all* had the most terms, whereas the subject seemed to have the least.

It is also shown in Figure 12 that a significant amount of terms are generated by synonym expansion SE method. In addition, by not applying the stemming method, as shown in SA_NOSTM and SE_NOSTM, more terms are generated compared to other methods.

3) RETRIEVAL EFFICIENCY

Finally, we also measured the retrieval efficiency as represented by the processing time elapsed for executing a retrieval process in each query. We argued in the previous section that although the processing time might be highly dependent on the hardware used, we can still compare the influence of the different text analyses performed in the trials. We obtained the results by averaging the processing times from three trial repetitions. This processing time was generated using the Java *nanotime* library, which provided a more precise calculation as it was captured within nanoseconds.

In Figure 13, it is clear the retrieval process that used the case feature subject was the fastest above all other features. In addition to performing synonym expansion, as shown in any SE methods, it uses a slightly slower processing time. Overall, applying stemming and stop words removal does not seem to significantly affect the processing time.

VII. DISCUSSION

In this section, we provide a further justification of our results by exploring the influence of text analysis, case feature selection, and the dataset.

A. INFLUENCE OF TEXT ANALYSIS

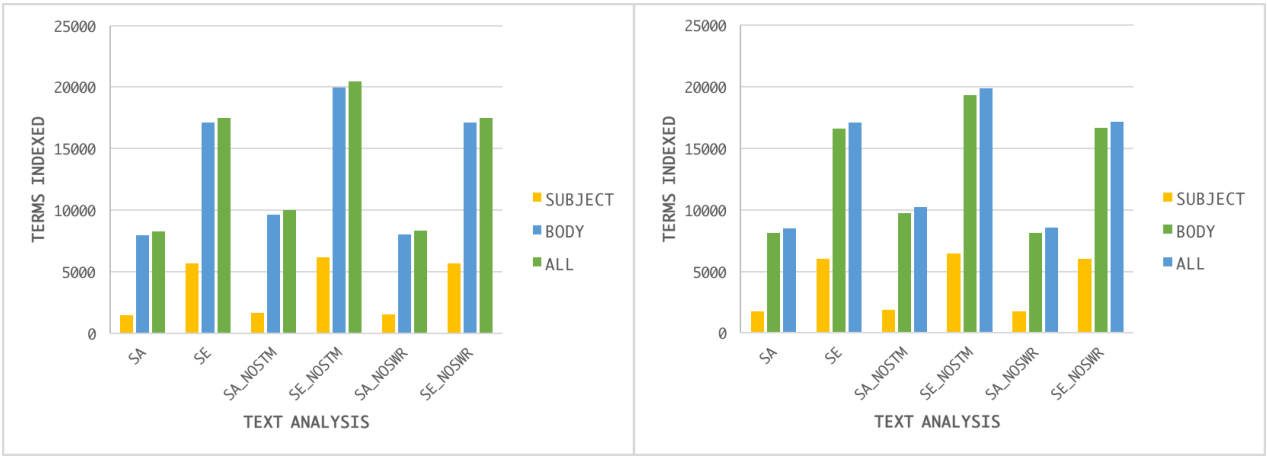
During the experiment, we performed trials using several text analysis techniques. First was the lexical analysis. This technique removes whitespace, symbols, and punctuation marks e.g. apostrophe (’), a dot (.), and a slash (/). As all the words in the content were transformed to lowercase, there is a likelihood of increasing the chance of matching that particular word. This normalisation is typically useful when an abbreviation is misspelt, for instance, *EtNG* instead of *ETNG*.

While lexical analysis technique allows terms to be *treated* as equal, it can be seen that the chance of matching could be improved by applying the second technique, stemming. For instance, the word *telling* could match *tell* if the suffixing is chopped. Another example of stemming is where the term *handling* can match *handle*. The new term *handl* seems to

TABLE 3. Total terms indexed from experiments in both dataset.

Terms indexed	SA[1]	SE[2]	SA_ NOSTM[3]	SE_ NOSTM[4]	SA_ NOSWR[5]	SE_ NOSWR[6]
Lexical analysis	✓	✓	✓	✓	✓	✓
Stopwords removal	✓	✓	✓	✓		
Stemmer	✓	✓			✓	✓
Synonym expansion		✓		✓		✓
Germany-C						
All	8292	17474	9994	**20456**	8325	17507
Body	7999	17101	9622	**19978**	8032	17134
Subject	1482	5678	1651	**6187**	1512	5709
Farmer-D						
All	12060	17110	10200	**19864**	8561	17143
Body	11214	16618	9746	**19296**	8154	16651
Subject	2317	6027	1898	**6455**	1766	6052

[1]Standard analysis, [2]Synonym expansion
[3]Standard analysis without stemmer, [4]Synonym expansion without stemmer
[5]Standard analysis without stopwords removal, [6]Synonym expansion without stopwords removal



(a) (b)

FIGURE 12. Index size as represented in a count of terms indexed. (a) germany-c. (b) farmer-d.

be an unknown English word for human, but the computer could process it. It should be considered however, that the stemming process would typically be done after the synonym matching process; otherwise, the chance of matching a word with its synonym pairs might be less likely. The stemmed word *handl* might have no match, but *handling* might.

