A Meta-learning Approach for Selecting between Response Automation Strategies in a Help-desk Domain

Yuval Marom[†] and **Ingrid Zukerman**[†] and **Nathalie Japkowicz**[‡]
[†] Faculty of Information Technology
Monash University
Clayton, Victoria 3800, Australia
yuvalmarom@gmail.com, ingrid@csse.monash.edu.au

[‡] School of Information Technology and Eng.
University of Ottawa
Ottawa, Ontario, K1N 6N5, Canada
nat@site.uottawa.ca

Abstract

We present a corpus-based approach for the automation of help-desk responses to users' email requests. Automation is performed on the basis of the similarity between a request and previous requests, which affects both the content included in a response and the strategy used to produce it. The latter is the focus of this paper, which introduces a meta-learning mechanism that selects between different information-gathering strategies, such as document retrieval and multi-document summarization. Our results show that this mechanism outperforms a random strategy-selection policy, and performs competitively with a gold baseline that always selects the best strategy.

Introduction

The help-desk domain offers interesting challenges to response automation in that on one hand, responses are generalized to fit standard solutions, and on the other hand, they are tailored to the initiating request in order to meet specific customer needs. For example, the first sentence of the response in the left panel of Figure 1 is tailored to the user's request, and the remainder is generic. This means that responses sent to different customers contain varying degrees of overlap. Thus, generating a response for a new request may involve re-using an existing response in its entirety, putting together parts of responses that match individual components of the request, or composing a completely new response. This suggests that different strategies may be suitable for generating a response, depending on the content of the initiating request, and how well it matches previous requests. However, existing corpus-based approaches to help-desk response generation have considered only single strategies in isolation, e.g., (Carmel, Shtalhaim, & Soffer 2000; Berger & Mittal 2000).

In our previous work, we investigated several corpus-based response-generation methods separately. This was a good way of analyzing the strengths and weaknesses of each method, but it did not yield a fully automated solution to the problem. Here, we follow up on this previous work by investigating a way to integrate these methods within a unified system. Specifically, we consider a meta-learning mechanism that decides on the most suitable response-generation method for a particular request on the basis of (1) the similarity between this request and previous requests, and (2) the previous performance of the methods in question.

The rest of the paper is organized as follows. In the next section, we describe our domain and corpus, and present our response-generation strategies. We then introduce our meta-learning mechanism, followed by our evaluation. In the last two sections, we discuss related work and present concluding remarks.

Response-generation Methods

Our work is based on a corpus of email dialogues between customers and help-desk operators at Hewlett-Packard. The complete corpus consists of 30,000 email dialogues, but to focus our work we used a sub-corpus of 6,659 two-turn dialogues where the answers were reasonably concise (15 lines at most). These dialogues deal with a variety of customer requests, which include requests for technical assistance, inquiries about products, and queries about how to return faulty products or parts.

In previous work, we showed that features of our corpus suggest different response-generation methods (Zukerman & Marom 2006). A key observation we made from our corpus was that requests containing precise information, such as product names or part specifications, sometimes elicit helpful, precise answers referring to this information, while other times they elicit answers that do not refer to the query terms, but contain generic information (e.g., referring customers to another help group). The examples in the left and center panels in Figure 1 illustrate the first situation, while the example in the right panel illustrates the second situation. This observation suggests two main strategies for generating responses: *retrieval* and *prediction*. Retrieval returns an information item by matching its terms to query terms (Salton & McGill 1983). In contrast, prediction uses correlations between features of requests and responses to select an information item. The example in the right panel of Figure 1 illustrates a request-response pair where the terms in the response do not match any of the terms in the request, but a few of the terms in the request are predictive of the response (terms such as "firewall", "CP-2W" and "network" indicate that the query is network-related and should be redirected).

There is an internal battery inside this ipaq which I believe has to be sent in, in order to have it replaced. The model is i3650, serial# 4G12DW36K9RK. . . . The approximate cost for replacing the battery would be $120.00. Please call technical support at 888- phone- number , options 3, 1. If you are a first time caller, . . .	*Do I need Compaq driver software for my armada 1500 docking station? This in order to be able to re-install win 98?* I would recommend to install the latest system rompaq, on the laptop and the docking station. Just select the model of computer.	*Is there a way to disable the NAT firewall on the CP-2W so I don't get a private ip address through the wireless network?* Unfortunately, you have reached the incorrect eResponse queue for your unit. Your device is supported at the following link, or at 888- phone- number .

