A System for Recommending Briefing Drafts from Non-textual Events

Mohit Kumar
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, USA
mohitkum@cs.cmu.edu

Dipanjan Das
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, USA
dipanjan@cs.cmu.edu

Alexander I. Rudnicky
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, USA
air@cs.cmu.edu

ABSTRACT

With the increasing ubiquity of email and web related technologies in supporting and documenting tasks there arises a need to summarize a user's interactions and activities over a period of time. We describe a learning-based system for generating reports based on a mix of text and event data. The report drafting system is part of a learning cognitive assistant, implemented using web technologies. The system incorporates several stages of processing, including aggregation, template-filling and importance ranking. Aggregators and templates were based on a corpus of reports evaluated by human judges. Importance and granularity were learned from this corpus as well. The system can learn to accurately predict the report writing behavior and produces significant performance improvements relative to a non-learning system.

INTRODUCTION

In this paper we describe a system for generating reports that forms a part of a learning cognitive assistant. The assistant aids a human being in handling sudden crisis situations that may arise during a conflict resolution scenario. The purpose of the briefing assistant, as a part of the larger system, is to summarize a variety of events that occur during the span of conflict resolution. The primary difference of this work from previous research on summarization lies in the fact that we attempt to create briefings from real world 'events' rather than solely from natural language text. Most often, summarization techniques from single as well as multiple documents use an extractive paradigm, where no natural language generation methods are used. We arrive at a compromise between purely extractive and abstractive techniques by designing a set of templates that have wide coverage on the types of activities during system operation then customizing these to reflect actual event patterns. This set of templates is determined by performing corpus study of human generated briefings after test sessions, noting the range of briefing items that human beings use in summaries.

We create candidates for a briefing by filling up dynamic fields in templates that take up different values over different sessions. The filling up process relies on a set of aggregators that accumulate user activity on different tasks that the assistant supports. User activity is logged by components of the larger system for consumption of component modules such as the briefing assistant. During template design, we noticed that in our corpus, certain classes of templates which convey the same information have different granularities in terms of quality. The granularity difference creates templates summarizing the same information but in different ways. A ranking model is designed for ranking the tentative set of templates that helps extract the most relevant ones given a session. It is modeled as a consensus-based classifier, where individual classifier models are built for each user in the training set. The prediction scores of each classifier are combined to produce a final rank of each template. The briefing system's recommendations are the four top-ranking templates. As discussed in section , we observe a significant improvement in performance between the system with learning and without learning, evaluated using objective as well as subjective metrics.

The plan of the paper is as follows. We describe relevant work from existing literature in the next section. Then, we provide an overview of the full cognitive assistant featuring the underlying model of the briefing system. A detailed description of its components, including the ranking model is outlined in the following section. All the experiments performed and the results achieved are described next, following which we conclude the current work with a mention of directions of future work.

RELATED WORK

The literature describes various summarization systems that generate short briefings. [21] elaborates the SUMMONS system. The authors of the paper point out that the objective of a briefing system is to convey information to the user that he or she is interested in. [16] describes another briefing system that considers user preference, but does not interact with the user while creating the briefing outline, unlike our system. [14] elaborates yet another briefing system that consumes user's feedback for model tuning and feature discovery and learns a personalized model of a user writing a report. However, it focuses more on personalization and text based extractive briefings.

Focusing on event-based summarization, [7] talks about identification of sub-events in multiple documents to create a summary. The work focuses on dividing an article into events and accumulating perspectives of different documents on the same topic to create a useful summary. [9] mentions the use of event-based features in extractive summarization and investigates the results of their system compared with a baseline feature set corresponding to other state of the art summarization methods. [15, 24, 25] describe similar work based on events occurring in text and techniques of summarizing from single and multiple documents. However, unlike the case at hand, all the work on event based summarization has text as the content on which the techniques are applied.

In spite of there being a colossal amount of work on text summarization, non-textual summarization techniques have not been investigated in abundance. [17] explores summarization from events (e.g. weather, financial and medical knowledge bases) rather than from reduction of free text. The events are selected by analyzing domain dependent semantic patterns, link analysis of different events and statistical analysis. [18] describes another summarization approach that takes non-textual data as input. They describe two systems that generate summaries of basketball games and that of telephone network planning activity, and convey as much information as possible through a short amount of text. Their work relies on complex natural language generation techniques, as opposed to our approach of minimal linguistic involvement in summarizing events. [19] describes another such system that generates briefings in the medical domain.