Furthermore, not all the terms should be indexed. There are set of words that are likely to appear frequently in textual content, and these are commonly identified as stopwords. While humans have no issue with this, a computer considers these words as less rare in term weighting if they occur too often. Interestingly, we found applying lexical analysis and stopwords removal generally improved the number of relevant retrieved emails, as shown in the increase of the MRR score. It seems possible that the removal of these words could reduce the noise in the content, and might increase the chance of distinct terms to be fairly calculated while comparing between two documents.

Another text analysis technique used in evaluation is synonym expansion. This technique is performed by matching terms with their corresponding synonyms. In our experiment, we utilised WordNet as additional linguistic knowledge, and this provided the database of semantic relations. In an early observation, we found this technique could extend the capability of exact matching by searching through a word's synonym pairs.

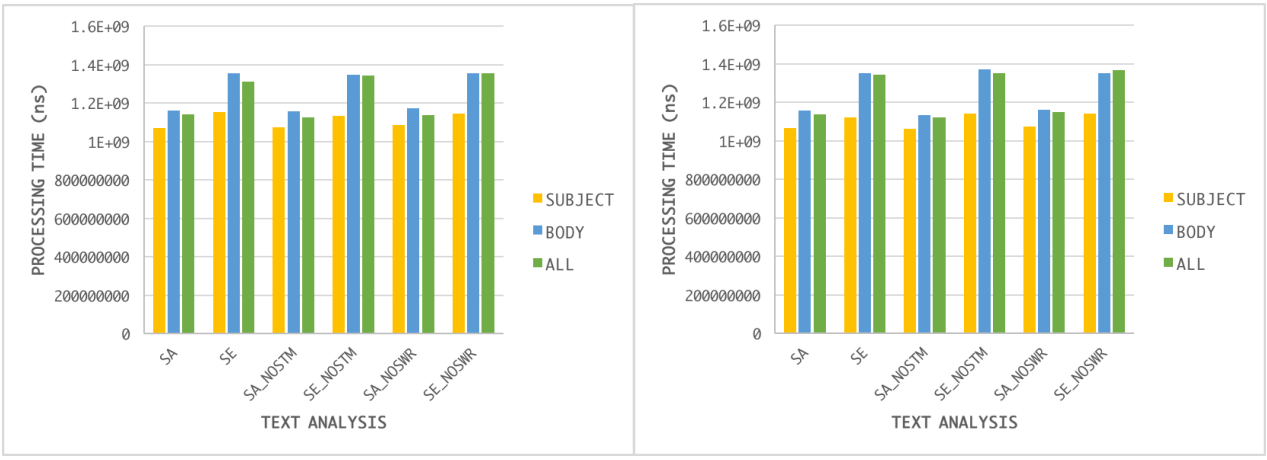


FIGURE 13. Retrieval efficiency as represented in processing time elapsed. (a) germany-c. (b) farmer-d.

The next observation shows a trend where experiments that use synonym expansion tend to show an increase in the retrieval execution time for each query. This might be due to the larger number of terms to be calculated between the documents. The increasing number of terms occurred because the synonym words were also considered in the similarity calculation.

B. FEATURE SELECTION

We also observed that different selected features could affect the retrieval of relevant messages. By using the case feature subject, the retrieval effectiveness yielded a higher score compared to when the body was used. There could be several possible explanations for this result. First, the subject line is likely to have fewer words, which might lead to a higher chance of matching keywords. Second, we found the subject content contains a relatively uniform pattern. It is a trivial problem for the algorithm if it only considers the subject, because of the high number of exact matching words. As can be seen from the experiment results, it is challenging for the case feature body to generate a higher MRR score since our retrieved email did not usually match the relevance judgments. However, our observations showed that applying the synonym expansion technique could increase the chance of matching relevant results, even in the lower rank. This finding is similar to [4], who argues that synonym expansion leads to an improvement in the similarity measurement. Hence, we presume this might be related to the collection method implemented by [41] when building the relevance judgments. In the next section, we discuss our justifications for this issue.

