Received: 8 March 2016 | Revised: 2 August 2017 | Accepted: 8 September 2017

DOI: 10.1111/exsy.12251

WILEY Expert Systems

SPECIAL ISSUE PAPER

Automated email answering by text-pattern matching: Performance and error analysis

Eriks Sneiders[1] | Jonas Sjöbergh[2] | Alyaa Alfalahi[1]

[1] Stockholm University, Stockholm, Sweden

[2] Hokkaido University, Sapporo, Japan

Correspondence
Eriks Sneiders, Department of Computer and Systems Sciences (DSV), Stockholm University, Stockholm, Sweden
Email: eriks@dsv.su.se

Abstract

Automated answering of frequent email inquiries can be implemented as a text categorization task with narrow text categories, where all messages in 1 text category have the same answer. Such email categorization should be optimized for high precision and at least acceptable recall. One such high-precision email categorization method is matching of surface text patterns to incoming email messages. In order to assess the upper performance limits of text-pattern matching, we conducted extensive tests with almost 10,000 messages. Our results show that automated email answering with precision around 90% and recall 50–75% is feasible. In order to achieve this, however, the system must work with multiword expressions rather than stand-alone words. Furthermore, we argue that the system has to distinguish the context of an email inquiry from the actual need that created the inquiry—a question, request, or complaint. We have discovered and analysed 12 reasons why text-pattern matching may fail.

KEYWORDS

automated email answering, email categorization, question answering, text patterns

1 | INTRODUCTION

Over one third of the worldwide population is expected to be email users by 2019 (Email Statistics Report, 2015), despite the advances of social media, despite the fact that email is not popular in developing countries (e.g., in China exchange of personal electronic messages started with SMS for mobile phones [He, 2008]). The Radicati Group predicts an unbelievable figure of 128.8-billion business emails sent daily in 2019 (Email Statistics Report, 2015).

Many of business emails target contact centres of private and public organizations because email is a standard communication channel. And this channel does require resources. For example, answering a citizen's email sent to the Swedish Pension Agency takes about 10 min; the 99,000 messages per year require 9–10 full-time employments to answer them (Henkel, Perjons, & Sneiders, 2017). The cost is the prime reason why private contact centres often outsource their business to low-wage countries. A nonoutsourcing option for saving money, as well as saving people's time, is question- or email-answering systems that deliver instant answers. In countries such as Sweden, people do not mind "talking" to a computer as long as the answers are correct (Lewan, 2008). Australians seem to share a similar mindset (Kotadia, 2007).

This research originates in automated question answering in a closed domain, more precisely its least complex version—text-pattern-based frequently asked questions (FAQ) retrieval. Our first system matched manually crafted text patterns to user questions and retrieved relevant FAQs (Sneiders, 1999). Years later, this technique proved quite practical in commercial settings: The system could answer about 70% of the incoming queries with both precision and recall around 90% (Sneiders, 2009). A more advanced version of the system operated question templates that covered the conceptual model of a database (Sneiders, 2002). The system matched data items from the database to the user question; then applied text patterns, associated with the question templates, to find possible relationships between the matching data items in the question; and then ran database queries, associated with the question templates, to verify the relationships. On success, an answer template generated the answer.

The next challenge was to test how useful text-pattern matching could be for email answering. Iwai, Iida, Akiyoshi, and Komoda (2010) had observed that 30–40% of the email flow to a contact centre were relevant to previously published FAQs. Our own observations at several contact centres confirm the feasibility of such figures. Therefore, we started exploring automated email answering by FAQ retrieval, which is a text categorization task where each text category is flagged by an FAQ.

Our question-answering system was redeveloped into an email-answering system and tested in two domains, each in its own language (Sneiders, 2010). The system reached recall as high as 75%. Around 85% of the answered messages were answered correctly. Another 4% were answered almost correctly: The text patterns properly identified the request, but some additional information in the message implied that the author of the message could not use the standard solution proposed in the reply.

As the next step, we compared our text-pattern matching technique with machine-learning-based text categorization and the bag-of-words text representation (Dalianis, Sjöbergh, & Sneiders, 2011). Such comparison was particularly interesting because email answering requires narrow answer-oriented text categories whereas the traditional text categories are broad and topic oriented. As expected, the machine learning and the bag-of-words rendered higher recall whereas the text-pattern matching had higher precision. The size of the data set was, however, small, and the results therefore unreliable.

This article reports on a number of experiments with our text-pattern matching technique and almost 10,000 email messages. The messages arrived in four batches; therefore, we iteratively tested and improved the text patterns. Support vector machine (SVM) and the bag-of-words text representation were the baseline technique. We were looking for the answers to the following questions:

- What maximum levels of answer accuracy can we achieve by text-pattern matching? We assume that manually crafted text patterns deliver higher answer accuracy than that of any automatically generated text patterns. Together with the results from two other domains and two languages (Sneiders, 2010), this study outlines expected upper levels of answer accuracy by text-pattern matching.
- How does the answer accuracy changes when we iteratively test and then modify the text patterns using new data in the same text categories? Continuous maintenance of the text patterns would normally occur in a real-life email-answering system. Still, we are not aware of any study that has observed the changes in answer accuracy during the course of such maintenance.
- When text-pattern matching fails, what is the cause of the failure? Knowing the types and causes of the failures is important for development of the text patterns. In particular, such knowledge may be valuable for the developers of text-pattern extraction or matching algorithms. We are not aware of any published study of text-pattern matching errors in text retrieval and categorization.
- How much better, or worse, matching of carefully crafted text patterns performs compared with a bag-of-words machine learning baseline system? Such a comparison shows the gain from investment into development of the text patterns, as opposed to working with the bag-of-words document representation.

Answering these research questions makes a significant contribution to the research in automated email answering, which is a scarcely researched field of study (see Sneiders, 2016a for an overview of email answering techniques). Answering the second and third research questions makes a unique contribution to text retrieval and categorization by text-pattern matching.

We would like to emphasize that we work with manually crafted surface text patterns, and there are two reasons why we do that. First, our text patterns are built following a certain logical structure of email messages sent to contact centres, and it is difficult to reproduce that structure in the text patterns by using the well known statistical- and machine-learning text-processing techniques. Second, our work with text pattern generation has begun (Felzmann, 2015; Mametja, 2017), but is not yet finished.

A condensed version of this article has been presented at a conference (Sneiders, Sjöbergh, & Alfalahi, 2016). This version includes much more detail, explains the text patterns and how they match query text, and presents more context for interpreting the test results. The article is organized as follows. Section 2 introduces two vital elements of an email inquiry—context description and response trigger—which together determine the answer. Section 3 explains our text-pattern matching technique. Section 4 describes the settings of the experiments: the test data, the performance measures, and the test cases. Section 5 presents the test results for the text-pattern matching, which includes detailed performance figures, error analysis, and analysis of the query text that matches the text patterns. Section 6 presents the results of the tests specific for the baseline system. Section 7 compares the performance of the text-pattern matching and the baseline system. Section 8 summarizes performance figures of related systems and, thus, shows us the context of our own experiment results. Section 9 concludes the article.

2 | CONTEXT DESCRIPTION AND RESPONSE TRIGGER IN AN EMAIL INQUIRY

Text patterns for email answering mimic the language that people use in their email messages and follow a certain logical structure embodied in the messages. We have explored and explicated this structure, which became the "theoretical foundation" of why text-pattern matching in email answering works and can deliver good results.

Through observation of tens of thousands of email messages sent to contact centres, we have discovered that the initial inquiry almost always consists of two parts—(a) a description of the context of the inquiry and (b) a response trigger. The response trigger can be a question waiting to be answered, a request to carry out an activity and report the results, or a statement that requires a response (e.g., a complaint). For automated email answering, requests for information and requests to complete a task are of interest—the system can deliver information or redirect the user to a self-service application.

A simple question, explicit or implicit, is the easiest type of inquiry—both the context and the response trigger are placed in one sentence. Example,

> *I wonder whether I **get** any **payment** in April because I haven't received any notification which I usually get 25th every month.*

"Get" and "payment" are the keywords that define the subject of the inquiry. "Wonder" raises the question to be answered. The second "I" disambiguates the question. "In April" specifies a detail of the inquiry pinpointed by the subject. The entire question is a response trigger.

Sometimes, relevant keywords describe the context whereas the response trigger does not contain any domain keywords. Let us consider the example:

> *I sent you an **application** for **parental benefits** more than a week ago, how is my case doing?*

"Application" and "parental benefits" define the subject of the inquiry, "I sent" specifies a detail of the context. "How is my case doing" is the response trigger that does not contain any domain keywords and is meaningful only in a particular context.

Often, the context and the response trigger come in separate sentences, which make it more difficult to identify the inquiry. Let us consider the sentence:

> *I **applied** for **housing allowance** on 13 January 2016.*

It is a meaningful statement that introduces a story, yet alone, it does not ask for any response. The person continues the story by telling that she has received half of the amount she expected, and now, she would like to know:

> *When will the rest of it come?*

This sentence is the question to be answered; it is the response trigger and makes sense only together with its context. The context and the response trigger may come in separate sentences in a random sequence, possibly having other sentences in between. It may happen that the response trigger does not contain any domain keywords even in a separate sentence.

In Section 5.3, we verify the formats of the context descriptions and the response triggers in a sample of email messages; Table 10 confirms our findings.

A context description provides keywords that are useful for topic-related text clustering and categorization. In order to answer a message correctly, the system must take the next step and identify the request, the need, and why the message was sent in the first place. Paying attention to the nuances is crucial if the message contains several inquiries, which means several response triggers with one or several context descriptions. So far, we have found only one reference to a clear separation of context description and response trigger in automated email answering:

> *Determining the factual content of an e-mail is not sufficient to answer it correctly – its purpose is also very important. […] In the e-mails we are analyzing, the same set of data can be extracted from e-mails that have different purposes. The answer to be generated for these emails must therefore be different.(Kosseim, Beauregard, & Lapalme, 2001)*

The factual data here describe the context; the purpose is a response trigger.

Resolving the purpose of an email message is not a new research problem. There exists research that applies speech-act theory in order to categorize workplace email messages according to the purpose of the message, not the topic. Khosravi and Wilks (1999) analysed 1,000 email messages and developed a text-pattern matching system that tagged sentences in messages with 10 nuanced request labels, where requests for action, information, and permission were expressed directly or implicitly in statements and questions. Corston-Oliver, Ringger, Gamon, and Campbell (2004) had an ambition to create a system that would analyse an email message and add action items to the receiver's to-do-list. Sentences in a message were labelled "salutation," "social chit-chat," "task," "proposal to meet," "promise," "farewell," and various components of email signature. Best candidates for action items were "tasks," "proposals to meet," and "promises." All these requests and action items are, in fact, analogous to our response triggers.

Cohen, Carvalho, and Mitchell (2004) tagged the entire message, not individual sentences, according to the intent of the sender. The intent was identified with help of a small ontology of email acts, where the main actions were "request," "propose," "amend," "commit," "deliver," and the subjects of these actions were "information," "meeting," "data," and so forth. Goldstein and Sabin (2006) took a broader look at the email tasks and defined 12 email genres according to their task, including not only the familiar directives, commitments, and requests for information but also expression of feelings, document forwarding, and advertising and spam.