Our system describes a model that derives final briefing items from a voting scheme between learned models of users, but the approach is distinguishable from collaborative filtering techniques of creating recommendations [20, 12] where usually there is a step involving a profiling process of different users who have a set of items to rate or choose from. Patterns common to several users are searched for, to calculate a prediction for a new user. In contrast, our application creates an environment where the status of different tasks may vary widely, and hence for different users, the available options may differ to large extents. Below, we describe our approach towards a summarization method that aggregates real world events to form candidate briefing templates, learns to rank candidate items and produces a final briefing for the user.

SYSTEM OVERVIEW

Domain Description

The target domain of the briefing application is a cognitive assistant that uses machine learning (ML) to assist a human being to solve a set of tasks whose number might turn out to be very large and the environment might be affected by a sudden crisis. The long-term objective of this work is to develop technology that will

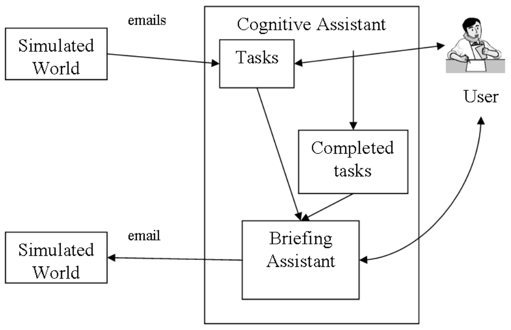


Figure 1. Role of the Briefing Assistant.

be able to function "in the wild", without significant intervention by experts and without the need for expert knowledge on behalf of the end user. The current domain is a conference scheduling scenario, where a number of entities such as sessions, speakers, conference venue, the venue's rooms, different vendors supplying materials to the various sessions etc. co-occur and need to be scheduled by a human being.

The scenario is simulated such that a set of task requests arrive in the form of emails in the user's inbox. A finite amount of time is assigned to the user to complete a part of this set, towards a goal of scheduling the conference while taking into account various constraints and difficulties that arise over the course of a session. As crisis situations arise during the session, the system helps the user solve the problem. A supervisor of the system user (the system user's role can be visualized as a secretary to the former) asks for a report through an email, to brief him about the activities that went on during the session. It is at this juncture that the Briefing Assistant (BA) comes into picture. The function of the BA is to aggregate information about all the completed tasks and the tasks that were ought to be completed, rank the information in an order that the module thinks to be appropriate and suggest to the user briefing bullets that he might further modify or add to and create the final report. Figure 1 gives an overview of the domain and the role of the briefing assistant with respect to the larger world.

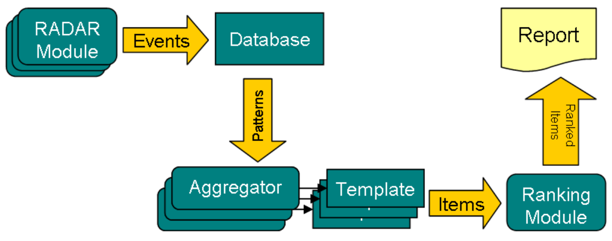


Figure 2. Briefing Assistant Data Flow.

Detailed Overview

To gain a better understanding of the BA's operations, the reader requires a detailed overview of the module's interactions with the other parts of the larger system. As indicated by Figure 1, incoming emails are converted to tasks automatically with minimal user participation by an email classifier [1]. Eight types of tasks can occur in the conference scheduling scenario, and each email is classified into one or more categories, thus creating a "to-do" list for the user handling the system. The eight tasks are defined as follows: *CHANGE-ROOM*: This task corresponds to changing the attributes of a room that belongs to the conference world. Attributes can be the conference capacity of the room, the number of audio-visual equipments present in the room and so on. *CHANGE-SPEAKER*: Similar to the previous one, this task changes the attributes of a speaker of the conference. An attribute can be the availability of the speaker, which may affect the way his sessions fit in to the final conference schedule. *CHANGE-SESSION*: This task changes the attributes of a particular session in the conference. An example attribute of a session is the attendance of the session. *WEB-VIO*: VIO stands for virtual information officer, and these tasks are requests asking for website updates. The conference has a website that contains information about speakers, their papers, their contact information, etc. Often, some information is missing or incorrect and WEB-VIO emails ask for the necessary changes on the website. *WEB-WBE*: These tasks refer to batch updates on the website and provide a link to a file (say .csv) which should be used to make the update. *INFO-REQUEST*: This task type is for generic requests for information: directions, website pointers, answers to simple questions, etc. *MISC-ACTION*: This type of task is used for miscellaneous tasks that do not fit into the other task types and are largely satisfied with vendor orders. For example, if someone asks for a vegetarian meal or a slide projector for a session, it will be classified as a MISC-ACTION task. *BRIEFING*: This is the task that requests a briefing of the ongoing activities in the conference scheduling scenario.