Figure 1: Sample request-response pairs.

Another key observation we made from our corpus is that while there is high language variability in the requests written by users, the responses exhibit strong regularities, mainly due to the fact that operators are equipped with in-house manuals containing prescribed answers. We have observed that these regularities can occur at different levels of granularity, with two particular granularities of interest: *document* and *sentence*.

These two observations led us to posit response-generation methods that combine the above strategies – retrieval and prediction – with the two levels of granularity – document and sentence. Here we describe the most successful of these methods.

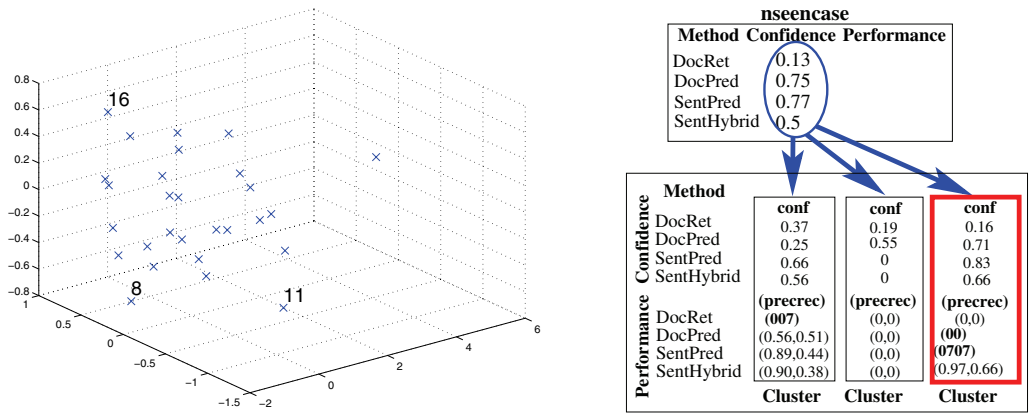
- *Document Retrieval (Doc-Ret)*. This method matches a new request with previous request-response pairs on the basis of the content terms in the request. Cosine similarity is used to calculate a retrieval score (Salton & McGill 1983), and a request-response pair is considered acceptable if the score exceeds a minimum threshold. The response from the top retrieved pair is then re-used for the new request. It is worth noting that we also tried matching requests with previous requests alone, and with previous responses alone, but the request-response combination proved most successful.
- *Document Prediction (Doc-Pred)*. This method first clusters response documents, and then learns mappings between terms in the request emails and the response document clusters. A new request is addressed by considering the response cluster with the highest prediction score, and if this score is higher than a confidence threshold, a representative response document (closest to the centroid) is selected as the response to the request.
- *Sentence Prediction (Sent-Pred)*. This method is similar to *Doc-Pred*, but it starts by clustering response sentences, rather than complete documents. It then produces a response for a new request by considering all the sentence clusters that are predicted with sufficient confidence, extracting a representative sentence (closest to the centroid) from each of these clusters, and employing multi-document summarization techniques to collate these sentences into a single response.
- *Sentence Prediction-Retrieval Hybrid (Sent-Hybrid)*. This method extends *Sent-Pred* by means of a retrieval mechanism, which selects a sentence from each promising cluster based on how well the sentence matches the terms in a request (rather than selecting the sentence closest to the centroid). This retrieval mechanism is activated when a representative sentence cannot be confidently selected from a cluster, which happens when the cluster is not sufficiently uniform (and hence a sentence close to the centroid is not representative).

Confidence and Performance

The details of the implementation of the various methods and their evaluation are presented in (Zukerman & Marom 2006; Marom & Zukerman 2007). For the purpose of this paper it is important to point out that each of the methods has its own confidence measure for producing a response to a given request. The two sentence-level methods (*Sent-Pred* and *Sent-Hybrid*) can produce partial responses, which happens when there is insufficient evidence to predict all the sentences required for a complete response. These methods have a high confidence in the response so long as they are confident about each of the individual sentences. In contrast, the document-level methods (*Doc-Ret* and *Doc-Pred*) either produce a complete response or do not produce any response.

Our evaluation showed that each of the methods can confidently address a substantial portion of the requests in our corpus, ranging from 34% to 43%, while overall the methods address a combined 72%. Further analysis showed that some requests were uniquely addressed by a specific method, while others could be addressed by more than one method. For the latter case, we compared the individual performance of the methods in terms of the quality of the generated responses. This showed that although the document-level methods were favoured because they produced complete responses, the sentence-level methods provided a useful alternative when a high-quality complete response could not be produced.

