Non-textual Event Summarization by Applying Machine Learning to Template-based Language Generation

Mohit Kumar, Dipanjan Das, Sachin Agarwal, Alexander I. Rudnicky

CMU-LTI-09-012

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213
www.lti.cs.cmu.edu

June 8th, 2009

Non-textual Event Summarization by Applying Machine Learning to Template-based Language Generation

Mohit Kumar and **Dipanjan Das** and **Sachin Agarwal** and **Alexander I. Rudnicky**
Language Technologies Institute
Carnegie Mellon University, Pittsburgh, USA
mohitkum,dipanjan,sachina,air@cs.cmu.edu

Abstract

We describe a learning-based system that creates draft reports based on observation of people preparing such reports in a target domain (conference replanning). The reports (or briefings) are based on a mix of text and event data. The latter consist of task creation and completion actions, collected from a wide variety of sources within the target environment. The report drafting system is part of a larger learning-based cognitive assistant system that improves the quality of its assistance based on an opportunity to learn from observation. The briefing system learns to rank order a set of natural language templates whose slots are filled by aggregators that observe the event stream in the system while people perform the conference replanning task. The system can learn to accurately predict the briefing assembly behavior and shows significant performance improvements relative to a non-learning system. The work described in this paper demonstrates that it's possible to create meaningful descriptions of activity in a human information processing task based on event streams and to automatically learn to assemble useful reports from this information.

1 Introduction

In this paper we describe a system for recommending items for a briefing following a session with a crisis management system. The briefing system is learning-based, in that it initially observes how one set of users creates such briefings then tries to generate draft reports for subsequent users. This system, the Briefing Assistant, is one of a set of learning-based cognitive assistants each of which observes users and tries to learn to assist the users in performing their tasks faster and more accurately.

The difference between this work from most previous efforts, primarily based on text-extraction approaches is our emphasis on learning to summarize event patterns. This work also differs in its emphasis on learning from report preparation behavior in context.

Report generation from non-textual sources has been previously explored in the Natural Language Generation (NLG) community in a variety of domains, based on, for example, a database of events. However, a purely generative approach is not suitable in our circumstances, as we want to summarize a variety of tasks that the user is performing and present a summary tailored to an intended audience (in particular, reports are generated in response to a specific request). Thus we approach the problem by applying learning techniques combined with a template-based generation system to instantiate the briefing-worthy report items. The task of instantiating the briefing-worthy items is similar to the task of Content Selection (Duboue, 2004) in the Generation pipeline however our approach is preferable as it requires minimal linguistic involvement. Our choice of a template-based generative system was motivated by recent discussions in the NLG community (van Deemter et al., 2005) about the practicality and effectiveness of this approach.

The set of templates used in the current instantiation of the Briefing Assistant was derived from a corpus of human-generated briefings collected in a pre-

vious experiment using the same crisis management system. The set of templates was designed to cover the range of items that users in that experiment chose to include in their reports. In the Briefing Assistant, report item instantiation is a two-step process: A set of aggregators scan the event record generated during a session for particular patterns; these patterns are distilled in a manner appropriate for a class of reportable information, then the slots of the suitable templates are filled to convey the information textually. We found that information can be conveyed at different levels of granularity (for example, qualitatively or quantitatively). The different levels of granularity provide templates summarizing the same information but in different ways. Examples 1 and 2 in Table 1 illustrate examples of templates with different granularity belonging to the same category, where Example 2 is more detailed than Example 1.

The appropriate choice of granularity for a particular session becomes something that the system can learn about. Modeling is performed using 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 for each template. The four top-ranking templates then form the briefing system's recommendations. As discussed in section 4, we observe a significant improvement in report quality for the system with learning compared to one 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 the system description featuring the domain overview, interaction of briefing assistant with the larger system, the underlying model of the briefing system and a description of its components, including the ranking model and features. Experiments and the results obtained are described next, following which we provide some conclusions.

2 Related Work

The literature describes various summarization systems that generate short briefings. (Radev and McKeown, 1998) point out that the objective of a good briefing is to communicate the information that is interesting to the audience. (Mani et al., 2000) described another briefing system that considers user preference, but does not interact with the user while creating the briefing outline, unlike our system. (Kumar et al., 2007b) elaborated 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 on personalization and extractive summaries derived from text.