BA interacts with three modules of the larger system. They are described as follows:
1. Space Time Planner (STP): The space time planner is the module that manages and schedules different sessions, rooms and times in an intelligent manner [10], satisfying many constraints. The constraints can be the availability of speakers, unavailability of parts of buildings and so on. It enumerates the sessions that are moved by its scheduler. It also lists the sessions that have not been scheduled in any room.

2. Task Manager (TM): The TM [11] forms the backbone of the system and provides the user with a web based console [8] that contains all the proposed tasks that need to be performed. The CHANGE-ROOM, CHANGE-SESSION and CHANGE-SPEAKER tasks are such that the form provided in the console helps the user complete it. All changes are systematically logged by the Task Manager. The WEB-VIO task provides a link that activates the interface of the VIO module. The web updates are written back to the TM and it is logged similarly.

3. Natural Language Processing (NLP): The NLP module analyzes the responses to vendor orders placed by the user. The vendor order requests come in the form of MISC-ACTION emails. The orders can be either for food, flowers, audio-visual equipments or security. They are placed by the user using an external web-portal. The corresponding vendors send responses or confirmations that are parsed by the NLP-module. These parses form an important input for the briefing assistant to report.

A crisis in the conference scheduling world can have many definitions. The present world defines it in the form of a sudden unavailability of a part of a building where many sessions were initially scheduled. The unavailability spans over a period of a few days or the entire duration of the conference. The user is notified of the crisis, and with the help of the system, he reschedules the sessions with optimality.

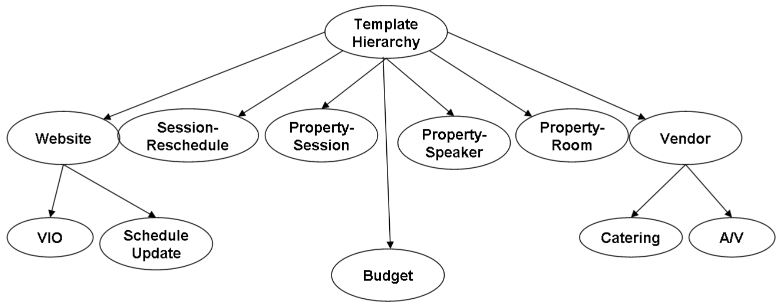
The Briefing Assistant Model

We modeled the problem of Briefing generation in the current domain as non-textual event-based summarization. The events are the task creation and task completion actions logged by the various specialists. To circumvent the problem of natural language generation we designed a set of templates, based on the actual briefings written by users. More details about the template design process are given in the following subsection. Based on this set of templates we identified the patterns that needed to be extracted from the event logs in order to populate the templates.

The overall data flow of the BA is shown in Figure 2. The various specialist modules generate task related events that are logged in a database. From this database the aggregators extract the important patterns. These patterns are used to populate the set of templates that form the candidate briefing items. The candidate briefing items are then ranked by the Ranking module and evaluated against the user's selection.