Response quality was measured automatically, where we performed a textual comparison between a generated re-



(a) Dimensions produced by PCA

(b) Centroids of three clusters

Figure 2: Clusters obtained from the training set.

sponse and the response provided by a help-desk operator for the request in question. This involved using the precision, recall and F-score metrics from Information Retrieval (Salton & McGill 1983). Precision measures how correct a generated response is (what proportion of its content terms appear in the operator's response), while recall measures how complete it is (what proportion of the content terms in the operator's response appear in the generated response). F-score is the harmonic mean of precision and recall, which thus provides an overall indication of how correct and complete a response is. We validated the automatic evaluation with a small user study (Marom & Zukerman 2007).

In this paper, we extend our previous work by developing a process which can automatically select one of the response-generation methods to address a new request. The individual confidence measures are not comparable to each other (e.g., the retrieval score in *Doc-Ret* is different to the prediction probability in *Doc-Pred*), and so we cannot simply pick the method with highest confidence. However, the performances of the different methods are comparable. Therefore, we would like to establish a link between confidence and performance. In other words, we need a meta-level process that can predict the performance of a method from its confidence, based on previous experience.

Meta-learning

The aim of the meta-learning component is to be able to predict which of the different methods performs best in handling an unseen request, given their individual levels of confidence. Following Lekakos and Giaglis (2007), one approach is supervised learning, where a winning method is selected for each case in the training set, the training case is labelled accordingly, and the system is then trained to predict a winner for unseen cases. In our situation, there is not always one single winner (two methods can perform similarly well for a given request), and there are different ways to pick winners (for example, based on F-score or precision). Therefore, such an approach would require the utilization of subjective heuristics for creating labels, which would significantly influence what is being learned.

Instead, we take an unsupervised approach that finds patterns in the data (confidence values coupled with performance scores), and then attempts to fit unseen data to these patterns in order to make a decision. Heuristics are still needed to make this decision, but they are applied by users of the system (i.e., the organization running the help-desk) only after the learning is complete (discussed below). These users need to make subjective decisions, such as what is a good performance and whether multiple methods are equally suitable. The advantages of an unsupervised approach are firstly that the effort required in setting these subjective criteria is reduced, and secondly, any future changes in these criteria will not require re-training the system.

Training

We train the system by clustering the "experiences" of the methods in addressing requests, where each experience is characterized by the confidence value of each method and its subsequent performance, reflected by precision and recall. To this effect, we use the clustering program Snob, which performs mixture modelling coupled with model selection based on the Minimum Message Length criterion (Wallace 2005).[1] The program decides what is the best clustering of the data, taking into account the statistical distribution of the data. To illustrate what we get from the clustering, consider Figure 2(a), which shows the clusters produced by Snob. The figure is a projection of the centroids of these clusters onto the three most significant dimensions discovered by Principal Component Analysis (PCA).[2] The top part of Figure 2(b) is discussed in the Testing section. The bottom part of Figure 2(b) shows the (unprojected) centroid values (confidence and (precision,recall)) for three of the clusters. These clusters were chosen because they illustrate three different situations of interest.

Single winner: Cluster 8 shows the case where a single strategy is clearly preferred. In this case the winner is

[1] We prefer this program because one does not need to specify the number of clusters a-priori.

[2] These dimensions account for 95% of the variation in the data.

Doc-Ret: its precision and recall values in this cluster are 0.91 and 0.76, respectively.

No winner: Cluster 11 shows a case where none of the methods do well. They all result in precision and recall values of 0.

Multiple winners: In Cluster 16, both *Doc-Pred* and *Sent-Pred* are competitive, exhibiting precision and recall values of (0.90, 0.89) and (0.97, 0.78), respectively. A decision between the two methods will depend on whether we have a preference for precision or recall, as we will see below.

Testing

We test the system with a set of unseen requests, which we feed to each of the response-generation methods. Each method produces a confidence value for each request. We do not know in advance how each method will perform — this information is missing, and we predict it on the basis of the clusters obtained from the training set. Our prediction of how well the different methods perform on an unseen case are based on (1) how well the unseen case fits each of the clusters, and (2) the average performance values in each cluster as indicated by its centroid.