C. DATASET CRITIQUE

In Section VI-A we talked about the challenge of performing an evaluation using a real-life email dataset due to the privacy concerns. However, we overcame this by discovering that the Enron email dataset is available online. This dataset came in a collection of raw messages, until some researchers annotated them for study purposes [41].

When we performed an evaluation using this dataset, we found the results from the subject field tended to be high, as seen in the MRR score when compared to that of the body feature. We presume the process of performing a relevance judgment might be to consider the small number of features instead of all features in the email message. Then we found, according to the authors, the relevant messages were annotated using the subject line and time stamp only [41]. This also explains why using the feature all still generated a higher result than the body. This could be due to the combination of the content subject and body, but it still provides a higher chance of matching the relevance judgments.

We also noticed in a number of messages, the subject line contains fewer meaningful words in terms of context. We consider this as less meaningful since they have an empty value or are only one word with three or fewer characters. While it might reflect the reality of email communication, where people are likely to reply using the same subject, it is challenging in a retrieval task, which considers the subject only. Thus, it is important to consider another field such as the body itself.

Finally, we concluded that this dataset is more appropriate for finding a relevant email by considering the email header, such as sender, recipient, date, and time. The subject field is also suitable for use as a case retrieval feature. However, we found the body field is less useful since the annotation process might not consider this field.

VIII. CONCLUSIONS AND FUTURE WORK

The paper presented a novel prototype for Smart Email Client based on a problem-solving framework called Case-Based Reasoning (CBR). We built a retrieval algorithm that finds similar cases beyond exact matching by using text processing and semantics analysis techniques. The algorithm intelligently retrieves similar past queries and

their respective replies according to the incoming query. We evaluated our prototype rigorously based on three key parameters: the quality of retrieval results, index size, and processing time elapsed. We used two Enron employees mailbox data (`germany-c` and `farmer-d`) as our dataset. We have shown that the results we achieved under all three parameters are promising.

Future development includes performing an automatic name extraction using the information extraction technique; this information could be used at the beginning of the reply message as the salutation. On the other hand, the name-based entity can be omitted from the reply message in order to provide a more generic reply message. We also have aim to test our email client in ICT department of our university Which deals with the IT related issues of staff and students. We will validate the accuracy of predicted replies with the original replies sent by the operator.