Template Design

Thirty-three subjects were asked to use the prior version of the system under various configurations and to write Briefings [13]. They were given broad guidelines, for example that the Briefing should have 4 bullet points corresponding to the most important activities that should be reported. The briefings were then evaluated by a panel of three judges. The judges assigned scores (0-4) to each of the bullets based on the coverage of the crisis, clarity and conciseness, accuracy and the correct level of granularity. For designing the templates, we selected 81 briefing items from 26 users that had an average judge's score greater than an empirically chosen threshold (1.67). We were lenient in setting the threshold as we wanted to capture templates with wide coverage. Additionally, we added a few templates that we believed should be present in the set corresponding to data that is difficult to obtain (for example the



(a)

```
<Template class="VIO" example="Most of the ARDRA
 website updates requested by the conference attendees
 related to email addresses, phone numbers and paper
 titles have been done.">
      <qualitative num> of the ARDRA website updates
      related to <updated website fields>+ have been
      done.
 </Template>
```

(b)

Figure 3. (a) The category tree showing the information types that we expect in a briefing (b) An example template in XML format. The class attribute represents the template category as depicted in Figure 3. The example attribute enumerates a briefing item obtained by filling up this template type. The template slots are <qualitative_num> and <updated website fields>+ which are instantiated as most and email addresses, names, phone numbers, paper titles respectively in the example briefing item.

number of sessions moved) and negative templates (the user may not realize that he hasn't completed certain tasks). So we have a set of 29 templates that correspond to nine broad categories. These nine categories do not correspond to the task types rather they are the information types that we expect to see in a briefing. Figure 3 (a) shows the tree corresponding the nine categories. Figure 3 (b) shows an example template.

The templates categories are defined as follows:
1. Website updates: (a) VIO: As mentioned earlier, VIO stands for Virtual Information Officer and it maintains a website that contains details about speakers, their papers, organizations and so forth. The VIO-related tasks that the user completes include (for example) changing an incorrect paper title. This template corresponds to such updates and summarizes the number of such updates done during a session. (b) Schedule Update: During the scheduling process of the conference, the user moves sessions around with the help of the system. They might decide to publish the schedule on the website after significant changes. This template attempts to capture this website publish.

2. Session-Reschedule: This template captures the number of sessions that have been rescheduled because of the crisis scenario. The crisis corresponds to the unavailability of rooms over a span of time, and the user is forced to reschedule the sessions scheduled in these rooms.

3. Property-Session: Whenever an email arrives to inform the user of an increase or a decrease of attendance of important sessions, the user updates the session properties. This template covers such updates.

4. Property-Speaker: Speakers send in emails asking to change their availability during the span of the conference, as and when their personal schedules change. This availability might affect the schedule of different sessions too. The current template tries to aggregate the changes in the speaker availability.

5. Property-Room: When a crisis hits the conference scheduling process, new rooms might become available to compensate for the loss of previously scheduled rooms. Such updates are captured by this template.

6. Vendor Orders: (a) Catering Vendors: As part of the miscellaneous tasks, there are requests for special meals for particular sessions from the conference attendees. The user uses different vendor quotes to confirm orders to catering vendors. This template summarizes such orders for different sessions and different types of requests. (b) A/V Vendors: Similar to the previous template, this template creates a summary of confirmed vendor orders that deal with audio visual equipments like laptops, slide projectors etc for presentations.

7. Budget Related: This template captures the total approximate budget of the conference. The information has two components: rental cost of the rooms and vendor order cost.

COMPONENT DESCRIPTION

Aggregators

The aggregators populate the set of templates that are the candidate items for a briefing. These components are a set of methods that mine the logs and databases and summarize the different user interactions that occur during a session. The aggregators are in the following four main categories:

TM-Based: These aggregators accumulate information about events that occur as a result of the user's interaction with the system console webpage and VIO. The Property-Session, Property-Speaker, Property-Room and VIO templates are the ones that use these aggregators.

STP-Based: The STP-Based aggregators infer details of the sessions that have been rescheduled. These aggregators serve the Session-Reschedule category of template. The STP also indicates whether the session has been published on the website, and this indication serves as input to the Schedule-Update template.

NLP-Based: The NLP pipeline in the system provides an API for accessing the annotations that it performs on emails. The vendor-order confirmation emails are parsed by the pipeline and entities like number of special meals, the cost of the meals etc are annotated. The interface is read by these aggregators to extract information about the Vendor-related templates.

Ranking Model and classifiers

The ranking module ranks the set of candidate templates to help select the most relevant ones for inclusion in the final briefing. The ranking system is modeled as a consensus-based classifier, where individual classifier models are built for each user in the training set. The prediction scores of each classifier are combined to produce a final rank of each template. BA's recommendations are the four top-ranking templates.