The top part of Figure 2(b) shows an example of an unseen case, whose confidence values are most similar to those in the centroid of Cluster 16. If the match is very strong, we can select a method using the performance values in the centroid (this will depend on whether there is a preference towards precision or recall, as we will see below). However, we may not always have a strong match between an unseen case and a cluster. For example, the cluster closest to Cluster 16 is Cluster 15 (not labelled in Figure 2(a)). It contains similar confidence values, but its precision and recall values for *Doc-Pred* and *Sent-Pred* are (0.76, 0.66) and (0.84, 0.67) respectively, leading to a selection of *Sent-Pred* (regardless of any preferences for precision or recall).

We implement the testing step using Snob, which is able to accept data with missing values. It fits unseen data to the clusters obtained from training by maximizing the likelihood of these data, taking into account the missing information. Then, for each data point x we get $\Pr(c_i|x)$, the posterior probability of each cluster c_i given the new data point. This probability takes into account any uncertainty that arises when similar configurations of confidence values result in different performances. For example, for the unseen case in Figure 2(b), Snob may assign posterior probabilities of 0.5 and 0.3 to Clusters 16 and 15, respectively (and lower probabilities to weaker-matching clusters, such as Cluster 8). These posterior probabilities indicate how well an unseen case matches each of the clusters. We utilize this information in two alternative ways for calculating an overall estimate of performance: *Max*, which considers only the best-matching cluster; and *Weighted*, which considers all clusters, weighted by their posterior probabilities.

- **Max:** define $\widehat{p_k}$, the estimated precision of method k, as
$$\widehat{p_k} = p_k^{(i^*)}, \;\; i^* = \arg\max_i \Pr(c_i|x) \qquad (1)$$
where $p_k^{(i)}$ is the precision component for method k in the centroid of cluster i.

- **Weighted:** define $\widehat{p_k}$ as
$$\widehat{p_k} = \sum_i \Pr(c_i|x) \times p_k^{(i)} \qquad (2)$$

We perform a similar calculation for recall, and repeat for all the methods.

In order to select a method for a given request, we need to combine our estimates of precision and recall into an overall estimate of performance, and then choose the method with the best performance. The standard approach for combining precision and recall is to compute their harmonic mean — the F-score. However, in order to accommodate different levels of preference for precision or recall, as discussed above, we use the following weighted F-score calculation:

$$\text{F-score} = \left\{ \frac{w}{\text{Precision}} + \frac{1-w}{\text{Recall}} \right\}^{-1} \qquad (3)$$

where w is a weight between 0 and 1. When $w = 0.5$ we have the standard usage of F-score, and for values greater than 0.5, we have a preference for a high precision. For example, if we set $w = 0.5$, the precision and recall values of Cluster 16 above (Figure 2) translate to respective F-scores of 0.895 and 0.865 for *Doc-Pred* and *Sent-Pred*, leading to a choice of *Doc-Pred*. In contrast, if we set $w = 0.75$, the respective F-scores are 0.897 and 0.914, leading to a choice of *Sent-Pred*.

Evaluation

We evaluate the meta-learning system by looking at the quality of the response generated by the selected method. As we discussed and validated with a user study in our previous work, we believe that the main concern of an automated help-desk system is obtaining high precision. It is better to give partial but correct information than to misguide a user, especially when the user is aware that the response is automated. Therefore, we focus the evaluation on precision and F-score. The former indicates how correct a response is without penalizing it for incomplete information, while the latter indicates how correct and complete the response is. In order to evaluate the learning setup presented in the previous section, we employ a 5-fold cross-validation procedure to generate five different training and testing splits in our corpus.

Testing scenarios

We chose testing scenarios that compare the two alternative approaches for estimating performance (Equations 1 and 2), as well as the effect of favouring precision when selecting between methods via the F-score calculation (Equation 3). We also devised some baselines to help ground our results:

1. Random: select between the methods randomly.
2. Gold50: select between the methods based on their actual performance (as opposed to their estimated performance), using $w = 0.5$ in Equation 3.
3. Gold75: as above, but with $w = 0.75$.

The meta-learning scenarios are:

4. Weighted50: use the weighted alternative for estimating performance (Equation 2), with $w = 0.5$ in Equation 3.

Table 1: Averaged results (standard deviation in brackets).

