4-2017

eAssistant: Cognitive Assistance for Identification and Auto-Triage of Actionable Conversations

Hamid R. Motahari Nezhad

Kalpa Gunaratna
Wright State University - Main Campus, gunaratna.2@wright.edu

Juan Cappi

eAssistant: Cognitive Assistance for Identification and Auto-Triage of Actionable Conversations

Hamid R. Motahari Nezhad
IBM Almaden Research Center
San Jose CA, USA
motahari@us.ibm.com

Kalpa Gunaratna*
Kno.e.sis Center
Wright State University
Dayton OH, USA
kalpa@knoesis.org

Juan Cappi
IBM Almaden Research Center
San Jose CA, USA
jmcappi@us.ibm.com

ABSTRACT

The browser and screen have been the main user interfaces of the Web and mobile apps. The notification mechanism is an evolution in the user interaction paradigm by keeping users updated without checking applications. Conversational agents are posed to be the next revolution in user interaction paradigms. However, without intelligence on the triage of content served by the interaction and content differentiation in applications, interaction paradigms may still place the burden of information overload on users. In this paper, we focus on the problem of intelligent identification of actionable information in the content served by applications, and in particular in productivity applications (such as email, chat, messaging, social collaboration tools, etc.). We present `eAssistant`, which offers a novel fine-grained action identification method in an adaptive, personalizable, and online-trainable manner, and a cognitive agent/API that uses action information and user-centric conversation characteristics to auto-triage user conversations. The introduced method identifies individual actions and associated metadata; it is extensible in terms of the number of action classes; it learns in an online and continuous manner via user interactions and feedback, and it is personalizable to different users. We have evaluated the proposed method using real-world datasets. The results show that the method achieves higher accuracy compared to traditional ways of formulating the problem, while exhibiting additional desired properties of online, personalized, and adaptive learning. In `eAssistant`, we introduce a multi-dimensional learning model of conversations auto-triage, defined based on a user study and NLP-based information extraction techniques, to auto-triage user conversations on social collaboration and productivity tools.

*The work has been done while the author has been a research intern at IBM Almaden Research Center. All authors have made significant equal contributions.

WWW 2017 Companion, April 3-7, 2017, Perth, Australia.
ACM 978-1-4503-4914-7/17/04.
http://dx.doi.org/10.1145/3041021.3054147

Keywords

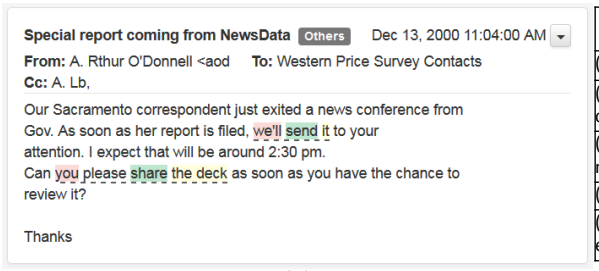
Cognitive Assistance; Natural Language Interface; Natural Language Understanding; Information Extraction; Personalization; Online Learning; Activity Management

1. INTRODUCTION

From the advent of the Web, the browser has been the key user interface for consumption and interactions with Web application systems [23]. Since then, Web browsers have evolved in different directions including offering active mechanisms for user interaction [10]. Application development for mobile shifted the focus from Web applications to native mobile applications. Next, the notification mechanism became a defacto method for delivering updates to users, and reduced reliance on apps to the extent that some called it a threat to the app concept, relevance, and usage [1]. Going forward, conversational agents with advanced natural language and voice dialog capabilities are posed to revolutionize the user interface paradigm and become the default user interaction method for applications [18]. Nevertheless, without intelligence of the content served by applications, neither of notification-based method nor conversational agents may help with the issues of information overload and productivity issues specifically for enterprise workers [19], rather they may further contribute to overwhelming the users and distort their interaction experience.

The key problem investigated in this paper is the intelligent identification of pieces of content which is of interest to a user (an enterprise worker) across conversations channels on collaboration tools (e.g., emails, chat, messaging, and enterprise social collaboration tools) whether they are used through Web interfaces or mobile apps. The goal is to auto-triage conversations (and therefore notifications) that the user receives, thereby offering an intelligent and cognitive user interface.

As a factor of the relevancy and of the interest of a piece of content to the user in an enterprise context, we focus on content that contains statements calling on, suggesting, requesting, and making commitments to the user of interest in natural language conversations, which we refer to as *actionable statements*. While there are many classes of actionable statements, for the sake of the description of the proposed method in this paper, without loss of generality, we focus on two generic classes of actionable statements, i.e. *requests* and *promises*. A request is a statement that is asking for an action to be taken by the receiver of the message in a conversation. For instance, “Can you please send me the file by tomorrow?" is an example of an explicit request. On the other hand, a promise is a commitment made by the sender of the message to one or more recipients (e.g., “I will send you the agreement documentation").