REFERENCES

[1] S. Radicati and Q. Hoang, "Email statistics report, 2015-2019," Radicati Group, Palo Alto, CA, USA, Tech. Rep., 2015.
[2] L. Lamontagne and G. Lapalme, "Applying case-based reasoning to email response," in *Proc. 5th Int. Conf. Enterprise Inf. Syst. (ICEIS)*, 2003, pp. 115–123.
[3] A. Hliaoutakis, G. Varelas, E. Voutsakis, E. G. Petrakis, and E. Milios, "Information retrieval by semantic similarity," *Int. J. Semantic Web Inf. Syst.*, vol. 2, no. 3, pp. 55–73, 2006.
[4] K. Abdalgader and A. Skabar, "Short-text similarity measurement using word sense disambiguation and synonym expansion," in *Proc. Australas. Joint Conf. Artif. Intell.* Springer, 2010, pp. 435–444.
[5] A. Stahl and T. R. Roth-Berghofer, "Rapid prototyping of CBR applications with the open source tool myCBR," in *Proc. Eur. Conf. Case-Based Reasoning*. Springer, 2008, pp. 615–629.
[6] B. Lopez, "Case-based reasoning: A concise introduction," *Synth. Lectures Artif. Intell. Mach. Learn.*, vol. 7, no. 1, pp. 1–103, 2013.
[7] I. W. S. Linggawa, M. A. Naeem, R. Pears, and G. Weber, "Reusing past replies to respond to new email: A case-based reasoning approach," M.S. thesis, Dept. Master Comput. Inf. Sci., Univ. Auckland, Auckland, New Zealand, 2017.
[8] K. Coussement and D. Van den Poel, "Improving customer complaint management by automatic email classification using linguistic style features as predictors," *Decis. Support Syst.*, vol. 44, no. 4, pp. 870–882, 2008.
[9] C. Hendahewa and C. Shah, "Evaluating user search trails in exploratory search tasks," *Inf. Process. Manage.*, vol. 53, no. 4, pp. 905–922, 2017.
[10] E. Sneiders, "Review of the main approaches to automated email answering," in *New Advances in Information Systems and Technologies*. Springer, 2016, pp. 135–144.
[11] L. Kosseim, S. Beauregard, and G. Lapalme, "Using information extraction and natural language generation to answer e-mail," *Data Knowl. Eng.*, vol. 38, no. 1, pp. 85–100, 2001.
[12] R. Malik, L. V. Subramaniam, and S. Kaushik, "Automatically selecting answer templates to respond to customer emails," in *Proc. IJCAI*, vol. 7, 2007, pp. 1659–1664.
[13] S.-S. Weng and C.-K. Liu, "Using text classification and multiple concepts to answer e-mails," *Expert Syst. Appl.*, vol. 26, no. 4, pp. 529–543, 2004.
[14] A. Kannan *et al.*, "Smart reply: Automated response suggestion for email," in *Proc. KDD*, 2016, pp. 955–964.
[15] C. Van Gysel *et al.*, "Reply with: Proactive recommendation of email attachments," in *Proc. ACM Conf. Inf. Knowl. Manage. (CIKM)*. New York, NY, USA, 2017, pp. 327–336, doi: 10.1145/3132847.3132979.
[16] M. Sappelli, G. Pasi, S. Verberne, M. de Boer, and W. Kraaij, "Assessing e-mail intent and tasks in e-mail messages," *Inf. Sci.*, vols. 358–359, pp. 1–17, Sep. 2016.
[17] E. Sneiders, J. Sjöbergh, and A. Alfalahi, "Email answering by matching question and context-specific text patterns: Performance and error analysis," in *New Advances in Information Systems and Technologies*. Springer, 2016, pp. 123–133.
[18] T. Ayodele and S. Zhou, "Applying machine learning techniques for e-mail management: Solution with intelligent e-mail reply prediction," *J. Eng. Technol. Res.*, vol. 1, no. 7, pp. 143–151, 2009.
[19] T. Ayodele and S. Zhou, "Applying machine learning algorithms for email management," in *Proc. 3rd Int. Conf. Pervasive Comput. Appl. (ICPCA)*, vol. 1, Oct. 2008, pp. 339–344.
[20] P. Goswami, E. Gaussier, and M.-R. Amini, "Exploring the space of information retrieval term scoring functions," *Inf. Process. Manage.*, vol. 53, no. 2, pp. 454–472, 2017.
[21] M. Dredze, T. Brooks, J. Carroll, J. Magarick, J. Blitzer, and F. Pereira, "Intelligent email: Reply and attachment prediction," in *Proc. 13th Int. Conf. Intell. User Interfaces*, 2008, pp. 321–324.
[22] L. Yang, S. T. Dumais, P. N. Bennett, and A. H. Awadallah, "Characterizing and predicting enterprise email reply behavior," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, Tokyo, Japan, 2017, pp. 235–244.
[23] G. Lapalme and L. Kosseim, "Mercure: Towards an automatic e-mail follow-up system," *IEEE Comput. Intell. Bull.*, vol. 2, no. 1, pp. 14–18, Jun. 2003.
[24] T. Mikolov, K. Chen, G. Corrado, and J. Dean. (2013). "Efficient estimation of word representations in vector space." [Online]. Available: https://arxiv.org/abs/1301.3781
[25] S. Clinchant and F. Perronnin, "Aggregating continuous word embeddings for information retrieval," in *Proc. Workshop Continuous Vector Space Models Compositionality*, 2013, pp. 100–109.
[26] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning deep structured semantic models for Web search using clickthrough data," in *Proc. 22nd ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2013, pp. 2333–2338.
[27] I. Vulić and M.-F. Moens, "Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2015, pp. 363–372.
[28] D. Ganguly, D. Roy, M. Mitra, and G. J. Jones, "Word embedding based generalized language model for information retrieval," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2015, pp. 795–798.
[29] G. Zuccon, B. Koopman, P. Bruza, and L. Azzopardi, "Integrating and evaluating neural word embeddings in information retrieval," in *Proc. 20th Australas. Document Comput. Symp.*, 2015, p. 12.
[30] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 957–966.
[31] T. Kenter and M. De Rijke, "Short text similarity with word embeddings," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 1411–1420.
[32] B. Mitra, E. Nalisnick, N. Craswell, and R. Caruana. (2016). "A dual embedding space model for document ranking." [Online]. Available: https://arxiv.org/abs/1602.01137
[33] Q. Ai, L. Yang, J. Guo, and W. B. Croft, "Improving language estimation with the paragraph vector model for ad-hoc retrieval," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2016, pp. 869–872.
[34] H. Zamani and W. B. Croft, "Embedding-based query language models," in *Proc. ACM Int. Conf. Theory Inf. Retr.*, 2016, pp. 147–156.
[35] H. Zamani and W. B. Croft, "Estimating embedding vectors for queries," in *Proc. ACM Int. Conf. Theory Inf. Retr.*, 2016, pp. 123–132.
[36] W. R. Hewlett and M. Freed, "An email assistant that learns to suggest reusable replies," in *Proc. AAAI Workshop*, 2008, pp. 28–35.
[37] I. Watson, "Case-based reasoning is a methodology not a technology," *Knowl.-Based Syst.*, vol. 12, nos. 5–6, pp. 303–308, 1999.
[38] M. Lenz, B. Bartsch-Spörl, H.-D. Burkhard, and S. Wess, *Case-Based Reasoning Technology: From Foundations to Applications*, vol. 1400. Springer, 2003.
[39] A. Białecki, R. Muir, G. Ingersoll, and L. Imagination, "Apache lucene 4," in *Proc. Workshop Open Source Inf. Retr. (SIGIR)*, 2012, p. 17.
[40] B. Klimt and Y. Yang, "The enron corpus: A new dataset for email classification research," in *Proc. Eur. Conf. Mach. Learn.* Springer, 2004, pp. 217–226.
[41] E. Minkov, W. W. Cohen, and A. Y. Ng, "Contextual search and name disambiguation in email using graphs," in *Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2006, pp. 27–34.
[42] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan, "Expected reciprocal rank for graded relevance," in *Proc. 18th ACM Conf. Inf. Knowl. Manage.*, 2009, pp. 621–630.
[43] E. M. Voorhees, "Variations in relevance judgments and the measurement of retrieval effectiveness," *Inf. Process. Manage.*, vol. 36, no. 5, pp. 697–716, 2000.