We used the Minorthird package [3] for developing the system. We experimented with eleven different learning schemes for the individual models. The schemes were Naive Bayes, Voted Perceptron, Support Vector Machines, Ranking Perceptron [4], K Nearest Neighbor, Decision Tree, AdaBoost [22], Passive Aggressive learner [6], Maximum Entropy learner, Balanced Winnow [2] and Boosted Ranking learner [5].

Features

The features used in the system can be classified as static or dynamic. The static features are the properties of the templates irrespective of the user's activity whereas the dynamic features are based on the actual events that took place. The static features are:-

1. template id - Feature corresponding to the unique template number. Instead of treating this feature as an 'enumeration' we converted it to a set of 29 boolean features corresponding to each of the templates.
2. template class -Feature corresponding to the information category type. Similar to template id, we converted this feature from an enumeration of 9 categories corresponding to the Figure 3 to a set of 9 boolean features.
3. negativity - Boolean feature indicating whether the template indicates that tasks were incomplete.
4. abstraction - Boolean feature indicating if the template contains abstraction over some underlying entities for example usage of 'personal information' instead of 'name, address, phone number etc'.
5. granularity - Feature corresponding to the granularity of the template. For example if it's a detailed template or a succinct one. We designed templates with 5 levels of granularity and hence we converted this feature to a set of 5 boolean features.

The dynamic features are:-

1. qualitative feature - Feature corresponding to the qualitative value of the template; it is valid only for certain templates. The feature depends on the user's activity for example if the user completed 4 out of the proposed 10 WEB-VIO tasks then the qualitative feature would be 'some'. We considered four levels of qualitative values (few, some, most and all) and thus have a set of 4 boolean features.
2. individual task coverage and time taken- Numeric features corresponding to the percentage of tasks that are 'covered' by the template and the percentage of time spent on the tasks related to the template. For example, if there were 15 WEB-VIO tasks that were done and the total number of tasks done was 75, then the value of this feature for the related templates would be $15/75 = 0.2$. This feature captures the intuition about the relative amount of work that the user did associated with this template, the higher the amount of work done, the more likely for that template to be of greater importance.
3. global task coverage and time taken - Numeric features corresponding to the percentage of tasks completed and percentage of time spent on the tasks relative to other users. For example, if the global task completion average for WEB-VIO tasks is 15 out of the possible 30 tasks and the user has completed 30 then the value of this feature would be $30/15 = 2$. The intuition for this feature is that if the user has spent more (or less) time than the average user he may want to report (or not report) about it. The higher the feature value, more likely for that template to be of greater importance and vice versa.

Feature Selection: We used the Information Gain (IG) metric for feature selection. We experimented with seven different cut-off values $All, 20, 15, 10, 7, 5, 4$ for the total number of selected features. The experiments are detailed in the next section.

EXPERIMENTS AND RESULTS

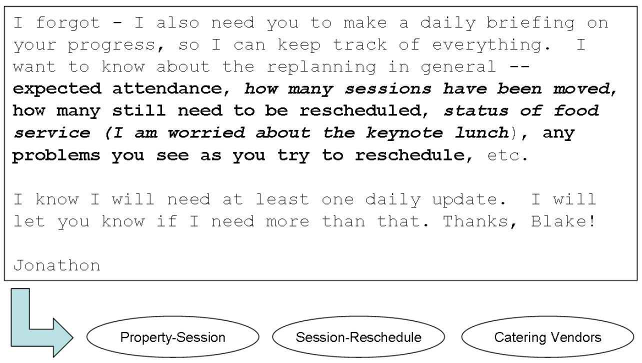
Experimental Setup

The briefing system was evaluated as a part of the larger cognitive assistant. [23] describes the test methodology for the entire system in detail. The premise of the evaluation was 'Learning in the Wild' (LITW) which implies that the system should learn through user interaction and should not depend on knowledge engineering. In our report drafting system, the parameters that were chosen based on the training user data i.e 'In the Wild' were the learning scheme in the ranking module for individual models and the IG-based Feature selection threshold. Although it is semantically the same as parameter selection in the machine learning (ML) framework, it is different in the definition of the parameters where the parameters of the system are characteristics such as the learning scheme and feature selection threshold instead of the parameters of the learning algorithms themselves (eg. Support Vectors in SVM classifiers). This paradigm gives the system the adaptability to operate in the wild where the data characteristics are not well defined.