	All cases		Cases with estimated precision ≥ 0.8		
	F-score Ave	**Precision Ave**	**F-score Ave**	**Precision (Ave)**	**Coverage**
Random	0.376 (0.33)	0.558 (0.37)	0.696 (0.25)	0.955 (0.06)	37.6%
Gold50	0.548 (0.30)	0.725 (0.26)	0.732 (0.26)	0.934 (0.06)	53.0%
Gold75	0.537 (0.29)	0.781 (0.25)	0.689 (0.26)	0.952 (0.06)	60.6%
Weighted50	0.512 (0.30)	0.727 (0.27)	0.649 (0.29)	0.874 (0.18)	57.1%
Weighted75	0.499 (0.28)	0.776 (0.26)	0.626 (0.27)	0.919 (0.16)	57.1%
Max50	0.507 (0.31)	0.704 (0.28)	0.648 (0.30)	0.844 (0.22)	56.7%
Max75	0.498 (0.28)	0.768 (0.27)	0.629 (0.27)	0.911 (0.17)	56.7%

5. Weighted75: as above, but with $w = 0.75$.
6. Max50: use the argmax alternative for estimating performance (Equation 1), with $w = 0.5$.
7. Max75: as above, but with $w = 0.75$.

Further, as we saw from Cluster 11 in Figure 2, the estimated performance can be very low for all the methods. Therefore, we repeat the above testing scenarios under a more practical setting, where the system has the choice not to select any methods if the estimated performance is poor. We envisage that a practical system would behave in this manner, in the sense that a request for which none of the existing methods can produce a response will be passed to an operator. As mentioned above, we consider precision to be an important practical criterion, and so the repeated tests are carried out by considering only cases where the estimated precision is above 0.8. For these tests we also report on coverage: the percentage of cases where this condition is met. Note that the baselines do not have an estimated precision because they do not use the meta-learning system. However, for completeness, we repeat the tests for them as well, with a threshold on the real precision.

Results

Table 1 shows the results averaged over all the cases in the corpus (with standard deviation in brackets). The left-hand side corresponds to the setting where the system always selects a method. As expected, the Random baseline has the worst performance. The Gold baselines outperform their corresponding meta-learning counterparts, but the differences in precision are not statistically significant between the Gold and the Weighted scenarios (using a t-test with a 1% significance level). Comparing the Weighted and Max scenarios, the former is superior, but this is only statistically significant for the difference in precision values between Weighted50 and Max50. Comparing the results when methods are selected based on a standard F-score calculation versus a precision-favouring calculation ($w = 0.5$ versus $w = 0.75$ in Equation 3), precision is significantly higher for the latter in all testing scenarios, as expected. Although this is at the expense of a reduced F-score, this reduction is not as pronounced as the increase in precision.

The right-hand side of Table 1 shows the results when we restrict the system to select a method only when the estimated precision is ≥ 0.8. We see that the meta-learning scenarios cover a proportion of the requests that is comparable to the Gold baselines (approximately 57%), and that all the results are substantially improved. As expected, all the precision values are high, and also more consistent than before (lower standard deviation). This result is more impressive for the meta-learning scenarios, as their selection between methods is based on *estimated* precision, as opposed to the baselines, whose selections are based on *actual* precision, which is not available in practice. Comparing the Weighted and the Max meta-learning methods, there are no significant differences in F-score, but Weighted outperforms Max on precision (though the difference between Weighted75 and Max75 produces a p-value of 0.035, which is not significant at the 1% level). Comparing $w = 0.5$ versus $w = 0.75$, for both Weighted and Max the increase in precision is larger than the reduction in F-score (all differences are significant at the 1% level).

Summary

We summarize the results as follows.

- The meta-learning system significantly outperforms the random selection baseline, and is competitive with the gold baseline.
- There is an overall preference for using the weighted alternative to estimate performance (i.e., using Equation 2 rather than 1). This alternative is better able to handle the uncertainty that arises when a particular configuration of confidence values results in different performances.
- There is an overall preference for precision when selecting between methods, as the increase in precision is larger than the reduction in F-score. This is largely due to the fact that we have methods that produce partial responses (namely *Sent-Pred* and *Sent-Hybrid*), where a precision-favouring selection approach prefers a response that is more precise over a response that may be more complete, as we saw in Cluster 16 in Figure 2.
- Since the system can estimate performance prior to producing a response, it is able to opt for a non-response rather than risk producing a bad one. The decision of what is a bad response should be made by the organization using the system. With the stringent criterion we have chosen (precision≥ 0.8), the system is able to focus on approximately 57% of the requests with a much improved performance.

Related Research

The problem of corpus-based help-desk response automation has not received much attention to-date. When it has, the proposed solutions have always consisted of single rather than meta-level strategies. This is the case of *eResponder* (Carmel, Shtalhaim, & Soffer 2000) which retrieves a list of request-response pairs and ranks them for the user, similarly to our *Doc-Ret* method; and of the approaches described in (Bickel & Scheffer 2004) and (Berger & Mittal 2000), which implement solutions similar to our *Doc-Pred* and *Sent-Pred* methods respectively. Our work, in contrast, considers several methods as well as a meta-level strategy to combine them.