Lampert, Dale, and Paris (2008a) added details to the analysis of requests and commitments in workplace email. Requests and commitments may be conditional or unconditional, explicit or implicit. An example of implicit request is "Can you send me the curves and trades for January 18?" The request appears as a yes or no question, yet the receiver is expected to act upon it, not to answer it. Neither do we answer rhetorical questions or pleasantries—polite social utterances like "How are you?" Requests and commitments may be made on behalf of the writer or a third person. Clarifying these details proved crucial for improving agreement between human annotators who labelled utterances containing requests and commitments. Lampert, Dale, and Paris (2008b) pinpoint the difficulties in identifying requests and commitments, where the most prominent one is

locus ambiguity: Whereas human annotators tend to agree that the message contains a request or a commitment, they may not agree on exactly which utterances contain them.

The research on task-related email categorization in workplace email and the research in automated email answering at contact centres have at least one thing in common: Requests for information and requests to complete a task are task-related speech acts; they are also response triggers. The task-related speech acts and the response triggers may be explicit or implicit. They may be ambiguous and vague.

Our work in automated email answering bridges topic-related and task-related email categorization, two so far separate fields of study. Explicit definition of context description and response trigger, two logical components of an email inquiry sent to a contact centre, is a novel contribution of this research. We argue that selection of a good answer would be inherently faulty if we did not explicitly address both the context and the request. Our text patterns do identify the context and the request (see Section 3.2), unlike most research in automated email answering (see Sneiders, 2016a for an overview of email answering techniques).

3 | TEXT-PATTERN-BASED EMAIL ANSWERING

Our email answering system has been used in three configurations. First, the system can be set to answer Simple Mail Transfer Protocol messages. In such a case, the reply (a) informs the reader that it has been automatically generated, (b) interprets the inquiry by showing the matching FAQs, (c) answers the FAQs and consequently the inquiry, and finally (d) tells what to do if the reader is not satisfied with the answer. The second configuration generates draft replies for contact-centre agents who inspect and edit, if needed, the text before sending it. The third configuration is a web-form based messaging service. Being interactive and proactive, the system tries to answer the inquiry before the inquiry is sent to the contact centre.

3.1 | Question templates

The manually crafted text patterns and the process of matching them to query messages have been introduced in Sneiders (2010). This section summarizes the principles of the text-pattern matching technique.

The system operates a set of standard answers to the most frequent email inquiries. Each answer is linked to several question templates where each template contains two kinds of text patterns—required and forbidden text patterns, see Figure 1. The required text pattern embodies relevant wording of the future query messages to be answered by this standard answer. The forbidden text pattern handles exceptions; it states what wording must not be present in a query message even if the message matches the required pattern.

When a new query message arrives, the system matches each question template to the query, paragraph by paragraph (often, there is only one relevant paragraph). A paragraph sets the boundaries for a piece of text that describes one or several concepts, their attributes and relationships. Hence,

- For each question template,
  - The system matches the required text pattern to each query paragraph separately.
  - If at least one query paragraph matches the required text pattern, then the question template is accepted unless
    - The forbidden text pattern matches any paragraph of this query. The system searches for undesired wordings, encoded in the forbidden text pattern, in the entire query message in order to reduce the risk of incorrect answer.
- If the standard answer has at least one accepted question template then the standard answer is accepted.

As we see in the matching process, the scope of a standard answer is one query paragraph, which works well for the vast majority of email queries. One query message may eventually have several answers that complement each other.

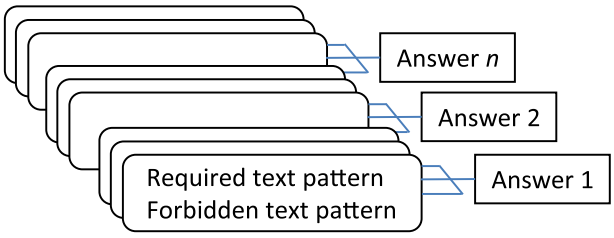

FIGURE 1 Answers and their question templates with two kinds of text patterns

3.2 | Context description and response trigger in a text pattern

The text patterns resemble regular expressions, but they are optimized for matching a piece of text; therefore, they are easier to handle than regular expressions. A text pattern must be able to capture both the context description and the response trigger in a query message. Following is a sample text pattern translated from Swedish.

[$me_we_etc :::; $need $acquire $apply $register :::::; $allowance $subsidy $child_support]
[$me_we_etc :::; change changed changing move moved moving ::::::; bank $bank_account];
[can could $want shall should :::; [you $social_service ::::::::; $send] [$me_we_etc ::::::::; $obtain] :::::; $fill_in_form]

The example shows two sets of phrases in one text pattern. First come two phrases in one set; each phrase identifies its own context. The context phrases are followed by a phrase (in the second set) that identifies the response trigger, which is common for both contexts. There may be several phrases for different contexts and different response triggers; each context must be compatible with each response trigger. For noncompatible contexts and response triggers, a new text pattern must be made. There is no defined order in which the context and the response trigger must follow each other in a matching query message.

In a query message, the request may be explicit or implicit, but it has to be clearly stated. A text pattern cannot match a vague story that requires human reasoning and background knowledge.

3.3 | Rule-based text categorization, Boolean retrieval

A rule-based text classifier operates a set of classification rules that are linked to a text category. The feature set of a document must comply with these rules if the document is to be assigned that text category (Aggarwal & Zhai, 2012). In our research, the text patterns are such rules, the standard answer is such a text category. Hence, our text-pattern matching system is a rule-based text classifier.

If we consider a phrase as one term, and a text pattern consists of sets of terms, and then the text patterns follow the standard Boolean retrieval model where the terms in one set are joined with the OR-operator, and the sets are joined with AND. Phrases introduce the proximity operator and term order; hence, we have the extended Boolean retrieval model where our text patterns resemble queries for the Westlaw legal retrieval system (Manning, Raghavan, & Schütze, 2009, section 1.4).

Rigorous extended-Boolean queries, which our text patterns are if we see incoming email messages as a document collection, are convenient for precise encoding of information needs. Furthermore, Boolean queries are language independent (at least with respect to the Indo-European languages) and domain independent. As a powerful lightweight tool, Boolean queries are viable in commercial settings. One example is Genesys, a vendor of solutions for contact centres, that offers so called "screening rules" – Java-implemented regular expressions – for routing of incoming email messages (according to information from technical sales support at the company).

3.4 | Summary of the features of the text patterns

The design of our text patterns observes the following core features:

- All text patterns capture the context description and the response trigger in a query message.
- Strict matching rules. The text patterns are Boolean queries. The matching alternatives are decided by the designer of the text pattern, not the system, and are encoded directly into the text pattern.
- Word order. A text pattern should often allow any order of significant words. The freedom of matching of idioms is more restricted.
- Distance between words. In a phrase, we may define any distance between words in a matching query sentence. We rarely require that matching words lie next to each other. Still, we rarely permit unlimited distance between the words in a phrase either. Words that appear far apart from each other are not likely to belong to the same phrase.
- Redundancy. Each text pattern expects wordings of future query texts. We guess those future wordings by observing the past wordings, yet the guess contains uncertainty. We compensate the uncertainty by limited redundancy of context-dependent synonyms that designate a concept, an attribute of a concept, a relationship between concepts, or disambiguate the context of the text pattern. A sufficient set of context-dependent synonyms is an essential property of well-working text patterns. In our experiments, we used a generic synonym dictionary and manually selected synonyms that subjectively felt applicable in the particular context, even if not very likely used. The synonym selection process was as intuitive as the text patterns themselves. About 70% of the generic synonyms from the dictionary usually fit the context. Furthermore, we used the best of our judgement and selected words and idioms that had paradigmatic relationships (i.e., context-dependent synonymy; Sahlgren, 2006, p. 60). If we had a large amount of text data, we could apply the techniques of distributional semantics (Sahlgren, 2006, pp. 37–45) to automatically identify paradigmatic relationships.
- The text patterns are subjective. The syntax of the text patterns is a tool in the hands of a craftsman. The result—a text pattern—depends on the skills of the craftsman. Such skills include domain knowledge, language ability, and the ability to determine expressions that are critical for individual text patterns.

4 | EVALUATION METHOD

The following subsections discuss the test data as well as the design of the test cases for the text-pattern matching and for the baseline system—SVM. The test results and analysis follow in Sections 5–7.

4.1 | Test data

Our test data are 9,663 email messages sent by citizens to handling officers of the Swedish Social Insurance Agency ("Försäkringskassan" in Swedish). The data were released by the agency to a small group of researchers during a research project aimed at increasing the efficiency and quality of email communication between citizens and public organizations (Knutsson, Pargman, Dalianis, Rosell, & Sneiders, 2010). Before the messages were made available for research, they were anonymized—stripped of sensitive information such as social security numbers, telephone numbers, names, email, and postal addresses. For the purpose of our experiments, we extracted the message body; if there was a dialogue thread, we extracted the chronologically first inquiry. Hence, our test messages were pieces of plain text with no dialogues or metadata. We identified five most common types of inquiries that could be answered by a standard answer and defined five text categories, see Table 1. Finally, we manually labelled each message according to its text category. Preprocessing of the messages is covered in Dalianis et al. (2011).

The email messages arrived over a period of time in four batches, called collections A, B, C, and D. A collection contained all messages that the agency had answered over a period of a few days. Table 2 shows the distribution of the messages across these collections and the text categories. The sum of all messages across categories in one collection is larger than the number of messages in the collection because some messages belong to two categories. In collection A, there are 6 messages that belong to two categories; in B, 9 messages; in C, 6 messages; and in D, 8 messages.

The email collections are not available in the public domain.

4.2 | Performance measures

Because we treated email answering with a standard answer as a text categorization task, we measured precision and recall in order to see how well the system placed a message into the right category Cat1 through Cat5 and Rest. Precision P is the share of messages that the system has correctly placed in a text category among all the messages that the system has placed in this category:

$$P = |\,number\ of\ placed\&relevant\,| \,/\, |\,number\ of\ all\ placed\,|\,.$$

Recall R is the share of messages that the system has correctly placed in a text category among all the messages that belong to this category:

$$R = |\,number\ of\ placed\&relevant\,| \,/\, |\,number\ of\ all\ relevant\,|\,.$$

F-score is a weighted average of precision and recall. We use

$$F = (2 \bullet P \bullet R)/(P + R).$$

TABLE 1 Text categories for automated answering

ID	Category title
Cat1	Please send me a fill-in-form.
Cat2	When will you decide my housing allowance?
Cat3	How many days of parental benefits do remain for my child?
Cat4	How much will I get in my future pension?
Cat5	When do I get my money?
Rest	All other messages

TABLE 2 Number of messages by collection and text category

	Number of messages in							
	Collection	Cat1	Cat2	Cat3	Cat4	Cat5	Rest	Sum
Collection A	2,437	94	76	51	29	362	1,831	2,443
Collection B	1,967	76	62	49	30	269	1,490	1,976
Collection C	2,473	109	105	53	22	387	1,803	2,479
Collection D	2,786	148	79	45	78	393	2,051	2,794
Total	9,663	427	322	198	159	1,411	7,175	9,692

Low *F*-score means low accuracy of categorization; high *F*-score means that either both precision and recall are high or that one of them is very high.

Precision and recall can be measured per category or in the entire collection across all the categories at once. If categorization is done in the entire collection and each message is assigned exactly one category, then precision is equal to recall, and they both are equal to *F*-score. In such a case, we call them categorization accuracy. Accuracy is the proportion of correctly classified documents.