In order to show the specific influence of learning on overall performance, there are two Radar conditions - one with learning (PlusL) and one without (MinusL). In the context of the evaluation, learning is only LITW. Learning acquired through knowledge engineering (template design etc) or through brute force encoding (e.g. email stimulus processing module mentioned in the following subsection etc) would be available in both the PlusL and MinusL Radar conditions.

Email Stimulus

In the simulated crisis for the conference planning domain, the briefing was triggered through a mail that explicitly mentioned some specific information request. Thus to customize the briefing report to the particular email, a natural language processing module extracted the relevant categories of information requested. The details of the module are beyond the scope of the current paper as it is external to our system; implemented by other groups in the project, taking into account our template categories described in the template design section. Figure 4 shows a sample briefing email stimulus and the corresponding template categories. The mapping from email text to category is as follows: "expected attendance" - Property-Session; "how many sessions have been rescheduled", "how many still need to be rescheduled", "any problems you see as you try to reschedule" - Session-Reschedule; "status of food service (I am worried about the keynote lunch)" - Catering Vendors. This module had no learning component.



I forgot - I also need you to make a daily briefing on your progress, so I can keep track of everything. I want to know about the replanning in general -- **expected attendance, *how many sessions have been moved*, how many still need to be rescheduled, *status of food service (I am worried about the keynote lunch*), any problems you see as you try to reschedule,** etc.

I know I will need at least one daily update. I will let you know if I need more than that. Thanks, Blake!

Jonathon

Figure 4. Template categories corresponding to the Briefing request email.

Training

Eleven expert[1] users were asked to train the system by using the RADAR system and generate the end of session briefing using the BA GUI. BA is run in the MinusL mode i.e. all the templates are populated by the aggregators but it makes random recommendations to the user. The expert user is expected to select the best possible four templates. The training data consists of the features extracted from the activity log of the user for the session and the labeled templates. The stimulus message for the training users did not contain any specific information request so that the importance rankings of the templates are without any bias.

[1]Members of the project from other groups who were aware of the scenario and various system functionalities but not the ML methods

Test

Subjects were recruited to use the system in MinusL and PlusL condition, although they were not aware of the condition of the system and they were not involved with the RADAR project. The details of the subject recruitment criteria are mentioned in [23]. We had 42 test runs in MinusL condition and 32 in PlusL condition. Out of these runs, 19 subjects in MinusL and 30 subjects in PlusL wrote a briefing using the BA. We report the evaluation scores for this set of users who used the BA for the briefing.

Evaluation

The base performance metric is Recall, defined in terms of the briefing templates recommended by the system compared to the templates ultimately selected by the user. We justify this by noting that Recall can be directly linked to the expected time savings for the eventual users of a prospective summarization system based on the ideas developed in this study. We calculate two variants of the Recall: *Category-based* - calculated by matching the categories of the BA recommended templates and user selected ones ignoring the granularity and *Template-based* - calculated by matching the exact templates. The intuition for the first metric is to find out the most important category of information request and the latter is the exact template with the correct granularity of information.

We also did subjective human evaluation of the final briefings by a panel of three judges. As mentioned in the 'Template Design' section, the judges assigned scores (0-4) to each of the bullets based on the coverage of the crisis, clarity and conciseness, accuracy and the correct level of granularity. They were advised about certain briefing-specific characteristics (e.g. negative bullet items are useful and hence should be rated favorably). They were also requested to include a global assessment of the report quality, a measure to evaluate the coverage of the requests in the briefing stimulus email message.

Experiment

The automatic evaluation metric used for the trained system configuration is the *Template-based* recall measure. To obtain the final system configuration, we automatically evaluate the system under the various combinations of parameter settings with eleven different learning schemes and seven different feature selection threshold (as mentioned in previous sections). Thus there are a total of 77 different configurations that are tested. For each configuration, we do a eleven-fold cross-validation between the 11 training users i.e. we leave one user as the test user and consider the remaining ten users as training users. We average the performance across the 11 test cases and obtain the final score for the configuration. We choose the configuration with the highest score as the final trained system configuration. We would break a tie between two system configurations by choosing the system with the simpler learning scheme[2] otherwise choosing the smaller feature threshold.

The final system configuration obtained for our current test is a learning scheme of Balanced Winnow [2] and a feature threshold of Top 7 features.