For event-based summarization, (Daniel et al., 2003) described identification of sub-events in multiple documents and accumulating perspectives of different documents on the same topic to create a useful summary. (Filatova and Hatzivassiloglou, 2004) mentioned the use of event-based features in extractive summarization and compared it with a baseline feature set corresponding to other state of the art summarization methods. (Li et al., 2006; Wu, 2006; Xu et al., 2006) describe similar work based on events occurring in text and techniques for summarizing from single and multiple documents. However, unlike the case at hand, all the work on event-based summarization used text as source material.

In contrast, (Maybury, 1995) explored summarization from events (e.g. weather, financial and medical knowledge bases) rather than free text. The events are selected by analyzing domain dependent semantic patterns, link analysis of different events and statistical analysis. (Saint-Paul et al., 2005) detailed a general purpose database summarization system called SaintEtiq that provides multi-resolution summaries of structured data based on a conceptual clustering algorithm. Their motivation is to organize patterns in sensible groups to get an overview of the entire database. In our case we are interested in identifying only briefing-worthy events.

Non-textual summarization has also been explored in the Natural Language Generation (NLG) community within the broad task of generating reports based on database of events in specific domains such as medical (Portet et al., 2009), weather (Belz, 2007), sports (Oh and Shrobe, 2009) etc. However, in our case we want to summarize a variety of tasks that the user is performing and present a summary to an intended audience (as defined by a report request).

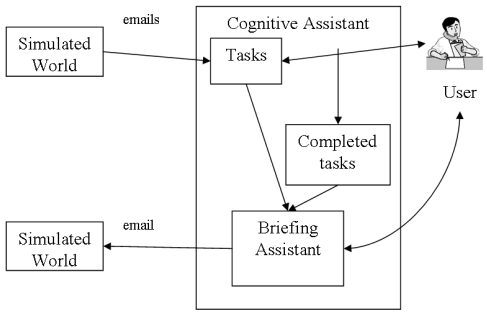

Figure 1: Role of the Briefing Assistant.

Recent advances in NLG research use statistical approaches at various stages of processing in the generation pipeline like content selection (Duboue and McKeown, 2003; Barzilay and Lee, 2004), probabilistic generation rules (Belz, 2007). Our proposed approach differs from these in that we apply machine learning after generation of all the templates, as a post-processing step, to rank them for inclusion in the final briefing. We could have used a general purpose template-based generation framework like TG/2 (Busemann, 2005), but since the number of templates and their corresponding aggregators is limited, we chose an approach based on string manipulation.

We found in our work that an approach based on modeling individual users and then combining the outputs of such models using a voting scheme gives the best results, although our approach is distinguishable from collaborative filtering techniques used for driving recommendation systems (Melville et al., 2002; Hofmann, 2004). We believe this is due to the fact that the individual sessions from which ranking models are learned, although they range over the same collection of component tasks, can lead to very different (human-generated) reports. That is, the particular history of a session will affect what is considered to be briefing-worthy.

3 System Overview

3.1 Domain Description

The Briefing Assistant application is one of several cognitive assistants that are supposed to help a human being operate in a crisis management domain. All assistants are learning-based, in that they are given the opportunity to observe one set of humans work in the domain. They are then given to a new set of users who also perform the same tasks. The expectation is that the new group will do better by benefiting from the cognitive assistant's previous experiences. The long-term objective of this work is to develop technologies that will be able to function "in the wild" by learning from experience. Specifically the goal is to lessen dependence on expert developers and to minimize the need for expert knowledge (about the cognitive assistant) required of the end user. The domain used in this research is conference replanning and requires the user (and the assistant) to deal with a variety of entities such as sessions, speakers, conference venues, a venue's different rooms and resources, various vendors supplying different materials for the conference and so on. A set of specialized applications are provided (for example for scheduling and for briefing preparation) but the bulk of the activity is driven by a stream of email (describing problems, introducing constraints and making requests).

A crisis in the conference replanning world can have many definitions. The present scenario involves the sudden unavailability of a part of a building in which many of the 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, has to (optimally) reschedule the sessions and rearrange resources, all within a limited time span.