For our statistical significance tests, we used McNemar's test (Everitt, 1977). We tested whether two systems made the same decisions by comparing the number of items *N*, classified correctly by System 1 but classified incorrectly by System 2, to the number of items *M*, classified incorrectly by System 1 but classified correctly by System 2. The test statistic is calculated as $(N - M)^2 / (N + M)$. Under the null hypothesis that there is no difference in performance between the systems, this has a chi-squared distribution with 1 degree of freedom if the numbers *N* and *M* are large. If the chi-square result is significant, then the null hypothesis is rejected, and we say that the difference in the classification results is significant.

4.3 | Test cases for text-pattern matching

Initially, we had only collection A for "training" and collection B for testing. We write "training" in quotes because it was not training as in machine learning. Because our text patterns are manually crafted, training means our own observation of email messages, our own learning of how people express themselves, designing initial text patterns, adjusting them to new "training" messages, and so on until all the "training" messages have been processed. During the "training", our prime concern was answer precision—if the message is answered, it must be answered correctly. We worked on improving recall as well, while keeping precision high.

Over some period of time, collections C and D arrived. The new collections made it possible to measure stepwise improvement of the text patterns: We could add the test collection to the "training" data, "retrain" the text patterns, and take a new test collection. Stepwise improvement of the text patterns happens during the maintenance of a real-life system: The maintainers observe how well the system answers messages from the email flow, analyse the errors, update the text patterns to deal with these errors, make some new errors, and observe the system's performance with the next email flow.

Different test collections, however, do not allow seeing the true effect of stepwise improvement. Therefore, we used the final collection D to test text patterns "trained" on the collections A + B and A + B + C. Unfortunately, we could not use D in order to test patterns trained solely on A because by the time D arrived, the old A-trained text patterns were already lost. Table 3 summarizes the "training" and test data sets for each test case. The same tests were also included in the performance evaluation of the baseline system—SVM.

Apart from being anonymized, the message text in the input of the text-pattern matching system was not altered. The system had spelling correction built into the pattern matching process (Sneiders, 2010). The text patterns contained word stems; therefore, stemming or lemmatization of the input text was not necessary. The text patterns were designed to recognize compound words; therefore, compound splitting during the text preprocessing was not necessary.

4.4 | Test cases for SVM only

In order to compare the text-pattern matching system with the standard machine learning methods, we did similar experiments on the same data using the Waikato environment for knowledge analysis (WEKA) framework (Hall et al., 2009). We used the SVM learning method implemented in WEKA. The only features that represented the text were individual terms.

Figure 2 illustrates the principle of SVM by an idealized two-class categorization case (Manning et al., 2009, section 15). Each document is a vector in a vector space (circles and triangles in the figure are the endpoints of the vectors in a two-dimensional space), and there is a void between two sets of training documents. Support vectors are the training documents that define the borders of the void (the dotted lines in the figure) so that the void is as broad as possible. During the SVM training process, a decision hyperplane (in case of two dimensions—a line) is constructed in the middle of the void so that the margins between the hyperplane and the support vectors are maximally large. After the training is completed, the decision hyperplane assigns one of the two text categories to the test documents. Because SVM relies on support vectors, it works well also with small sets of training data as long as the support vectors are representative for the categorization task.

An alternative to SVM could be Naïve Bayes, which is also implemented in WEKA. In a previously conducted test A-B for SVM and Naïve Bayes (Dalianis et al., 2011), both techniques had similar performance whereas SVM had a slightly better precision; we prioritize precision. In some

TABLE 3 Evaluation cases with fixed "training" and test data sets for the text-pattern matching and support vector machine (SVM)

Test name	"Training" collections	Test collection
A-B	A	B
AB-C	A + B	C
ABC-D	A + B + C	D
AB-D	A + B	D
SVM only: A-D	A	D

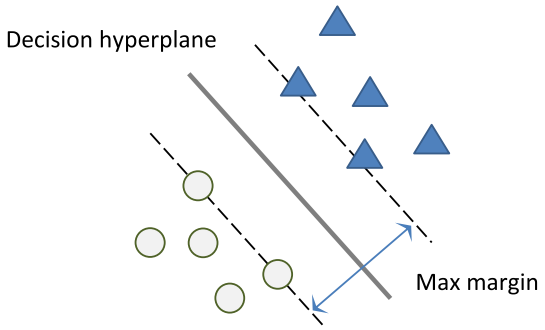

FIGURE 2 Building the support vector machine decision hyperplane with two sets of training data

older experiments with answer-specific email categorization (Busemann, Schmeier, & Arens, 2000; Scheffer, 2004), SVM outperformed Naïve Bayes. SVM has been used for answer-specific email categorization also by Bickel and Scheffer (2004), as well as Itakura, Kenmotsu, Oka, and Akiyoshi (2010).

Aggarwal and Zhai (2012) list over 10 text categorization methods, where popular ones are SVM, Bayesian classifiers, neural networks, rule-based classifiers (our text-pattern matching is one of them), decision trees, and *k*-NN. Although SVM appears to be the most popular choice, some studies (Colas & Brazdil, 2006; Youn & McLeod, 2007) show that SVM not always outperforms the other techniques.

Neural networks have been used in a variety of categorization tasks, such as image categorization based on the shape, colour, and texture of the depicted objects (Zhang et al., 2016). Some neural networks versus SVM studies show that both techniques may perform equally well (e.g., in text categorization [Zaghloul, Lee, & Trimi, 2009]), or SVM may do slightly better (e.g., in credit rating analysis [Huang, Chen, Hsu, Chen, & Wu, 2004] and drug or nondrug categorization by molecular structures [Byvatov, Fechner, Sadowski, & Schneider, 2003]).

We carried out two rounds of tests with SVM. One round was benchmarking the text-pattern matching system with the tests in Table 3. The other round was experiments with SVM itself in order to see how its performance depends on the amount of and the preprocessing of the input data. The latter test cases are shown in Table 4. These tests used 10-fold cross validation.

We evaluated the impact of three text-preprocessing methods applied to the SVM input: spelling correction, lemmatization, and compound splitting.

The first method was spelling error correction. We used the Stava (Domeij, 1994) spelling correction programme for Swedish.

The second method was lemmatization. In text categorization, lemmatization usually improves recall. We used the Granska part-of-speech tagger for Swedish (Carlberger & Kann, 1999) that also produces lemma information.

The third method was compound splitting. Swedish has very productive compounding of words; therefore, splitting compound words into their components can be useful in many text processing applications. In text categorization, compound splitting usually improves recall. We used a tool (Sjöbergh & Kann, 2004) based on Stava to split compound words.

In the experiments with SVM, we applied the three text preprocessing methods alone as well as combined. We started with spelling correction because email messages tend to be heavily misspelled, which makes both lemmatization and compound splitting less effective. Then we applied lemmatization and compound splitting, changing the order in which both methods were applied.

WEKA SVM cannot place a message into more than one text category, whereas 29 messages in Table 2 belong to two categories and cannot be treated by SVM properly. There is no perfect solution to this. The solution we chose was duplicating these 29 messages, each copy labelled with a different category, so that the number of messages in SVM input was equal to the number of message-category pairs, and SVM had an opportunity to put every message into the right category.

We used the LibSVM default settings, which means that SVM used a linear kernel function, normalization of the input data, a cost parameter of 1.0, and a termination criterion of 0.001. The documents were converted to SVM training and test data using the WEKA tool

TABLE 4 Evaluation cases with 10-fold cross validation for support vector machine

Text preprocessing	Data collections
None	A + B
None	A + B + C
None	A + B + C + D
Spelling	A + B + C + D
Lemma	A + B + C + D
Compound	A + B + C + D
Spelling → Compound	A + B + C + D
Spelling → Lemma↔compound	A + B + C + D

"StringToWordVector", which takes one message as a string and builds a vector where each dimension is a word, and the value is 0 if this word does not occur in the document, or a nonzero value if the word occurs (higher value the more times the word occurs). We used vectors of dimension 10,000, which seemed to be a good compromise. Smaller vectors gave worse categorization accuracy, and longer vectors made the computer run out of memory.

5 | EVALUATION RESULTS FOR TEXT-PATTERN MATCHING

This section presents the performance figures from the four text-pattern matching tests in Table 3. The results are followed by analysis of the errors of the text-pattern matching, and analysis of the query text that correctly matched the text patterns.

5.1 | Precision, recall, *F*-score

Appendix A shows the message categorization details for the tests defined in Table 3. Table 5 summarizes precision, recall, and *F*-score from the tests A-B, AB-C, and ABC-D. The performance figures are grouped by consecutive tests, which make it easier to follow how the figures change with each new test. Figure 5 in Section 7.1 shows the precision and recall trends for Cat1 through Cat5 between the three tests.

The total precision in Table 5 was calculated as the number of correctly placed messages divided by the sum of correctly and incorrectly placed messages across all the categories in the test collection. The total recall was calculated as the number of correctly placed messages divided by the number of message-category pairs across all the categories in the test collection. The number of messages is slightly lower than the number of message-category pairs because a few messages belong to two categories.

As expected, precision and recall between A-B and AB-C mostly improve. The test ABC-D, however, somewhat surprisingly shows a decrease in both precision and recall, despite the text patterns being presumably improved. The total drop with respect to the entire test collection is not big —0.01 precision and 0.02 recall. For Cat4, however, precision dropped by 0.11; for Cat3 recall dropped by 0.09.

One possible explanation of the performance drop could be overfitting (overtraining) of the text patterns. An increased number of training messages leads to an increased number of text patterns and an increased level of details in each pattern, which makes the set of text patterns more complex, which may lead to a performance drop.

Another explanation could be the use of a different test collection for each test. The performance drop may be caused by some test data sets being harder or easier to categorize than other test data sets.

The tests AB-D and ABC-D use the same test collection D and demonstrate a clearer effect of stepwise improvement between the training collections A + B and A + B + C. Table 6 shows the performance figures. In Figure 6 in Section 7.2, we can observe the precision and recall trends for Cat1 through Cat5 between the two tests. The performance trends between the tests AB-D and ABC-D show overall performance improvement along with more "training" data. Still, precision drops for Cat2 and Cat4 whereas their recall rises; recall drops for Cat3 whereas the precision rises. These three text categories have one thing in common—They are small.

Table 7 shows the details of the tests AB-D and ABC-D. In Cat2, Cat3, and Cat4, the number of incorrectly placed messages is 3 to 5, and the precision difference is caused by only one message. The recall change is caused by a difference of 2 to 3 correctly placed messages. In Cat5, for comparison, precision increases when the number of incorrectly placed messages decreases by 11 messages from 42 to 31. Because we have so little test and "training" data and the changes are small, we cannot make any firm conclusions about the system's performance change in Cat2, Cat3, and Cat4.

For the entire test collection D, precision increase between the tests AB-D and ABC-D was 0.02; recall increases was 0.01. It is a small change. To see if the difference was statistically significant, we calculated McNemar's test for the results of AB-D and ABC-D. The performance increase was statistically significant on the 5% level.