The combination of strategies has been successfully explored in question-answering research (Chu-Carroll *et al.* 2003; Rotaru & Litman 2005), and recommender systems (Lekakos & Giaglis 2007; Burke 2002). Lekakos and Giaglis divided meta-learning for recommender systems into two major classes, "merging" and "ensemble", each subdivided into the more specific subclasses suggested by Burke (2002). The merging category corresponds to techniques where the individual methods affect each other in different ways (this category encompasses Burke's "feature combination", "cascade", "feature augmentation" and "meta-level" sub-categories). The ensemble category corresponds to techniques where the predictions of the individual methods are combined to produce a final prediction (this category encompasses Burke's "weighted", "switching" and "mixed" sub-categories).

Our system falls into the ensemble category, since it combines the results of the various methods into a single outcome. More specifically, it fits into Burke's "switching" subcategory: selecting a single method on a case-by-case basis. A similar approach is taken by Rotaru and Litman (2005), but their system does not have any learning. Instead it uses a voting approach to select the answer that is provided by the majority of methods. Chu-Carroll *et al.*'s system (2003) belongs to the merging category of approaches, where the output from an individual method can be used as input to a different method (this corresponds to Burke's "cascade" sub-category). They are able to do this because the results of all the methods are comparable. For this reason there is no learning in this system: at each stage of the "cascade of methods", the method that performs best is selected. In contrast to these two systems, our system employs methods that are not comparable, because each is applicable in rather different circumstances. Therefore, we need to learn from experience when to use each method.

Conclusion

We have presented a corpus-based approach for the automation of help-desk responses. In our approach, the content of a suitable response to a request and the method employed to generate this response depend on the features of the request and on past experience with the corpus (composed of request-response email pairs).

We presented a system equipped with a meta-level component for selecting between different response-generation methods. These methods are able to indicate a degree of confidence in addressing a new request, based on previous experience. The selection process is governed by how these degrees of confidence translate to performance. Specifically, by considering the degree of confidence of the different methods when addressing a request, along with their performance in the past, we were able to implement an unsupervised learning scheme. This approach, which avoids the cost and subjectivity of human labeling required for supervised approaches, was shown to be appropriate for our task. Further, the practical settings that place an increased emphasis on precision were demonstrated to be beneficial.

Acknowledgments

This research was supported in part by grant LP0347470 from the Australian Research Council and by an endowment from Hewlett-Packard. The authors thank Hewlett-Packard for the extensive anonymized help-desk data.

References

Berger, A., and Mittal, V. 2000. Query-relevant summarization using FAQs. In *ACL2000 – Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, 294–301.

Bickel, S., and Scheffer, T. 2004. Learning from message pairs for automatic email answering. In *Proceedings of the European Conference on Machine Learning*.

Burke, R. 2002. Hybrid recommender systems. *User Modeling and User-Adapted Interaction* 12(4):331–370.

Carmel, D.; Shtalhaim, M.; and Soffer, A. 2000. eResponder: Electronic question responder. In *CoopIS '02: Proceedings of the 7th International Conference on Cooperative Information Systems*, 150–161.

Chu-Carroll, J.; Czuba, K.; Prager, J. M.; and Ittycheriah, A. 2003. In question answering, two heads are better than one. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 24–31.

Lekakos, G., and Giaglis, G. M. 2007. A hybrid approach for improving predictive accuracy of collaborative filtering algorithms. *User Modeling and User-Adapted Interaction* 17(1), 5–40.

Marom, Y., and Zukerman, I. 2007. A predictive approach to help-desk response generation. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, 1665–1670.

Rotaru, M., and Litman, D. J. 2005. Improving question answering for reading comprehension tests by combining multiple systems. In *Proceedings of the AAAI 2005 Workshop on Question Answering in Restricted Domains*.

Salton, G., and McGill, M. 1983. *An Introduction to Modern Information Retrieval*. McGraw Hill.

Wallace, C. 2005. *Statistical and Inductive Inference by Minimum Message Length*. Springer, Berlin, Germany.

Zukerman, I., and Marom, Y. 2006. A comparative study of information-gathering approaches for answering help-desk email inquiries. In *Proceedings of the 19th Australian Joint Conference on Artificial Intelligence*, 546–556.