The scenario is simulated such that a set of task requests arrive as 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 by making suggestions of various kinds. Specific to the Briefing Assistant (BA) agent, a supervisor (the system user's role can be visualized as a secretary to the former) sends an email asking for a report that will brief the activities that went on during the session. The function of BA is to aggregate information about all the completed tasks and the tasks that were ought to be completed, rank the information

and suggest to the user briefing items that he might further modify or add additional items to create the final report. Figure 1 gives an overview of the domain and the role of BA with respect to the larger system.

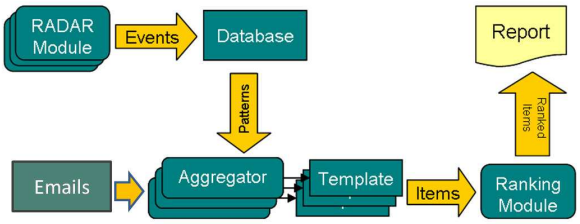

Figure 2: Briefing Assistant Data Flow.

3.2 Interaction with Rest of the Cognitive Assistant system

To gain a better understanding of the BA's operations, we provide a brief 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 (Bennett and Carbonell, 2005). There are eight types of tasks that are associated with email in our scenario and each email is classified into one or more task types, thus creating a "to-do" list for the user. In addition there is an ongoing task of optimizing the schedule of the conference to take into account all resource constraints. This task is not triggered by an email but is explained to the user as part of the introduction to the domain. The eight tasks associated with emails 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 to access relevant information regarding the various tasks. They are described as follows:

1. Space Time Planner (STP): The STP module (Fink et al., 2006) assists in the task of optimizing the schedule of the conference by satisfying many constraints. The constraints can be the availability of speakers, unavailability of parts of buildings and so on. BA uses the information regarding the sessions that are moved by its scheduler and also the sessions that have not been scheduled in any room.

2. Task Manager (TM): The TM (Garlan and Schmerl, 2007) forms the backbone of the larger system and provides the user with a web based console (Faulring and Myers, 2006) that contains all the proposed tasks that need to be performed. Change-Room, Change-Session and Change-Speaker tasks are accomplished by filling up forms provided in the console and the TM directly logs the related task events. The Web-Vio task is handled by a separate VIO module (Zimmerman et al., 2007) and the updates are reported back to the TM. Thus BA accesses the information regarding the above four tasks from TM.

3. Natural Language Processing (NLP): The NLP module analyzes the vendor orders placed by the

user corresponding to the Misc-Action tasks. The orders are places using an external web-portal. The orders can be either for food, flowers, audio-visual equipments or security. The corresponding vendors send responses or confirmations that are parsed by the NLP-module. BA uses this parsed information to report about the vendor orders.

We do not access any information regarding the Info-Request and Web-Wbe task as there are no templates corresponding to these tasks in our domain i.e. information regarding these tasks is not briefing worthy.

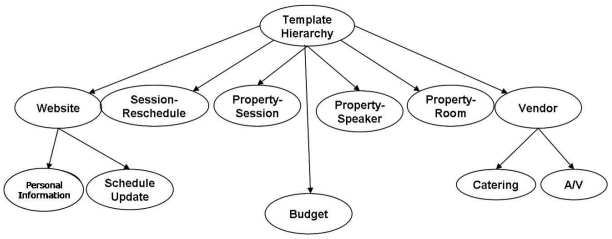

Figure 3: The category tree showing the information types that we expect in a briefing

3.3 The Briefing Assistant Model

We treat the task 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 various cognitive assistants in the system (so-called specialists). As part of the design phase for the template-based generation component, we identified a set of templates, based on the actual briefings written by users in a separate experiment. Ideally, we would like to adopt a corpus-based approach to automatically extract the templates in the domain, like (Kumar et al., 2008), but since the sample briefings available to us were very few, the application of such corpus-based techniques was not necessary. Additional details about the template design process are provided 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. A ranking model was also designed for ordering instantiations of this set of templates and to recommend the top 4 most relevant ones for a given session.

The overall data flow for BA during a session (runtime) is shown in Figure 2. The various specialist modules generate task related events that are logged in a database. The aggregators operate over this database and emails to extract relevant patterns. These patterns in turn are used to populate templates which constitute candidate briefing items. The candidate briefing items are then ordered by the ranking module and presented to the user.