TABLE 5 Precision, recall, and *F*-score from the tests A-B, AB-C, and ABC-D

	Precision			Recall			*F*-score		
Test	A-B	AB-C	ABC-D	A-B	AB-C	ABC-D	A-B	AB-C	ABC-D
Cat1	0.91	0.92	0.93	0.54	0.63	0.59	0.68	0.75	0.72
Cat2	0.97	1.00	0.91	0.53	0.56	0.65	0.69	0.72	0.76
Cat3	0.96	0.92	0.92	0.55	0.85	0.76	0.70	0.88	0.83
Cat4	0.88	1.00	0.89	0.50	0.45	0.44	0.63	0.63	0.59
Cat5	0.84	0.91	0.88	0.41	0.63	0.59	0.55	0.75	0.71
Rest	0.86	0.88	0.87	0.99	0.99	0.98	0.92	0.93	0.92
Total	0.86	0.89	0.88	0.86	0.89	0.87	0.86	0.89	0.88

TABLE 6 Precision, recall, and *F*-score from the tests AB-D and ABC-D

	Precision		Recall		F-score	
Test	AB-D	ABC-D	AB-D	ABC-D	AB-D	ABC-D
Cat1	0.89	0.93	0.55	0.59	0.68	0.72
Cat2	0.92	0.91	0.62	0.65	0.74	0.76
Cat3	0.90	0.92	0.82	0.76	0.86	0.83
Cat4	0.91	0.89	0.40	0.44	0.55	0.59
Cat5	0.84	0.88	0.55	0.59	0.67	0.71
Rest	0.86	0.87	0.97	0.98	0.91	0.92
Total	0.86	0.88	0.86	0.87	0.86	0.88

TABLE 7 Details of the tests AB-D and ABC-D

	Num relevant msg			Num correctly placed msg		Num incorrectly placed msg	
	Test msg	"Training" msg					
	D	A + B	A + B + C	AB-D	ABC-D	AB-D	ABC-D
Cat1	148	170	279	81	87	10	7
Cat2	79	138	243	49	51	4	5
Cat3	45	100	153	37	34	4	3
Cat4	78	59	81	31	34	3	4
Cat5	393	631	1,018	217	233	42	31

Note. Msg = messages

The text category Rest has high recall: with 97–99% success rate it identified the messages that should not be answered. Burke et al. (1997) have coined the term rejection – the share of nil-answers when there are no relevant answers in the database. That is, Burke's rejection and our Rest show the ability of the system not to deliver garbage if there is no relevant information to deliver.

5.2 | Error analysis

We manually inspected the reasons of incorrect category placements after the test AB-C. Collection C had 2,479 messages, of them, 676 messages belonged to Cat1 through Cat5. Four hundred twenty-eight messages in Cat1 through Cat5 where correctly placed; the remaining 248 were incorrectly placed into Rest. Thirty-four messages were incorrectly placed into one of the categories Cat1 through Cat5.

Missing answer, an unanswered message, means that the message was incorrectly placed into Rest. We established 9 error classes for that. Table 8 summarizes the distribution of unanswered messages per error class and text category.

5.2.1 | Err1: Unique wording

The wording in these unanswered messages does not resemble any text pattern in the system's database. There is little we can do to answer these messages. If the number of training messages rises and some of the unique messages start exhibiting common wordings, then we can create new text patterns to cover these wordings.

5.2.2 | Err2: Similar text pattern available

This error class is similar to Err1, except that there are one or more text patterns in the system's database that are relevant to the unanswered message. These text patterns include concepts that appear in the message, but wording of the details around these concepts is not covered by

TABLE 8 Distribution of unanswered messages per error class and text category

	Err1	Err2	Err3	Err4	Err5	Err6	Err7	Err8	Err9	Total
Cat1	19	7	7	1	2	1	1	2		40
Cat2	20	18	3	1			1	3		46
Cat3		1	5	1			1			8
Cat4	6	2	1			1	1		1	12
Cat5	41	23	27	4	9	18	10	5	5	142
Total	86	51	43	7	11	20	14	10	6	248
Share %	34.7	20.6	17.3	2.8	4.5	8.1	5.6	4.0	2.4	100

the text patterns. It is unclear how much we would benefit from modifying an existing text pattern in order to answer this particular message. If, however, the number of training messages rises, some of the unanswered messages in this error class may become useful for improving existing text patterns.

5.2.3 | Err3: Missing synonym or word order

There exists an almost matching text pattern in the system's database, but this text pattern did not match the message because of one concept. In the text pattern, the concept was represented by a number of synonyms, whereas the message used different one. A synonym may be a word or an expression; for example, "understand" and "make out" are synonyms. In some cases, the unanswered message used a generalization of a particular concept. If we want a text pattern to include more general synonyms, we need to redesign the text pattern and include expressions that specify the context in which these more general synonyms are used. Not specifying the context will lower precision.

In a very few cases, the text pattern did not match because it stipulated the opposite sequence of two matching words.

It should not be difficult to automatically discover new synonyms for concepts in existing text patterns. As the number of training messages rises, this task should become easier.

5.2.4 | Err4: Missing disambiguation in the query message

This error class is similar to Err3. There exists a nearly matching text pattern in the system's database. Much of the text pattern, usually a general key phrase, did match the message. Nevertheless, the text pattern also contained a number of disambiguating expressions that specified the context of the general key phrase, and none of these expressions matched the query message. For example, the statement "I'd like to have my money" may imply different things in different contexts; therefore, the text pattern requires that a disambiguating expression such as "for March–April" also matches. Then we can assume that the writer asks for a payment of social benefit for the given period, because the message is sent to a government agency whose main business is social benefits.

5.2.5 | Err5: Missing subject or action word in the message

This error class is similar to Err4. There exists a nearly matching text pattern in the system's database. Still, the nearly matching piece of text in the message did not contain the subject or action word required by the text pattern for a match. For example, the unanswered message says "I don't have my allowance", whereas the nearly matching text pattern requires "I don't have my allowance paid". Taking the subject or action word out of the text pattern would require adding disambiguating context expressions.

5.2.6 | Err6: Too nuanced message

This error class is similar to Err2. The message is rich in specific details. Nonetheless, this wealth of details does not match any recurring text pattern.

5.2.7 | Err7: Vague message

The message can be understood by a human who knows the overall context and can reason, but the information need or the purpose of the message is not clearly expressed; the response trigger is not clear or is missing.

5.2.8 | Err8: Untreated misspelling

Although the system does correct spelling mistakes, some misspellings may be left untreated, and relevant text patterns may fail to match because of one word. Some misspellings may be correctly spelled different words, such as "maid" instead of "made." Nonstandard abbreviations can be considered misspellings here.

There is little we can do with text patterns in order to treat spelling mistakes missed by a spellchecker. We can treat nonstandard abbreviations as missing synonyms in Err3.

5.2.9 | Err9: Ill-structured message

The message lacks proper sentences, or a sentence resembles a collection of words.

Err1 and Err2 account for 55.3% of all unanswered messages, as we see it in Table 8; in order to cover these messages by the text patterns, we need to add more text patterns and substantially redesign existing ones. Err3, Err4, and Err5 together account for 24.6% of all unanswered messages; these messages almost matched existing text patterns in the system's database. A complete match failed because details in the text pattern disagreed with the message text. Fixing these details requires minor changes in the existing text patterns.

Wrong answer means that the message was incorrectly placed into one of the categories Cat1, Cat3, or Cat5; Cat2 and Cat4 did not have any wrong answers. We established three error classes; Table 9 summarizes the distribution of incorrectly answered messages per error class and text category.

TABLE 9 Distribution of incorrectly answered messages per error class and text category

	Err10	Err11	Err12	Total
Cat1	4		2	6
Cat3		4		4
Cat5	12	1	11	24
Total	16	5	13	34
Share %	47.1	14.7	38.2	100

5.2.10 | Err10: Related context

The system has found a matching piece of text in the query message, but the meaning of that piece of text is different from what the text pattern expected because of different overall context of the message, which invalidates the standard answer.

5.2.11 | Err11: Context changes in a subordinate clause

The system has found a matching sentence in the query message, but a subordinate clause in the sentence makes the context of the sentence different from what the text pattern expected, which invalidates the standard answer.

5.2.12 | Err12: Meaning changes because of a word

The system has found a matching sentence in the query message, but there is a word or a phrase in the sentence that invalidates the expected meaning of the sentence and, therefore, invalidates the standard answer.

Section 3.1 introduced required and forbidden text patterns; the forbidden text patterns are supposed to deal with recurring expressions that invalidate the expected meaning of the text that matches the required text patterns.

5.3 | Context descriptions and response triggers in the test data

Section 2 defined three formats of context description and response trigger in an email inquiry. We examined 300 random messages in Cat1 through Cat5 in collection C in order to verify presence of these formats in our test data. Four messages belonged to two text categories; therefore, we had 304 message-category pairs.

Table 10 shows the number of message-category pairs per arrangement of the context description and the response trigger. Half of the inquiries is a simple question. Although a few messages do contain only one sentence; in most cases, there is an additional description of the context. The essence of the inquiry, sufficient for selecting a standard answer, is nevertheless summarized in one question.

In the cases of "vaguely written messages", the information need in the message can be understood by human readers who can interpret the text. "Understanding" of such an inquiry by a computer is likely to be difficult, which has resulted in the error class Err7 in Section 5.2.

Table 10 illustrates the phenomenon that makes email answering by text-pattern matching possible. The text of email messages to contact centres follows a certain logical structure, and the number of format options of this structure is limited. The text patterns cover these options. Furthermore, the essence of an inquiry is often embodied in one or two sentences.

5.4 | Traits of the text-pattern matching

When we introduced our text-pattern matching technique in Section 3, we could not say much about the matching pieces of the query text. Once we have completed the experiments and have 154 required text patterns, we can describe (a) the amount of the query text that the text patterns actually matched, (b) the distance between matching query words, and (c) their vocabulary. Such knowledge complements the error analysis in Section 5.2: We can use this knowledge to improve the text patterns, to fine-tune the algorithms for text-pattern extraction, and to improve the algorithms for matching of automatically extracted text patterns to email text.

After we had completed the test ABC-D, we ran the collections A + B + C + D through the email answering system once again. One thousand nine hundred nine messages were correctly placed into categories Cat1 through Cat5. Because one message could belong to two categories, the

TABLE 10 Arrangements of the context description and the response trigger and the number of corresponding message-category pairs

Arrangement of the context description and the response trigger		Num	Share %
Simple question	Keyword-rich inquiry	159	52
	Response trigger without domain keywords	8	3
	Vaguely written message	3	1
Context description and response trigger in separate sentences	Keyword-rich inquiry	87	28
	Response trigger without domain keywords	33	11
	Vaguely written message	14	5
Total		304	100

system made 1,918 correct categorization instances. We selected these categorization instances for a closer look at the traits of the text-pattern matching. We did not consider any incorrect categorization instances.

5.4.1 | Number of matching sentences

In 1,577 categorization instances, a text pattern matched only one sentence in the query message. In 327 cases, there were 2 sentences, and in 14 cases—3 sentences. Thus, 82% of the categorization instances had one matching sentence, whereas in Table 10, only 56% of all inquiries had the context description and the response trigger in one sentence. Apparently, the system was biased toward context description and response trigger in one sentence because such an inquiry was easier to detect than an inquiry scattered across two or three sentences.

5.4.2 | Number of matching query words

Figure 3 shows the number of matching words in a query message across the 1,918 correct categorization instances. Nineteen messages were correctly categorized because of only three matching words; these must have been very significant three words. Five to seven matching words are clearly most common—They occurred in 1,338 or about 70% of the correct categorization instances.

Figure 4 shows the share of matching words among all the words in a query message across the 1,918 correct categorization instances. In a very few cases, the entire message matched a text pattern. An example of such rare case is "When will you pay my money!!!" No "hello", no "good bye". In half of the categorization instances, less than 18% of the message text matched a text pattern.