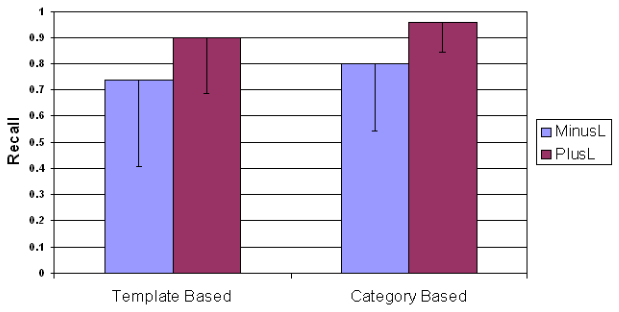


Figure 5. Recall values for MinusL and PlusL conditions.

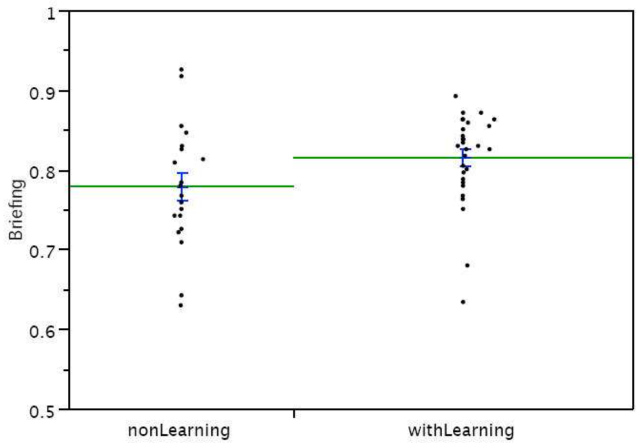


Figure 6. Briefing scores from the judges' panel for MinusL (nonLearning) and PlusL (withLearning) conditions.

Results

Figure 5 shows the Recall values for the MinusL and PlusL condition. The learning delta i.e. the difference between the recall values of MinusL and PlusL is 22% for *Template-based* recall and 19% for *Category-based* recall. These differences are significant at $p < 0.01$. The statistical significance for the *Template-based* metric, which was the metric used for selecting system parameters during the training phase, shows that we are successfully able to Learn in the Wild. Since the email stimulus processing module extracts the briefing categories from the email the *Category-based* and *Template-based* recall is expected to be high for the baseline MinusL case. In our test, the email stimuli had 3 category requests and so the *Category-based* recall of 0.8 and *Template-based* recall of 0.74 in MinusL is not a surprise.

[2]We defined our own complexity rating for each learning scheme

Feature	Freq	Feature	Freq
template_3b	8	template_4a	4
template_class_website	5	qualitative_num_all	4
template_8b	5	granularity_so_so	4
template_8a	5	template_9a	3
granularity_num_detailed	5	global_overall_timing	3

Table 1. Most frequently selected features across eleven training users with Top 7 features selected

Figure 6 shows the Judges' panel scores for the briefings for MinusL and PlusL condition. The learning delta in this case is 5% which is statistically significant with $p < 0.05$. The statistical significance of the learning delta validates that the briefings generated during PlusL conditions are better than MinusL condition. Also there is a positive correlation between the objective recall measures and the subjective human scores, although not statistically significant.

Feature analysis

Overall the system based on the Top 7 features performed the best across users. Table 1 shows the ten most frequently selected features across users for this system. There are four dynamic features in the set which indicates that the learning model is capturing the user's world state and the recommendations are related to the underlying events. This validates our claim of generating briefing reports from non-textual events.

CONCLUSION

Our experiments show that consensus-based model generalizes well across users and produces significant recall. We should note that our approach is not meant to discover general attributes of good reports; rather it provides a framework within which to rapidly learn the characteristics of good reports within a particular domain. As such, these custom attributes are more likely to lead to quality reports. In conjunction with this, we also note that a consensus-based approach yields the best performance, implying that it should be possible to apply data-driven techniques to learn consistent models in this domain.

FUTURE WORK

We have described a system that creates a briefing from events occurring in a learning cognitive assistant. However, the summarization process focuses on ranking different candidate items and selecting the most important ones from the set. The challenge of natural language generation in the summary creation process is obviated by adopting a template based approach rather than generating sentences from event data. An interesting direction of future work is the elimination of fixed templates that get populated, and the induction of templates from significant patterns in event data. The process of summary creation that we describe in this paper can be inverted, if instead of aggregating data to feed the fixed set of templates, we search for data patterns and create new templates summarizing them.