3.4 Template Design

Thirty-three subjects were asked to use the prior version of the system under various configurations and to write Briefings (Kumar et al., 2007a). 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 number of sessions moved) and negative templates (the user may not realize that he hasn't completed certain tasks). This yielded a set of 29 templates corresponding to nine broad categories. Note that these categories do not represent task types rather they are information types found in a briefing.[1] Figure 3 shows the tree corresponding to the nine categories.

The templates categories are defined as follows:

1. Website updates: (a) Personal Information: As mentioned earlier, Virtual Information Officer (VIO) maintains a website that contains details about speakers, their papers, organizations and so forth. The VIO-related tasks 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.

[1] Some of the tasks were not considered briefing worthy whereas some of the tasks have multiple categories associated with them.

(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 captures the information that the website has been updated with the schedule.

2. Session-Reschedule: This template aggregates the number of sessions that have been rescheduled because of the crisis scenario. As mentioned before, the crisis corresponds to the unavailability of rooms over a span of time, and the user, with the help of STP module, reschedules 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 email their updated availability and request rescheduling of their sessions accordingly. This template summarizes the changes in the speakers availability.

5. Property-Room: During the crisis, new rooms might become available to compensate for the loss of previously scheduled rooms. The template reflects these changes in room properties.

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. 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. Rental cost is calculated by obtaining the room usage details from the schedule and getting the rental cost of the room from the room properties. Vendor order cost is extracted by parsing the confirmed orders by the NLP module.

3.5 Aggregators

The system uses five kinds of aggregators: 'Numerical' (labelled:Num), 'Enumerators' (labelled:Enum), 'Set' (labelled:Set), 'Negators' (labelled:Neg) and 'Selectors' (labelled:Sel). Examples are in Table 1. Aggregators use two sources of information: the event data generated by the other cognitive assistant modules, and emails. 'Numerical' aggregators output numbers as illustrated in Examples 1 and 2 in the Table 1 (i.e. dollar amount corresponding to variables Total_Cost, Vendor_Cost and Room_Cost). 'Enumerator' aggregators output values from an enumeration specific to the template, based on domain logic; Example 3 shows the variable Qualitative_Num with value "All", other possible values could be "Some", "Few" and "Most of the". The aggregator counts the number of website updates and maps it to one of the possible values based on template logic. The 'Set' aggregator outputs a subset of values from a set of allowed values as illustrated by Example 3 where the aggregator has assigned the variable Updated_Fields the values "email addresses", "name" and "phone numbers" from a larger set of fields. 'Negators' use template specific logic to trigger negative templates as illustrated by Example 4 where the aggregator checks for zero updates, triggering this template. 'Selector' aggregators select particular information directly filling templates as illustrated by Example 5 where the aggregator selects the variable Session_Names as "Banquet" and "Keynote Lunch" and the Meal_Type as "kosher" and "vegan". The Selector aggregator is different from the Set aggregator as it doesn't have complete information about the possible set of values.

Examples 1 and 2 from Table 1 illustrate the difference in granularity of the templates corresponding to the same category 'Budget'. The variable Total_Cost is obtained by adding the constituent variables Vendor_Cost and Room_Cost. In Example 1, only the combined information about the Total_Cost is presented whereas Example 2 is more detailed and mentions the Total_Cost along with its constituents.

3.6 Ranking Model and classifiers

The ranking module orders candidate templates so that the four most relevant ones appear in the

Num	Aggregator Type	Template
1	Num	<Template class="Budget"> The total expenditure for the conference inclusive of vendor orders and room costs is: <Num:Total_Cost> $400,000 </Num:Total_Cost>. </Template>
2	Num	<Template class="Budget"> The total expenditure for the conference is: <Num:Total_Cost> $400,000 </Num:Total_Cost>, where vendor orders cost <Num:Vendor_Cost> $100,000 </Num:Vendor_Cost> and rooms cost <Num:Room_Cost> $300,000 </Num:Room_Cost>. </Template>
3	Enum,Set	<Template class="Personal Information"><Enum:Qualitative_Num> All </Enum:Qualitative_Num> ARDRA website updates requested by the conference attendees related to <Set:Updated_Fields> email addresses, names and phone numbers </Set:Updated_Fields> have been done. </Template>
4	Neg	<Template class="Personal Information"> None of the ARDRA website updates requested by conference participants have been done. </Template>
5	Sel	<Template class="Vendor - Catering"> To cater to the requests of the conference participants, I have updated the <Sel:Session_Name> Banquet and Keynote Lunch </Sel:Session_Name> to include <Sel:Meal_Type> kosher and vegan </Sel:Meal_Type> meals. </Template>