5.4.3 | Gaps between matching query words

A gap between two matching words in a query message is the number of nonmatching words between them. Of the total 9,835 gaps, 6,092 gaps (61.9%) were of Size 0, that is, the matching words were next to each other. Two thousand one hundred fifteen gaps (21.9%) were of Size 1, and 728 gaps (7.4%) were of Size 2. In 91.2% of the cases, the matching words had no more than 2 nonmatching words between them.

An *n*-gram is a sequence of *n* terms, mostly language words, without any gaps between them. The second column in Table 11 shows the number of extracted distinct *n*-grams made of lemmatized matching query words (an *n*-gram that appeared across the texts several times was counted as one). We decomposed larger *n*-grams into smaller ones that coexisted in the matching text across the 1,918 categorisation instances and were also accounted for in the second column of Table 11. The decomposition did not permit full or partial overlap of the component *n*-grams. The third and fourth columns show how many larger *n*-grams could be decomposed into coexisting bigrams or trigrams and 2- to 7-grams. Initially, we hoped

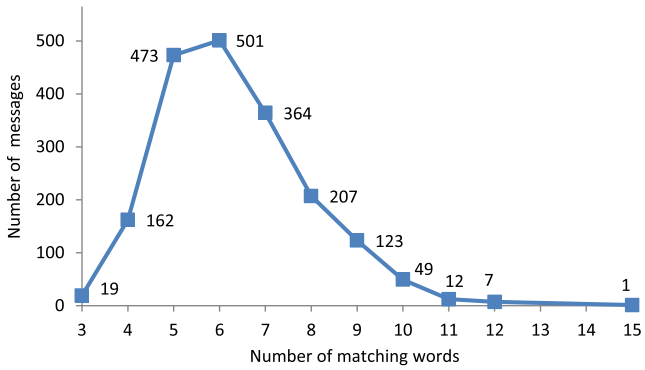

FIGURE 3 Number of matching query words and their frequency

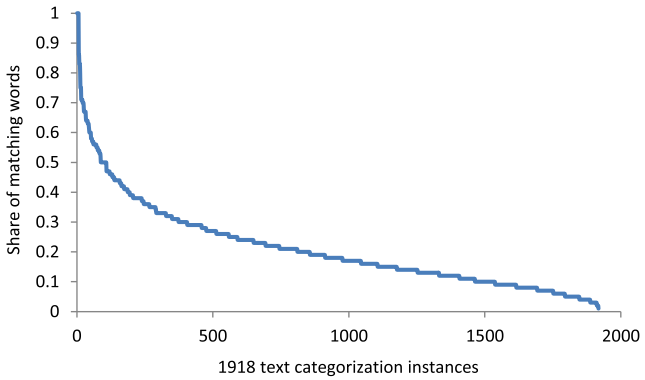

FIGURE 4 Share of matching words in a query message for each text categorization instance

TABLE 11 *N*-grams made of matching query words

Size	Num	Decomposable into 2- to 3-grams	Decomposable into 2- to 7-grams	
			Num	Share %
1-gram	428			
2-gram	544			
3-gram	451			
4-gram	282	62	62	22
5-gram	157	61	61	38.9
6-gram	74	7	15	20.3
7-gram	23	7	8	34.8
8-gram	8	1	5	62.5
9-gram	3	0	0	0
Total	1,970	138	151	

that the larger *n*-grams would easily split into smaller ones, and we could eventually obtain a compact set of small *n*-grams that cover most of the matching text. The third, fourth, and fifth columns show results different from what we were hoping for.

We have observed that an *n*-gram (a distinct *n*-gram counted in the second column) appears on average twice in the correctly categorized message text, which means that individual *n*-grams are not statistically representative. This explains the difficulty to decompose larger *n*-grams into smaller ones—The smaller ones are not statistically representative. Also, this explains the difficulty of automatic text-pattern extraction. Because our manually crafted text patterns represent each concept by a number of synonyms, we argue that concept patterns that cover semantically related words are likely to be more representative than *n*-grams. After all, we had 154 text patterns that placed 9,663 email messages into five text categories and "the rest."

5.4.4 | Vocabulary of the matching query words

We explored the most frequent matching words in the query messages. Table 12 shows the lemma of each word in Swedish and its translation into English. The words in italic tell what the email was about, these words are domain specific. Surprisingly, the seven most frequent matching words are ordinary words of the daily language; these words are not domain specific. A text pattern contains many common language words that disambiguate the text pattern.

We explored also part-of-speech patterns that could be text-category specific but did not find any such patterns (Sneiders, Eriksson, & Alfalahi, 2014).

6 | EVALUATION RESULTS FOR SVM ONLY

Before we compare the results of the text-pattern matching with those of SVM, we explore how the amount of training data and text preprocessing methods influence text categorization accuracy—the share of correctly categorized messages—for SVM.

Table 13 shows the results of 10-fold cross validation for the tests defined in Table 4. In order to see the effect of the amount of training data on SVM, we used the data sets comprising all text categories in collections A + B, A + B + C, and A + B + C + D. No text preprocessing was applied. The first three rows in Table 13 show the test results. The performance drops when we add more data (i.e., collection D) although we would expect more data to lead to better performance. The text pattern matching tests also showed a similar trend—ABC-D had lower precision than AB-C. This suggests that the messages in collection D are more difficult to categorize than the messages in the other collections.

Next, we evaluated the effect of different text preprocessing methods on collections A + B + C + D. Table 13, the third row and down, shows the accuracy and the number of correctly classified messages using no preprocessing, then using spelling correction, lemmatization, and compound splitting alone and combined. If several methods were applied, they were applied in the order shown in Table 13.

Text preprocessing improved the accuracy by 0.009. Although it does not seem much, the difference is 83 messages, which is more that Cat3 or Cat4 have in any collection A, B, C, or D. The highest accuracy was achieved with compound splitting only.

7 | TEXT-PATTERN MATCHING VS. SVM

We compare our text-pattern matching system with SVM by comparing their precision and recall values in the same tests. We observe (a) which system has higher values and (b) the trends of these values across consecutive tests. The values for the text-pattern matching system come from Tables 5 and 6. The values for SVM have not been presented previously. The input of SVM was preprocessed by spelling correction and compound splitting in order to meet the analogous features built into the text-pattern matching system.

TABLE 12 Most frequent lemmas of matching query words

Lemma	English	Num
Jag	I	841
Få	Get	724
När	When	467
Inte	Not	448
Undra	Wonder	386
Ha	Have	371
Hur	How	334
Skicka	*Send*	294
Blankett	*Form*	293
Peng	*Money*	274
Min	My	258
Bostads-bidrag	*Housing allowance*	246
Komma	Come	244
Utbetalning	*Payment*	226
Ni	You	223
För	For	198
ersättning	*Compensation*	159
Kunna	Can	157
Många	Many	149
Vilja	Want	147
Dag	*Day*	142
Vara	Be	137
På	On	126
Ansökan	*Application*	122
Det	This/that	111
Ta	Take	111
Föräldra-penning	*Parental allowance*	157
Ut	Out	94
Pension	*Pension*	88
Betala	*Pay*	82

TABLE 13 Categorization accuracy with 10-fold cross validation for support vector machine

Text preprocessing	Data set	Accuracy	Num correctly classified message
None	A + B	0.857	3,786 of 4,419
None	A + B + C	0.863	5,954 of 6,892
None	A + B + C + D	0.860	8,335 of 9,692
Spelling	A + B + C + D	0.866	8,395 of 9,692
Lemma	A + B + C + D	0.861	8,342 of 9,692
Compound	A + B + C + D	0.869	8,418 of 9,692
Spelling → Compound	A + B + C + D	0.868	8,409 of 9,692
Spelling → Lemma → Compound	A + B + C + D	0.861	8,349 of 9,692
Spelling → Compound → Lemma	A + B + C + D	0.861	8,344 of 9,692

7.1 | A-B, AB-C, and ABC-D

Table 14 puts side by side the precision and recall values for the text-pattern matching system and SVM from the tests A-B, AB-C, and ABC-D. "Total" for SVM is weighted average where WEKA calculated the average value across all the text categories considering the number of messages in each category. Figure 5 illustrates the precision and recall trends for both systems across the tests.

Bold typeface in Table 14 highlights the text-pattern matching system-SVM value pairs, where text-pattern matching system is superior; italic typeface distinguishes the advantage of SVM. The comparison shows that the text-pattern matching has higher precision whereas SVM tends to have higher recall in Cat1 through Cat5; Rest naturally shows the opposite trend.

TABLE 14 Precision and recall from the tests A-B, AB-C, and ABC-D

	Precision						Recall					
	A-B		AB-C		ABC-D		A-B		AB-C		ABC-D	
Test	PM	SVM	PM	SVM	PM	SVM	PM	SVM	PM	SVM	PM	SVM
Cat1	**0.91**	**0.60**	**0.92**	**0.69**	**0.93**	**0.69**	**0.54**	**0.49**	**0.63**	**0.54**	*0.59*	*0.69*
Cat2	**0.97**	**0.73**	**1.00**	**0.85**	**0.91**	**0.69**	*0.53*	*0.60*	*0.56*	*0.77*	*0.65*	*0.86*
Cat3	**0.96**	**0.89**	**0.92**	**0.82**	**0.92**	**0.73**	*0.55*	*0.82*	0.85	0.85	*0.76*	*0.89*
Cat4	**0.88**	**0.77**	**1.00**	**0.85**	**0.89**	**0.70**	**0.50**	**0.43**	*0.45*	*0.50*	*0.44*	*0.58*
Cat5	**0.84**	**0.67**	**0.91**	**0.73**	**0.88**	**0.63**	*0.41*	*0.65*	*0.63*	*0.71*	*0.59*	*0.80*
Rest	*0.86*	*0.89*	*0.88*	*0.90*	*0.87*	*0.93*	**0.99**	**0.92**	**0.99**	**0.92**	**0.98**	**0.87**
Total	**0.86**	**0.84**	**0.89**	**0.86**	**0.88**	**0.85**	**0.86**	**0.85**	**0.89**	**0.86**	**0.87**	**0.84**

Note. SVM = support vector machine; PM = text-pattern matching system.

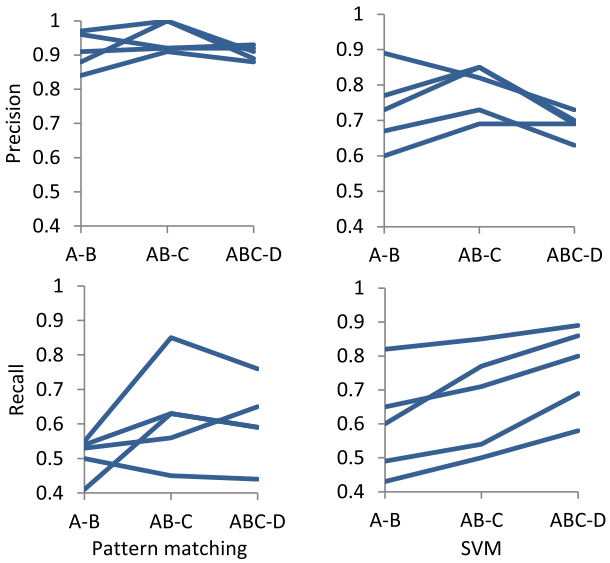

FIGURE 5 Precision and recall trends for Cat1 through Cat5 between the tests A-B, AB-C, ABC-D. SVM = support vector machine

The precision trends across the three tests are similar for both systems. For Cat2, Cat4, and Cat5, the precision values make a hump at AB-C. For Cat1, the trend is a weak precision increase. For Cat3, the trend is a weak precision drop for the text-pattern matching and a strong drop for SVM. Because both techniques demonstrate the same trend, apparently, the collection D is more difficult to categorize than the collection C.