Learning templates from natural language text that constitute human generated briefings without any human intervention is another attractive avenue of future research. Instead of having a fixed set of templates, a variety of templates can be learnt from example briefings automatically. A generic set of aggregators can be designed to capture many kinds of information that can arise in the subject scenario, in turn catering to the learnt templates.

Acknowledgements: We would like to express our gratitude to the entire RADAR project team for making this work possible. This work was supported by DARPA grant NBCHD030010. The content of the information in this publication does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred.

REFERENCES

1. P. N. Bennett and J. Carbonell. Detecting action-items in e-mail. In *Proceedings of SIGIR*, New York, NY, USA, 2005.
2. V. Carvalho, W. Cohen, and A. Blum. Improving winnow for nlp tasks: Voting schemes, regret minimization and online feature selection. In *CMU Technical Report*, 2006.
3. W. W. Cohen. Minorthird: Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data. http://minorthird.sourceforge.net, 2004.
4. M. Collins. Ranking algorithms for named-entity extraction: boosting and the voted perceptron. In *Proceedings of ACL*, Morristown, NJ, USA, 2001.
5. M. Collins and T. Koo. Discriminative reranking for natural language parsing. *Computational Linguistics*, 2005.
6. K. Crammer, O. Dekel, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. In *NIPS*, 2003.
7. N. Daniel, D. Radev, and T. Allison. Sub-event based multi-document summarization. In *Proceedings of HLT-NAACL*, 2003.
8. A. Faulring and B. A. Myers. Availability bars for calendar scheduling. In *Proceedings of CHI*, 2006.
9. E. Filatova and V. Hatzivassiloglou. Event-based extractive summarization. In S. S. Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, July 2004.
10. E. Fink, P. M. Jennings, U. Bardak, J. Oh, S. F. Smith, and J. G. Carbonell. Scheduling with uncertain resources: Search for a near-optimal solution. In *IEEE International Conference on Systems, Man, and Cybernetics*, 2006.
11. D. Garlan and B. Schmerl. The radar architecture for personal cognitive assistance. *International Journal of Software Engineering and Knowledge Engineering*, April 2007.
12. T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1), 2004.
13. M. Kumar, D. Das, and A. I. Rudnicky. Summarizing non-textual events with a 'briefing' focus. In *RIAO*, 2007.
14. M. Kumar, N. Garera, and A. I. Rudnicky. Learning from the report-writing behavior of individuals. In *IJCAI*, 2007.
15. W. Li, M. Wu, Q. Lu, W. Xu, and C. Yuan. Extractive summarization using inter- and intra-event relevance. In *Proceedings of ACL*, 2006.
16. I. Mani, K. Concepcion, and L. V. Guilder. Using summarization for automatic briefing generation. In *NAACL-ANLP 2000 Workshop on Automatic summarization*, 2000.
17. M. T. Maybury. Generating summaries from event data. *Inf. Process. Manage.*, 31(5), 1995.
18. K. McKeown, J. Robin, and K. Kukich. Generating concise natural language summaries. *Inf. Process. Manage.*, 31(5), 1995.
19. K. R. McKeown, S. Pan, J. Shaw, D. A. Jordan, and B. A. Allen. Language generation for multimedia healthcare briefings. In *Proceedings of ANLP*, 1997.
20. P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Eighteenth national conference on Artificial intelligence*, 2002.
21. D. R. Radev and K. R. McKeown. Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, 24(3), 1998.
22. R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3), 1999.
23. A. Steinfeld, R. Bennett, K. Cunningham, M. Lahut, P. Quinones, D. Wexler, D. Siewiorek, P. Cohen, J. Fitzgerald, O. Hansson, J. Hayes, M. Pool, and M. Drummond. The radar test methodology: Evaluating a multi-task machine learning system with humans in the loop. In *CMU Technical Report CMU-CS-06-125*, 2006.
24. M. Wu. Investigations on event-based summarization. In *ACL*, 2006.
25. W. Xu, W. Li, M. Wu, W. Li, and C. Yuan. Deriving event relevance from the ontology constructed with formal concept analysis. In *CICLing*, 2006.