Table 1: Illustrative set of Templates indicating the corresponding Aggregator type. Details are mentioned in Section 3.5.

briefing draft. The ranking system consists of a consensus-based classifier, based on individual classifier models for each user in the training set. The prediction from each classifier are combined to produce a final rank of each template.

We used the Minorthird package (Cohen, 2008) for modeling. Specifically we allowed the system to experiment with eleven different learning schemes and select the best one based on cross-validation within the training corpus. The schemes were Naive Bayes, Voted Perceptron, Support Vector Machines, Ranking Perceptron, K Nearest Neighbor, Decision Tree, AdaBoost, Passive Aggressive learner, Maximum Entropy learner, Balanced Winnow and Boosted Ranking learner.

3.7 Features

The features used in the system are static or dynamic. Static features reflect 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 include:

1. Template id - The unique template number expressed as a set of 29 boolean features corresponding to each of the templates.

2. Template class - The information category type. Similar to template id, the 9 categories (see Figure 3) were expressed as 9 boolean features.

3. Negativity - Boolean feature noting whether the template indicates that tasks were incomplete.

4. Abstraction - Boolean feature indicating if the template expresses an abstraction over some underlying entity, for example usage of 'personal information' instead of 'name, address, phone number etc'.

5. Granularity - Feature coding, for example, if it's a detailed template or a succinct one. We designed templates with 5 levels of granularity coded as a set of 5 boolean features. The dynamic features are:

1. Qualitative feature - Feature reflecting the status of task completion for a report item; it is valid only for certain templates. For example if the user completed 4 out of the proposed 10 'Website:Personal Information' tasks then the qualitative feature for 'Website:Personal Information' template is heuristically determined to be 'some'. We considered four levels of qualitative values (few, some, most and all) producing a set of 4 boolean features.

2. Individual task coverage and time taken - Numeric features corresponding to the percentage of completed tasks and the percentage of time spent on the tasks related to the template respectively. For example, if there were 15 'Website:Personal Information' tasks that were done and the total number of tasks done was 75, then the value of the task coverage feature for the related items would be 15/75 = 0.2. This feature captures the intuition about the relative amount of work done by the user associated with this item, the higher the amount of work done, the more likely for that item to be important.

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 respectively. For example, if the global task completion average for 'Website:Personal Information' 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, the more likely for that item to be important and vice versa.

We used the Information Gain (IG) metric for feature selection, experimenting 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.

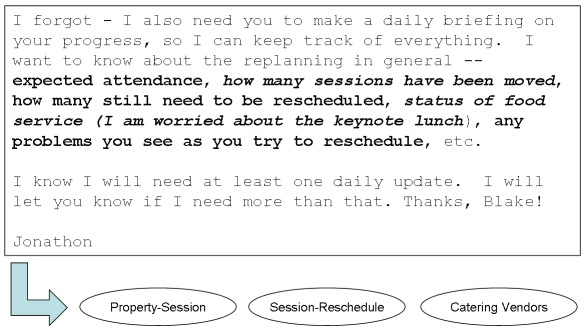

Figure 4: Template categories corresponding to the Briefing request email. The mapping from email text to category is as follows: "expected attendance" $->$ Property-Session; "how many sessions have been rescheduled", "how many still need to be rescheduled", "any problems you see as you try to reschedule" $->$ Session-Reschedule; "status of food service (I am worried about the keynote lunch)" $->$ Catering Vendors.

4 Experiments and Results

4.1 Experimental Setup

The briefing system was evaluated as a part of the larger cognitive assistant system. (Steinfeld et al., 2007) describes the overall test methodology. The goal of the evaluation was to evaluate 'Learning in the Wild' (LITW) which implies that the system should improve its performance through user interaction and not knowledge engineering. The briefing system, used data from a training phase to select features, a learning scheme for individual models and for setting model parameters. This approach differs from more conventional machine learning where models for user testing/deployment are typically selected a priori. On the other hand the present paradigm provides the system with an additional degree of freedom that allows it to adapt more readily to novel circumstances.