The recall values behave differently. For SVM, adding more training data keeps the recall rising. For the text-pattern matching, the recall and precision trends are vaguely similar. For Cat1, Cat3, and Cat5, the recall values make a hump at AB-C. For Cat2, the trend is a recall increase, and for Cat4—a drop.

7.2 | AB-D and ABC-D

The tests AB-D and ABC-D use the same test collection D and therefore demonstrate a clearer effect of more training data. Table 15 puts side by side the precision and recall values from the test AB-D for the text-pattern matching and SVM. Also here bold typeface highlights text-pattern matching system-SVM value pairs where text-pattern matching system is superior; italic typeface distinguishes the advantage of SVM. The text-pattern matching demonstrates superior precision, while neither of the both systems have any considerable recall advantage.

The results of the test A-D, also shown in Table 15, are discussed in Section 7.4 because the test is not defined for the text-pattern matching.

Figure 6 shows the precision and recall trends between the tests AB-D and ABC-D. For SVM, more training data leads to more aggressive categorization into Cat1 through Cat5 and increased recall. Because the amount of the training data is still small, more aggressive categorization lowers precision. The text-pattern matching has a statistically significant overall performance improvement (see Section 5.1): whereas precision slightly drops for Cat2 and Cat4, it has a stronger increase for Cat1, Cat3, and Cat5; recall rises in 4 of the 5 text categories—Cat1, Cat2, Cat4, and Cat5.

The text-pattern matching and SVM have different precision and recall trends. Updated text patterns more often lead to precision increase rather than precision drop. The reason, we believe, is matching of meaningful multiword expressions that capture the context of the inquiry and

TABLE 15 Precision and recall from the tests A-D and AB-D

	Precision			Recall		
	A-D	AB-D		A-D	AB-D	
Test	SVM	PM	SVM	SVM	PM	SVM
Cat1	0.77	**0.89**	**0.75**	0.51	0.55	0.55
Cat2	0.74	**0.92**	**0.81**	0.66	*0.62*	*0.77*
Cat3	0.83	**0.90**	**0.76**	0.76	**0.82**	**0.76**
Cat4	0.77	**0.91**	**0.78**	0.33	**0.40**	**0.37**
Cat5	0.68	**0.84**	**0.73**	0.66	*0.55*	*0.72*
Rest	0.87	*0.86*	*0.89*	0.92	**0.97**	**0.93**
Total	0.83	**0.86**	**0.85**	0.84	0.86	0.86

Note. SVM = support vector machine; PM = text-pattern matching system.

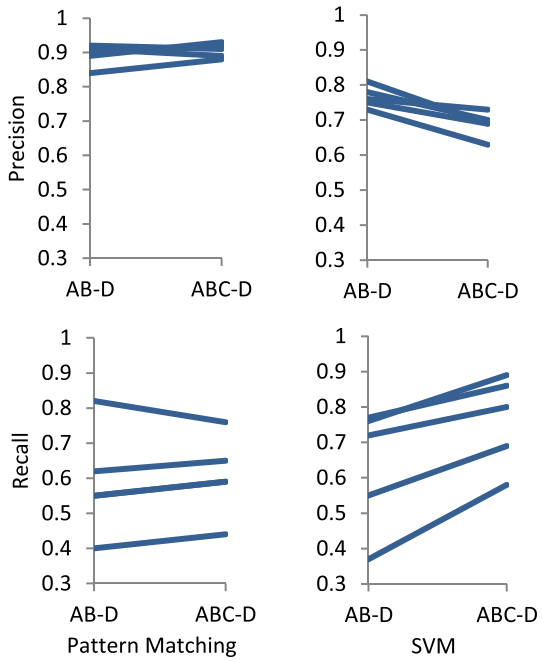

FIGURE 6 Precision and recall trends for Cat1 through Cat5 between the tests AB-D and ABC-D. The recall values for the text-pattern matching in Cat1 and Cat5 are identical, therefore the lines overlap. SVM = support vector machine

the user's intent; more "training" data leads to more refined expressions. Likewise, the updated text patterns improve recall because the variety of the multiword expressions increases along with more training data, yet this increased variety does not allow as much rise of recall as more training data does for SVM and the bag-of-words document representation.

7.3 | Precision versus recall, *F*-score

The main advantage of the text-pattern matching over SVM is higher precision. Precision differences between the text-pattern matching and SVM across 20 measurements (five text categories in the four tests A-B, AB-C, ABC-D, AB-D) range from 0.07 to 0.25. The average precision difference is 0.17.

SVM has higher average recall. Recall differences between the text-pattern matching and SVM across the 20 measurements range from −0.09 (the text-pattern matching has better recall) to 0.27 (SVM has better recall). The average recall difference is 0.09 in favour of SVM.

The average precision advantage of the text-pattern matching is twice as large as the average recall advantage of SVM, which leads to an F-score advantage shown in Figure 7. The explanation of such a strong precision advantage was outlined in the previous section: The text patterns (a) match meaningful multiword expressions and (b) capture the context description and the response trigger in a query message, whereas SVM matches document-size bags-of-words.

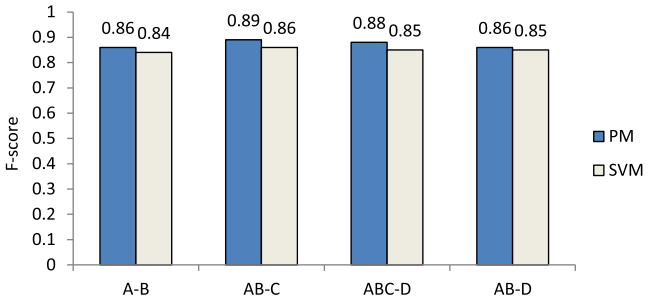

FIGURE 7 Text-pattern matching (PM) versus support vector machine (SVM): The total *F*-score in the entire test collection

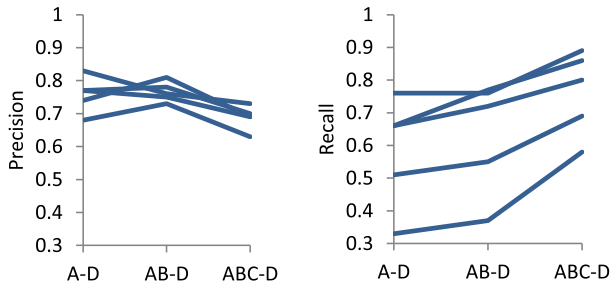

FIGURE 8 Precision and recall trends for Cat1 through Cat5 during the tests A-D, AB-D, ABC-D for support vector machine

7.4 | A-D, AB-D, and ABC-D for SVM

Figure 8 shows precision and recall trends across the consecutive tests A-D, AB-D, and ABC-D for SVM. With more training data, the precision generally decreases, whereas the recall generally improves. Because Cat1 through Cat5 are relatively small text categories (e.g., Cat4 has only 29 training messages in the first test, 59 in the second, and 81 in the last test), SVM gets more aggressive about classifying emails into these categories. This means more true-positive category placements—higher recall. And because the amount of the training data is still small, also more false-positive category placements—lower precision.

7.5 | Highlights

Figure 9 visualizes the highest and lowest precision and recall values obtained for any text category Cat1 trough Cat5 during any test defined in Table 3. The text-pattern matching figures come from Tables 5 and 6; the SVM figures come from Tables 14 and 15.

For the text-pattern matching, precision never drops below 0.84, which is the value obtained for Cat5 during the tests A-B and AB-D; therefore, it has two corresponding recall values. The highest precision value is 1; it was obtained during the test AB-C for Cat2 and Cat4; therefore, it also has two corresponding recall values. The lowest and highest recall values are 0.40 (Cat4, AB-D) and 0.85 (Cat3, AB-C), respectively.

For SVM, the lowest and highest precision values were obtained during the test A-B; they are 0.60 (Cat1) and 0.89 (Cat3). The lowest and highest recall values are 0.33 (Cat4, A-D) and 0.89 (Cat3, ABC-D), respectively.

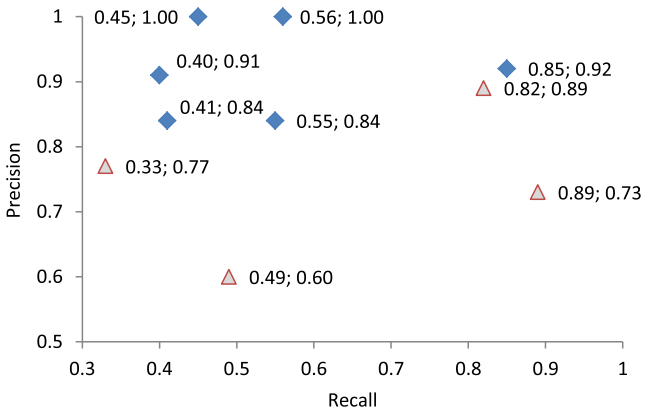

FIGURE 9 Lowest and highest (recall; precision). Diamonds represent the values of the text-pattern matching; triangles—Those of support vector machine

8 | TEST RESULTS IN CONTEXT

Spam filtering is by far the most well known activity related to email categorization. Spam filters may employ Naïve Bayes (first applied by Sahami, Dumais, Heckerman, & Horvitz, [1998], then improved by Graham [2003]), SVM (Amayri & Bouguila, 2010), decision trees (Zhang, Wang, Phillips, & Ji, 2014), and at least seven other methods (Guzella & Caminhas, 2009). Email categorization into folders, such as the work by Bekkerman (2004), seems to be the next most popular subject in this line of research. Task-related, as opposed to topic-related, email categorization was discussed in Section 2. This section summarizes performance figures found in closely related research, which is FAQ retrieval based on text-pattern matching and automated email answering, and provides the context for our own performance figures. The main source of information regarding email answering was the recently published review of the main approaches to automated email answering (Sneiders, 2016a).

Our text-pattern matching technique has a history in FAQ retrieval (Sneiders, 2009) and two email-answering applications (Sneiders, 2010). For the sake of comparison, we joined Cat1 through Cat5 into one quasicategory and applied the test AB-C to this quasicategory (AB-C yielded the best precision and recall among the tests in Table 3). Table 16 summarizes the performance figures of the four applications.

Please observe that precision and recall in Sneiders (2009, 2010) were calculated according to the tradition of document retrieval for each query separately in the set of retrieved documents, and then the average values were made. These average values are not completely comparable with precision and recall in text categorization.

The previous experiments with machine learning techniques for answer-specific email categorization demonstrate a wide range of results shown in Table 17. Notably, Yang and Kwok (2012) have categorization accuracy between 75.5% and 96.2% with k-NN and K-means++, which is a large difference.

Calculation of statistical text similarity without machine learning (Table 18) is based on presence of the query terms in the document. If the technique matches representative *n*-grams or concepts that cover multiple terms, then the technique needs a "training" phase where these *n*-grams and concepts are developed. Such training is equivalent to our own text-pattern development.