[44] M. E. J. Newman, "Power laws, Pareto distributions and Zipf's law," *Contemp. Phys.*, vol. 46, no. 5, pp. 323–351, Dec. 2005.
[45] L. Q. Ha, E. I. Sicilia-Garcia, J. Ming, and F. J. Smith, "Extension of Zipf's law to words and phrases," in *Proc. 19th Int. Conf. Comput. Linguistics*, 2002, pp. 1–6.

M. ASIF NAEEM received the master's (Hons.) and Ph.D. degrees in computer science from The University of Auckland, New Zealand. He is currently the Director of the Data Science Research Group and a Senior Lecturer with the School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland, New Zealand. He has about 15 years research, industrial, and teaching experience. He has published over 50 research papers in high repute journals, conferences, and workshops in his area. His recent research has been published in *Information Systems* and in the *Journal of Computational and Applied Mathematics* which are ranked A* and A, respectively, in Computing Research and Education Association. His research interests are data stream processing, real-time data warehousing, big data management, knowledge engineering, and data science. He received the Best Ph.D. Thesis Award from The University of Auckland. He has been reviewing for well-known journals and conferences in his area. He has been organizing an IEEE workshop IWDS since 2013.

I. WAYAN S. LINGGAWA received the master's degree from the School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland, New Zealand. His research interest includes information processing, information retrieval, and case-based reasoning. He received A+ in his master's Thesis. His master's study was funded by the Government of Indonesia under the commonwealth scholarship scheme.

AFTAB A. MUGHAL received the master's degree from the Blekinge Institute of Technology, Sweden, and the Ph.D. degree from the University of Otago, New Zealand. He is currently an IT Program Leader with AGI Education Limited, Auckland, New Zealand. He has published a number of papers in well reputed journals and conferences in his area. His research interest includes information management, software engineering, and databases.

CHRISTOF LUTTEROTH is currently a Senior Lecturer with the Department of Computer Science, University of Bath, U.K. He has received Marsden Fund, one of the prestigious grants in New Zealand. He has published more than 100 research articles and conference publications. His research interests are in the field of human–computer interaction, with a focus on eye-gaze interaction and health. He is researching new ways to enable users to control electronic devices using their eyes and new applications of virtual reality in healthcare. He is reviewing well-known journals and conferences in his area, including IEEE EDOC and IEEE INDIN.

GERALD WEBER received the Ph.D. degree from The Free University of Berlin. He joined The University of Auckland in 2003. He is currently a Senior Lecturer with the Department of Computer Science, The University of Auckland. He is also the Information Director of the *Proceedings of the VLDB Endowment*. He has co-authored the book *Form-Oriented Analysis*, and of over 40 peer-reviewed publications. His research interests include databases and data models, human–computer interaction and theory of computation. He has been a program chair of several conferences.