Two experimental conditions were used to differentiate performance based on knowledge engineering, designated MinusL and performance based on learning, designated PlusL. In the context of the evaluation, learning is only LITW. Learning acquired through knowledge engineering (template design etc) would be available in both the PlusL and MinusL conditions.

4.1.1 Email Trigger

In the simulated crisis for the conference replanning domain, the briefing was triggered through a mail containing explicit information requests, not known beforehand. Thus to customize the briefing report for the request, a natural language processing module identified the categories of information requested. The details of the module are beyond the scope of the current paper as it is external to our system; it took into account the template categories we earlier identified. Figure 4 shows a sample briefing email stimulus and the corresponding template categories.

4.1.2 Training

Eleven expert users[2] were asked to provide training by using the system then generating the end of session briefing using the BA GUI. For this training phase, no item ranking was performed by the system, i.e. all the templates were populated by the aggregators and recommendations were random. The expert user was asked to select the best possible four items and was further asked to judge the usefulness of the remaining items. The resulting training data consists of the activity log and extracted features and the user-labeled items. The trigger message for the training users did not contain any specific information request.

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

4.1.3 Test

Subjects were recruited to use the crisis management system in MinusL and PlusL condition, although they were not aware of the condition of the system and they were not involved with the project. The details of the subject recruitment criteria are mentioned in (Steinfeld et al., 2007). There were 54 test runs in the MinusL condition and 47 in the PlusL condition. Out of these runs, 29 subjects in MinusL and 43 subjects in PlusL wrote a briefing using the BA. We report the evaluation scores for this set.

4.2 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 users. We calculate two variants of 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 first metric indicates whether the right category of information was selected and the latter indicates whether the information was presented at the appropriate level of detail.

We also performed subjective human evaluation using 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. They were advised about certain briefing-specific characteristics (e.g. negative bullet items are useful and hence should be rated favorably). They were also asked to provide a global assessment of report quality, and evaluate the coverage of the requests in the briefing stimulus email message. This procedure was very similar to the one used as the basis for template selection.

4.3 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 a total of 77 different configurations 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 that configuration with the highest score as the final trained system configuration. Ties are broken by choosing the version with a simpler learning scheme[3] otherwise choosing the one with a lower feature count. The learned system configuration in the current test includes Balanced Winnow (Carvalho et al., 2006) and the Top 7 features.

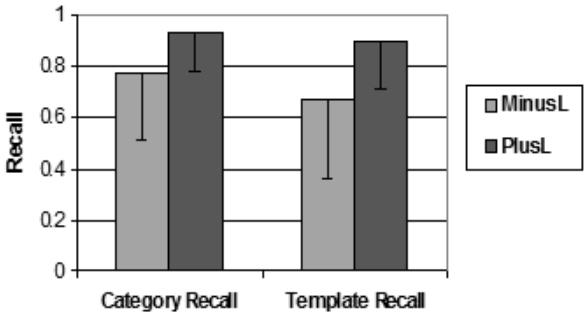

Figure 5: Recall values for MinusL and PlusL conditions.

4.4 Results

We noticed that four users in PlusL condition took more than 8 minutes to complete the briefing when the median time taken by the users in PlusL condition was 55 seconds, so we did not include these users in our analysis in order to maintain the homogeneity of the dataset. These four data points were identified as extreme outliers using a procedure suggested by (NIST, 2004). [4] There were no extreme outliers in MinusL condition.

Figure 5 shows the Recall values for the MinusL

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

[4] Extreme outliers are defined as data points that are outside the range $[Q1 - 3 * IQ, Q3 + 3 * IQ]$ in a box plot. $Q1$ is lower quartile, $Q3$ is upper quartile and IQ is the difference $(Q3 - Q1)$ is the interquartile range.