The results in Tables 17 and 18 are not directly comparable with our results because the measuring methods are different (e.g., precision and recall are not directly comparable with receiver operating characteristic) and because the data used in the experiments are different. Some data are easier to classify than other data in general, and some data may suite one method better than another method. Even though the results are not directly comparable, they give a rough idea of what can be expected. For example, we see that precision, recall, and categorization accuracy tend to lie way below 100%. Therefore, our own reference precision in Table 16, which is 92.6%, looks like a very good result. On the other hand, we see

TABLE 16 Text-pattern matching: The same software, four applications

	Categorization of email messages in the Cat1-Cat5 quasicategory, Swedish	Retrieval of email answers, insurance case, Swedish	Retrieval of email answers, telecom case, Latvian	Retrieval, FAQ answering, Swedish
Precision	0.926 remotely comparable	0.987	0.98	0.96
Recall	0.633 somewhat comparable	0.682	0.758	0.91

TABLE 17 Machine leaning for email categorization

Technique	Performance
SVM (Busemann et al., 2000)	4,490 messages in 47 categories. Accuracy 56%.
SVM (Scheffer, 2004)	528 messages in 9 categories. ROC curve, area under the curve 80–95%.
SVM (Bickel & Scheffer, 2004)	805 messages in 19 categories. Accuracy 42%.
k-NN, Naïve Bayes, ripper (Lapalme & Kosseim, 2003)	1,568 messages. Accuracy around 50%.
K-means++, Naïve Bayes, k-NN (Yang & Kwok, 2012)	3,015 messages in 200 categories. Accuracy: K-means++ 96.2%, Naïve Bayes 89.5%, k-NN 75.5%.
Jaccard coefficient and SVM (Itakura et al., 2010)	1,845 messages in 4 categories. Precision 60–98%, recall 70–95%.

Note. SVM = support vector machine; ROC = receiver operating characteristic.

TABLE 18 Statistical text similarity without machine learning

Technique	Performance
Matching of domain-specific uni, bi, and trigrams to the query (Malik, Subramaniam, & Kaushik, 2007)	400 test messages. 61% correct replies, 73.4% partially correct replies.
Cosine similarity between the query and archived messages (Lapalme & Kosseim, 2003)	Precision 57.9%. After query expansion 62%.
Matching of domain concepts with weighted terms attached to the answers (Weng & Liu, 2004)	Precision and recall around 80%.
Matching of email messages to webpages by matching of term pairs in the query to term pairs in the sentences of a webpage (Sneiders, 2016b)	Recall 83%. The average position of relevant webpages in the results list is 3.38 ± 3.93.

that precision, recall, and categorization accuracy above 50% should be easily achievable. Our own reference recall 63.3% looks like an average achievement.

9 | CONCLUSIONS

The main contributions of this article are measuring the performance of a text-pattern matching system for automated email answering and subsequent error analysis. Furthermore, the article introduces context description and response trigger as two essential components of an email inquiry sent to a contact centre. Both components are explicitly identified by the system, which arguably improves the answer accuracy.

We treated the task of automated email answering as a task of text categorization into answer-specific text categories and had precision and recall as the main measures of answer (i.e., text categorization) accuracy. The text patterns, specific for each text category, were manually crafted and refined considering both precision and recall, with precision having the priority. The highest and lowest precision values for any one text category were 1 (at recall 0.45 and 0.56) and 0.84 (at recall 0.41 and 0.55), respectively, (see details in Section 5.1, summary in Section 7.5). In the scope of the entire test collection, precision shifted between 0.86 and 0.89. The same text-pattern matching technique in two other domains and two languages (Sneiders, 2010) has demonstrated similar answer accuracy. Our conclusion is that matching of manually crafted text patterns to incoming email messages can achieve high precision at reasonable recall, both suited for automated email answering.

The baseline system—SVM with the bag-of-words document representation—has demonstrated lower precision but higher recall than that of the text-pattern matching (see details in Sections 7.1 and 7.2, summary in Section 7.5). The precision advantage of the text-patterns matching is twice as large as the recall advantage of SVM (see Section 7.3).

An email flow constantly provides us with new messages. Our experiments show that more "training" data may lead to inferior performance of the text-pattern matching system if different test data are used (see Table 5 and Figure 5). With the same test data and more "training" data, the system demonstrated statistically significant performance improvement (see Table 6 and Figure 6).

If the text-pattern matching system failed, more often, it did not deliver the right answer rather than delivered wrong one (see Section 5.2). In total, there were nine reasons why the system did not deliver an answer. Most often, in 55.3% of the undelivered answers, there was no applicable text pattern in the system's database, or the difference between the query text and the potentially applicable text pattern was significant. Twenty-four percent of the query messages that should have been answered, but were not, had an almost matching text pattern; the match did not happen because of one missing word or expression. We believe the text patterns in the latter case could be fixed by a semiautomated tool (such tool does not yet exist).

There were three reasons of a wrong answer: Either (a) the overall context of the query message invalidated the standard answer, or (b) a subordinate clause of a matching sentence in the query message changed the expected meaning of the sentence and invalidated the standard answer, or (c) one word or a phrase in a matching sentence did the same.

In the future, it would be interesting to do similar experiments on other data sets, for instance, in other languages or in domains with different vocabulary, to see if the results are sensitive to the type of emails to classify. It would also be interesting to have different people create text patterns based on the same training data and see how sensitive the results are to the text pattern creator.

ORCID

Eriks Sneiders http://orcid.org/0000-0002-2803-5139

REFERENCES

Aggarwal, C. C., & Zhai, C. (2012). A survey of text classification algorithms. In *Mining text data* (pp. 163–222). https://doi.org/10.1007/978-1-4614-3223-4_6

Amayri, O., & Bouguila, N. (2010). A study of spam filtering using support vector machines. *Artificial Intelligence Review*, *34*(1), 73–108. https://doi.org/10.1007/s10462-010-9166-x

Bekkerman, R. (2004). Automatic categorization of email into folders: Benchmark experiments on Enron and SRI corpora. In *Computer Science Department faculty publication series* (Vol. 218). Retrieved from http://scholarworks.umass.edu/cs_faculty_pubs/218

Bickel, S., & Scheffer, T. (2004). Learning from message pairs for automatic email answering. In J. F. Boulicaut, F. Esposito, F. Giannotti, & D. Pedreschi (Eds.), *Machine learning: ECML 2004* (Vol. 3201*Lecture Notes in Computer Science*) (pp. 87–98). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-540-30115-8_11

Burke, R. D., Hammond, K. J., Kulyukin, V., Lytinen, S. L., Tomuro, N., & Schoenberg, S. (1997). Question answering from frequently asked question files: Experiences with the FAQ Finder system. *AI Magazine*, *18*(2), 57–66. https://doi.org/10.1609/aimag.v18i2.1294

Busemann, S., Schmeier, S., & Arens, R. G. (2000, April). Message classification in the call center. In *Proceedings of the sixth conference on applied natural language processing* (pp. 158–165). Stroudsburg, PA, USA: Association for Computational Linguistics. https://doi.org/10.3115/974147.974169

Byvatov, E., Fechner, U., Sadowski, J., & Schneider, G. (2003). Comparison of support vector machine and artificial neural network systems for drug/nondrug classification. *Journal of Chemical Information and Computer Sciences*, *43*(6), 1882–1889. https://doi.org/10.1021/ci0341161

Carlberger, J., & Kann, V. (1999). Implementing an efficient part-of-speech tagger. *Software-Practice and Experience*, *29*(9), 815–832. doi.org/10.1002/(SICI)1097-024X(19990725)29:9%3C815::AID-SPE256%3E3.0.CO;2-F

Cohen, W. W., Carvalho, V. R., & Mitchell, T. M. (2004, July). Learning to Classify Email into "Speech Acts". In *Proceedings of Empirical Methods in Natural Language Processing*. EMNLP'2004 (Vol. 4, pp. 309–316).

Colas, F., & Brazdil, P. (2006). Comparison of SVM and some older classification algorithms in text classification tasks. In M. Bramer (Ed.), *Artificial intelligence in theory and practice* (Vol. 217*IFIP AI 2006. IFIP International Federation for Information Processing*) (pp. 169–178). Boston, MA, USA: Springer. https://doi.org/10.1007/978-0-387-34747-9_18

Corston-Oliver, S., Ringger, E., Gamon, M., & Campbell, R. (2004, July). Task-focused summarization of email. In *Proceedings of ACL-04 Workshop: Text Summarization Branches Out* (pp. 43–50).

Dalianis, H., Sjöbergh, J., & Sneiders, E. (2011, February). Comparing manual text patterns and machine learning for classification of e-mails for automatic answering by a government agency. In A. Gelbukh (Ed.), *Computational linguistics and intelligent text processing* (Vol. 6609*CICLing'2011. Lecture Notes in Computer Science*) (pp. 234–243). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-19437-5_19

Domeij, R. (1994). Språkgranskning med mönstermatchning i reguljära uttryck. *Technical report TRITA-NA-P9402*, NADA, Kungliga Tekniska Högskolan, Stockholm.

Email Statistics Report, 2015–2019. (2015, March). Retrieved from http://www.radicati.com/wp/wp-content/uploads/2015/02/Email-Statistics-Report-2015-2019-Executive-Summary.pdf

Everitt, B. S. (1977). *The analysis of contingency tables*. London, UK: Chapman & Hall.

Felzmann, S. (2015). *Automated text pattern extraction in the area of email answering*. Master thesis at the Department of Computer and Systems Sciences, Stockholm University, Sweden.

Goldstein, J., & Sabin, R. E. (2006, January). Using speech acts to categorize email and identify email genres. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences* (Vol. 3*HICSS'2006*) (pp. 50b–50b)IEEE. https://doi.org/10.1109/HICSS.2006.528

Graham, P. (2003). *Better Bayesian filtering*. In *Proceedings of Spam Conference, 2003*. Retrieved from http://www.paulgraham.com/better.html

Guzella, T. S., & Caminhas, W. M. (2009). A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, *36*(7), 10206–10222. https://doi.org/10.1016/j.eswa.2009.02.037

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, *11*(1), 10–18. https://doi.org/10.1145/1656274.1656278

He, Z. (2008). SMS in China: A major carrier of the nonofficial discourse universe. *The Information Society*, *24*(3), 182–190. https://doi.org/10.1080/01972240802020101

Henkel, M., Perjons, E., & Sneiders, E. (2017). Examining the potential of language technologies in public organizations by means of a business and IT architecture model. *International Journal of Information Management*, *37*(1), 1507–1516. https://doi.org/10.1016/j.ijinfomgt.2016.05.008

Huang, Z., Chen, H., Hsu, C. J., Chen, W. H., & Wu, S. (2004). Credit rating analysis with support vector machines and neural networks: A market comparative study. *Decision Support Systems*, *37*(4), 543–558. https://doi.org/10.1016/S0167-9236(03)00086-1

Itakura, K., Kenmotsu, M., Oka, H., & Akiyoshi, M. (2010, November). An identification method of inquiry e-mails to the matching FAQ for automatic question answering. In F. de Carvalho, A. P. de Leon, S. Rodríguez-González, J. F. De Paz Santana, J. M. C. Rodríguez (Eds.), *Distributed computing and artificial intelligence. Advances in intelligent and soft computing* (Vol. 79, pp. 213–219). Berlin, Heidelberg: Springer. DOI: https://doi.org/10.1007/978-3-642-14883-5_28

Iwai, K., Iida, K., Akiyoshi, M., & Komoda, N. (2010, July). A help desk support system with filtering and reusing e-mails. In *Proceedings of the 8th IEEE International Conference on Industrial Informatics* (*INDIN'2010*) (pp. 321–325)IEEE. https://doi.org/10.1109/INDIN.2010.5549401

Khosravi, H., & Wilks, Y. (1999). Routing email automatically by purpose not topic. *Natural Language Engineering*, *5*(3), 237–250.