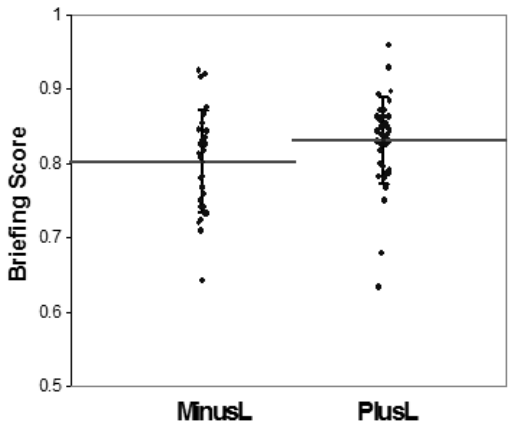

Figure 6: Briefing scores from the judges' panel for MinusL and PlusL conditions.

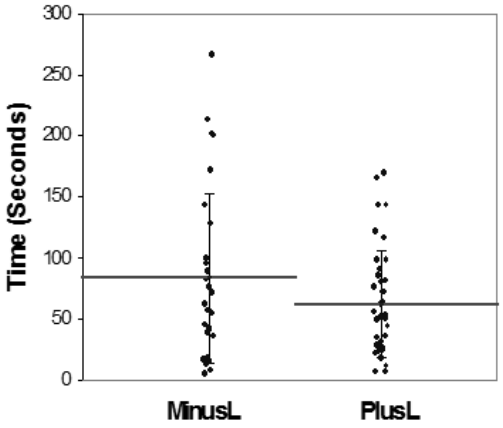

Figure 7: Briefing time taken for MinusL and PlusL conditions.

and PlusL conditions. The learning delta i.e. the difference between the recall values of PlusL and MinusL is 33% for *Template-based* recall and 21% for *Category-based* recall. These differences are significant at the $p < 0.001$ level. The statistical significance for the *Template-based* metric, which was the metric used for selecting system parameters during the training phase, shows that 'Learning in the Wild' works in this case. 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 0.77 and *Template-based* recall of 0.67 in MinusL is not unexpected.

Figure 6 shows the Judges' panel scores for the briefings in MinusL and PlusL condition. The learning delta in this case is 3.6% which is also statistically significant, at $p < 0.05$. The statistical significance of the learning delta validates that the briefings generated during PlusL conditions are better than MinusL condition. The absolute difference in the qualitative briefing scores between the two conditions is small because MinusL users can select from all candidates, while the recommendations they receive are random. Consequently they need to spend more time in finding the right items. The average time taken for a briefing in MinusL condition is about 83 seconds and 62 seconds in PlusL (see Figure 7). While the time difference is high (34%) it is not statistically significant due to high variance.

Amongst the top 10 most frequently selected features across users for this system, four of them are dynamic features. This indicates that the learning model is capturing the user's world state and the recommendations are related to the underlying events. We believe this validates the process we used to generate briefing reports from non-textual events.

5 Summary

Our approach is not meant to discover the generic attributes of good reports; rather it provides a framework within which to rapidly learn the attributes of good reports within a particular domain (as defined by information and its consumers). As such, these custom attributes are more likely to lead to quality reports. We note that a consensus-based modeling approach yields the best performance, indicating that data-driven techniques can be used to select useful models, in this class of domain and possibly others.

6 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 sen-

tences 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.

7 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

Regina Barzilay and Lillian Lee. 2004. Catching the drift: probabilistic content models, with applications to generation and summarization. In *Proceedings of the NAACL*.

Anja Belz. 2007. Probabilistic generation of weather forecast texts. In *Proceedings of NAACL-HLT*.

Paul N. Bennett and Jaime Carbonell. 2005. Detecting action-items in e-mail. In *Proceedings of SIGIR*.

Stephan Busemann. 2005. Ten years after: An update on TG/2 (and friends). In *Proceedings of European Natural Language Generation Workshop*.

Vitor Carvalho, William Cohen, and Avrim Blum. 2006. Improving winnow for nlp tasks: Voting schemes, regret minimization and online feature selection. In *CMU Technical Report*.

William W. Cohen. 2008. Minorthird: Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data. http://minorthird.sourceforge.net. Verified 8th June 2009.

Naomi Daniel, Dragomir Radev, and Timothy Allison. 2003. Sub-event based multi-document summarization. In *Proceedings of HLT-NAACL*.