Knutsson, O., Pargman, T.C., Dalianis, H., Rosell, M., & Sneiders, E. (2010). Increasing the efficiency and quality of e-mail communication in e-government using language technology. Retrieved from https://www.semanticscholar.org/paper/Increasing-the-Efficiency-and-Quality-of-E-mail-Co-Knutsson-Pargman/724131db1b257d54b5c9e15e46447ffaa5c8dc95

Kosseim, L., Beauregard, S., & Lapalme, G. (2001). Using information extraction and natural language generation to answer e-mail. *Data & Knowledge Engineering*, *38*(1), 85–100. https://doi.org/10.1016/S0169-023X(01)00018-0

Kotadia, M. (2007, August 23). Aussies prefer robots to call centres. Retrieved from http://www.zdnet.com/article/aussies-prefer-robots-to-call-centres/

Lampert, A., Dale, R., & Paris, C. (2008a, July). The nature of requests and commitments in email messages. In *Enhanced Messaging: Papers from the 2008 AAAI Workshop*. Technical Report WS-08-04 (pp. 42–47). AAAI Press.

Lampert, A., Dale, R., & Paris, C. (2008b, December). Requests and commitments in email are more complex than you think: Eight reasons to be cautious. In *Proceedings of Australasian Language Technology Association Workshop 2008* (Vol. 6, pp. 64–72).

Lapalme, G., & Kosseim, L. (2003). Mercure: Towards an automatic e-mail follow-up system. *IEEE Computational Intelligence Bulletin*, *2*(1), 14–18.

Lewan, M. (2008, November 25). Hjälpsamma svenskor tar över kundservicen. Retrieved from https://www.nyteknik.se/digitalisering/hjalpsamma-svenskor-tar-over-kundservicen-6413319

Malik, R., Subramaniam, L. V., & Kaushik, S. (2007, January). Automatically selecting answer templates to respond to customer emails. In R. Sangal, H. Mehta, & R. K. Bagga (Eds.), *Proceedings of the 20th international joint conference on Artifical intelligence* (*IJCAI'2007*) (pp. 1659–1664). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Mametja, L. (2017). *Comparison of email text patterns*. Master thesis at the Department of Computer and Systems Sciences, Stockholm University, Sweden.

Manning, C. D., Raghavan, P., & Schütze, H. (2009). *Introduction to information retrieval*. Cambridge: Cambridge University Press.

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998, July). A Bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 AAAI Workshop*, Technical Report WS-98-05. AAAI Press.

Sahlgren, M. (2006). *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. PhD thesis at the Department of Linguistics, Stockholm University, Sweden.

Scheffer, T. (2004). Email answering assistance by semi-supervised text classification. *Intelligent Data Analysis*, *8*(5), 481–493.

Sjöbergh, J., & Kann, V. (2004). Finding the correct interpretation of Swedish compounds, a Statistical Approach. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*. LREC'2004 (pp. 899–902).

Sneiders, E. (1999, November). Automated FAQ answering: Continued experience with shallow language understanding. In *Question Answering Systems: Papers from the 1999 AAAI Fall Symposium*, Technical Report FS-99-02 (pp. 97–107). AAAI Press.

Sneiders, E. (2002, June). Automated question answering using question templates that cover the conceptual model of the database. In B. Andersson, M. Bergholtz, & P. Johannesson (Eds.), *Natural language processing and information systems* (Vol. 2553*NLDB'2002. Lecture Notes in Computer Science*) (pp. 235–239). Berlin, Heidelberg: Springer. https://doi.org/10.1007/3-540-36271-1_24

Sneiders, E. (2009, May). Automated FAQ answering with question-specific knowledge representation for web self-service. In *Proceedings of the 2nd Conference on Human System Interactions, 2009* (*HSI'2009*) (pp. 298–305)IEEE. https://doi.org/10.1109/HSI.2009.5090996

Sneiders, E. (2010). Automated email answering by text pattern matching. In H. Loftsson, E. Rögnvaldsson, & S. Helgadóttir (Eds.), *Advances in Natural Language Processing. Proceedings of the 7th International Conference on NLP* (Vol. 6233*IceTAL'2010. Lecture Notes in Computer Science*) (pp. 381–392). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-14770-8_41

Sneiders, E. (2016a, March). Review of the main approaches to automated email answering. In Á. Rocha, A. Correia, H. Adeli, L. Reis, & M. Mendonça Teixeira (Eds.), *New advances in information systems and technologies*. WorldCIST'2016. Advances in Intelligent Systems and Computing (Vol. 444) (pp. 135–144). Cham: Springer. https://doi.org/10.1007/978-3-319-31232-3_13

Sneiders, E. (2016b, December). Text retrieval by term co-occurrences in a query-based vector space. In *Proceedings of COLING 2016: Technical Papers*. The 26th International Conference on Computational Linguistics, COLING'2016 (pp. 2356–2365).

Sneiders, E., Eriksson, G., & Alfalahi, A. (2014, September). Exploring the traits of manual e-mail categorization text patterns. In A. Przepiórkowski, & M. Ogrodniczuk (Eds.), *Advances in natural language processing* (Vol. 8686*Proceedings of the 9th International Conference on NLP, PolTAL'2014. Lecture Notes in Computer Science*) (pp. 337–344). Cham: Springer. https://doi.org/10.1007/978-3-319-10888-9_34

Sneiders, E., Sjöbergh, J., & Alfalahi, A. (2016, March). Email answering by matching question and context-specific text patterns: Performance and error analysis. In Á. Rocha, A. Correia, H. Adeli, L. Reis, & M. Mendonça Teixeira (Eds.), *New advances in information systems and technologies*. WorldCIST'2016. Advances in Intelligent Systems and Computing (Vol. 444) (pp. 7–133). Cham: Springer. https://doi.org/10.1007/978-3-319-31232-3_12

Weng, S.-S., & Liu, C.-K. (2004). Using text classification and multiple concepts to answer e-mails. *Expert Systems with Applications*, *26*(4), 529–543. https://doi.org/10.1016/j.eswa.2003.10.011

Yang, W., & Kwok, L. (2012, October). Improving the automatic email responding system for computer manufacturers via machine learning. In *Proceedings of the 12th International Conference on Information Management, Innovation Management and Industrial Engineering (ICIII)* (Vol. 3) (pp. 487–491)IEEE. https://doi.org/10.1109/ICIII.2012.6340024

Youn, S., & McLeod, D. (2007). A comparative study for email classification. In K. Elleithy (Ed.), *Advances and innovations in systems, computing sciences and software engineering* (pp. 387–391). Springer: Dordrecht. https://doi.org/10.1007/978-1-4020-6264-3_67

Zaghloul, W., Lee, S. M., & Trimi, S. (2009). Text classification: Neural networks vs support vector machines. *Industrial Management & Data Systems*, *109*(5), 708–717. https://doi.org/10.1108/02635570910957669

Zhang, Y., Phillips, P., Wang, S., Ji, G., Yang, J., & Wu, J. (2016). Fruit classification by biogeography-based optimization and feedforward neural network. *Expert Systems*, *33*(3), 239–253. https://doi.org/10.1111/exsy.12146

Zhang, Y., Wang, S., Phillips, P., & Ji, G. (2014). Binary PSO with mutation operator for feature selection using decision tree applied to spam detection. *Knowledge-Based Systems*, *64*, 22–31. https://doi.org/10.1016/j.knosys.2014.03.015

Eriks Sneiders is a researcher and senior lecturer at Stockholm University. His research interests started with automated question answering and gradually extended to automated email answering. Eriks has built, tested and deployed several question-answering and email-answering tools for domains such as insurance, telecom, social care, and has investigated how they improve the quality and efficiency of contact centres at public and private organizations. Eriks has a PhD in Computer Science from the Royal Institute of Technology (KTH), Stockholm, Sweden.

Jonas Sjöbergh received a Ph.D. in Computer Science from the Royal Institute of Technology (KTH), Stockholm, Sweden. He currently works at Hokkaido University in Sapporo, Japan. His research covers topics such as machine learning for natural language processing, artificial humour, and big data.

Alyaa Alfalahi has her background in teaching programming. As a research assistant at Stockholm University, she was involved in projects which investigated how language technology can improve the quality of the communication between citizens and public organizations.

How to cite this article: Sneiders E, Sjöbergh J, Alfalahi A. Automated email answering by text-pattern matching: Performance and error analysis. *Expert Systems*. 2017;e12251. https://doi.org/10.1111/exsy.12251

APPENDIX A

The tables below show detailed results of the text-pattern matching tests introduced in Section 4.3. “Num relevant” is the number of messages assigned to each text category in the test collection by human annotators. “Correctly or incorrectly placed” is the number of correct or incorrect messages that the system placed into each category during the test.

TABLE A.1 Results of the test A-B

	Num relevant	Correctly placed	Incorrectly placed	Precision	Recall	*F*-score
Cat1	76	41	4	0.91	0.54	0.68
Cat2	62	33	1	0.97	0.53	0.69
Cat3	49	27	1	0.96	0.55	0.70
Cat4	30	15	2	0.88	0.50	0.63
Cat5	269	111	21	0.84	0.41	0.55
Rest	1,490	1,467	247	0.86	0.99	0.92
Total	1,976	1,694	276	0.86	0.86	0.86

TABLE A.2 Results of the test AB-C

	Num relevant	Correctly placed	Incorrectly placed	Precision	Recall	*F*-score
Cat1	109	69	6	0.92	0.63	0.75
Cat2	105	59	0	1.00	0.56	0.72
Cat3	53	45	4	0.92	0.85	0.88
Cat4	22	10	0	1.00	0.45	0.63
Cat5	387	245	24	0.91	0.63	0.75
Rest	1,803	1,776	242	0.88	0.99	0.93
Total	2,479	2,204	276	0.89	0.89	0.89

TABLE A.3 Results of the test ABC-D

	Num relevant	Correctly placed	Incorrectly placed	Precision	Recall	*F*-score
Cat1	148	87	7	0.93	0.59	0.72
Cat2	79	51	5	0.91	0.65	0.76
Cat3	45	34	3	0.92	0.76	0.83
Cat4	78	34	4	0.89	0.44	0.59
Cat5	393	233	31	0.88	0.59	0.71
Rest	2,051	2,005	295	0.87	0.98	0.92
Total	2,794	2,444	345	0.88	0.87	0.88

TABLE A.4 Results of the test AB-D

	Num relevant	Correctly placed	Incorrectly placed	Precision	Recall	*F*-score
Cat1	148	81	10	0.89	0.55	0.68
Cat2	79	49	4	0.92	0.62	0.74
Cat3	45	37	4	0.90	0.82	0.86
Cat4	78	31	3	0.91	0.40	0.55
Cat5	393	217	42	0.84	0.55	0.67
Rest	2,051	1,997	318	0.86	0.97	0.91
Total	2,794	2,412	381	0.86	0.86	0.86