Pablo A. Duboue and Kathleen R. McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the EMNLP*.

Pablo A. Duboue. 2004. Indirect supervised learning of content selection logic. In *Proceedings of the INLG*.

Andrew Faulring and Brad A. Myers. 2006. Availability bars for calendar scheduling. In *Proceedings of CHI*.

Elena Filatova and Vasileios Hatzivassiloglou. 2004. Event-based extractive summarization. In *Text Summarization Branches Out: ACL-04 Workshop*.

Eugene Fink, P. Matthew Jennings, Ulas Bardak, Jean Oh, Stephen F. Smith, and Jaime G. Carbonell. 2006. Scheduling with uncertain resources: Search for a near-optimal solution. In *IEEE International Conference on Systems, Man, and Cybernetics*.

David Garlan and Bradley Schmerl. 2007. The radar architecture for personal cognitive assistance. *International Journal of Software Engineering and Knowledge Engineering*.

Thomas Hofmann. 2004. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*.

Mohit Kumar, Dipanjan Das, and Alexander I. Rudnicky. 2007a. Summarizing non-textual events with a 'briefing' focus. In *Proceedings of RIAO*.

Mohit Kumar, Nikesh Garera, and Alexander I. Rudnicky. 2007b. Learning from the report-writing behavior of individuals. In *Proceedings of IJCAI*.

Mohit Kumar, Dipanjan Das, and Alexander I. Rudnicky. 2008. Automatic extraction of briefing templates. In *Proceedings of the International Joint Conference on NLP*.

Wenjie Li, Mingli Wu, Qin Lu, Wei Xu, and Chunfa Yuan. 2006. Extractive summarization using inter- and intra- event relevance. In *Proceedings of ACL*.

Inderjeet Mani, Kristian Concepcion, and Linda Van Guilder. 2000. Using summarization for automatic briefing generation. In *NAACL-ANLP Workshop on Automatic summarization*.

Mark T. Maybury. 1995. Generating summaries from event data. *Information Process Management*.

Prem Melville, Raymod J. Mooney, and Ramadass Nagarajan. 2002. Content-boosted collaborative filtering for improved recommendations. In *Proceedings of Eighteenth national conference on Artificial Intelligence*.

NIST. 2004. Nist/sematech e-handbook of statistical methods. http://www.itl.nist.gov/div898/handbook/. Verified 8th Jun 2009.

Alice Oh and Howard Shrobe. 2009. Generating baseball summaries from multiple perspectives by reordering content. In *Proceedings of the INLG*.

F Portet, E Reiter, A Gatt, J Hunter, S Sripada, Y Freer, and C Sykes. 2009. Automatic generation of textual summaries from neonatal intensive care data. *Journal of Artificial Intelligence*.

Dragomir R. Radev and Kathleen R. McKeown. 1998. Generating natural language summaries from multiple on-line sources. *Journal of Computational Linguistics*.

Regis Saint-Paul, Guillaume Raschia, and Noureddine Mouaddib. 2005. General purpose database summarization. In *Proceedings of VLDB*.

A. Steinfeld, R. Bennett, K. Cunningham, M. Lahut, P. Quinones, D. Wexler, D. Siewiorek, J. Hayes, P. Cohen, J. Fitzgerald, O. Hansson, M. Pool, and M. Drummond. 2007. Evaluation of an integrated multi-task machine learning system with humans in the loop. In *Proceedings of NIST Performance Metrics for Intelligent Systems Workshop (PerMIS)*.

Kees van Deemter, Emiel Krahmer, and Marit Theune. 2005. Real versus template-based natural language generation: A false opposition? *Journal of Computational Linguistics*.

Mingli Wu. 2006. Investigations on event-based summarization. In *Proceedings of ACL*.

Wei Xu, Wenjie Li, Mingli Wu, Wei Li, and Chunfa Yuan. 2006. Deriving event relevance from the ontology constructed with formal concept analysis. In *Proceedings of CICLing*.

John Zimmerman, Anthony Tomasic, Isaac Simmons, Ian Hargraves, Ken Mohnkern, Jason Cornwell, and Robert Martin McGuire. 2007. VIO: A mixed-initiative approach to learning and automating procedural update tasks. In *Proceedings of CHI*.