(a)

Natural Language Sentence	# of Promises	# of Requests	Actionable Statement
(1) As soon as her report is filed, we'll send it to your attention.	1	0	Yes
(2) Can you please share the deck as soon as you have the chance to review it?	0	1	Yes
(3) Please review the document and I will set up a meeting next week to go over any questions or concerns.	1	1	Yes
(4) I will know more on this question later today.	0	0	No
(5) We have compiled a list of financial and physical trades executed from September 25 to September 27.	0	0	No

(b)

Figure 1: (a) A screenshot of an email (taken from the Enron email [12] corpus), in which a promise and a request are highlighted, and (b) examples of statements with promises, requests or neither.

The identification of such actionable statements is important within conversations as they allow for the auto-triage of conversations and therefore function as a way to limit the number of notification updates from Web applications, mobile apps, and social feeds. We conducted a user study for conversation auto-triage that revealed the problem of auto-triage is a multi-dimensional problem. Beyond understanding the content of conversations, it requires understanding the users (senders and receivers), aspects such as urgency, action type, etc. In addition, the user study called for the method to be personalizable, to continuously improve by learning from the user feedback, and take into account the domain that the user is operating in.

In this paper, we introduce a cognitive assistant framework called `eAssistant` for learning different classes of actionable statements in the context of human conversations in a personalized and online manner. `eAssistant` uses information from actionable statement identification along with other conversation information such as sender(s)/receiver(s) and target user(s) and features extracted from the content of conversations related to actions to provide an auto-triage mechanism for conversations. We use the term *cognitive assistance* to refer to offering an intelligent solution based on artificial intelligence, natural language processing, and machine learning for learning and identifying actionable statements and for auto-triage of conversations carried out over various Web, mobile, and social applications. Figure 1(a) shows an example of the actionable statement output of `eAssistant` identified in an email. Figure 1(b) shows examples of three actionable statements and two non-actionable statements in our email corpus.

The paper makes two major contributions. First, it presents a novel method for learning actionable statements at a *fine-grained* (action) level in natural language sentences through learning *action patterns* that are formed using an order-preserving signature of select tokens, PoS tags, combined with additional linguistic and semantic role features [16] of each action verb. The introduced method exhibits the following properties:

- The learned model is *white-box*, meaning that the model predictions can be explained by examining the learned action patterns.
- The presented method is *adaptive*, meaning that the candidate feature set for forming action patterns can be expanded as the learning continues over time.
- It is *online*, i.e. new action patterns can be learned in a continuous manner and learned instantly with new training samples through user feedback.
- The learned model can be *personalized* and *customized*. We define a hierarchical, tree-like structure for organizing the learned models in which the action patterns can be learned at the global level, the top node of the tree, and at personal levels positioned in the leaves. This structure allows for customizing of the learned model for specific organizations, domains, and individual users.

The next major contribution of the paper is defining a multi-dimensional framework for the auto-triage of natural language conversations by conducting a systematic user study and offering the `eAssistant` cognitive service, which takes into account the identified actionable statements, associated metadata such as timing, subject, objects, and projected importance of the senders/receivers, observed from past user conversations, to identify actionable conversations in a given conversation stream, specifically in emails, chats, messaging applications, and social collaboration streams. These set of features are used to formulate the problem as an adaptive and personalized classification problem, which uses a classifier along with a rule-based classification model, which is auto-generated and configured as the result of user feedback on the auto-triage classes.

We have evaluated the actionable statement learning and identification method, and auto-triage using conversation data from real-world settings. In particular, we have used conversations in the Enron Email Corpus [12] and a separate set of corporate email conversations. We compare and contrast the result of our method with that of a black-box classification method used for actionable sentence identification, which shows that our approach achieves favorable accuracy results. Additionally, it has the advantage of providing an action-level, online, adaptive, and personalizable learned model.

The paper is organized as follows. First, we describe the problem of actionable statement identification, followed by our proposed actionable learning and identification approach. Next, we present the conversation auto-triage process. Then, we present the implementation details of `eAssistant`, the evaluation, and the comparative analysis results. Finally, we discuss related work and conclude with future directions.

2. ACTIONABLE STATEMENT IDENTIFICATION PROBLEM

In general, actionable statements in conversation may be categorized into a number of classes depending on the context and domain, e.g. orders, needs, questions, commitments, etc. At the highest level of abstraction and without loss of generality, in this paper we assume actionable statements are captured in the two classes of *Promises* and *Requests* exchanged between human (knowledge) workers in an enterprise context. Given two parties of people X and Y and an action A, where $|\mathrm{X}| > 0$ and $|\mathrm{Y}| > 0$, we define a request and a promise as follows.

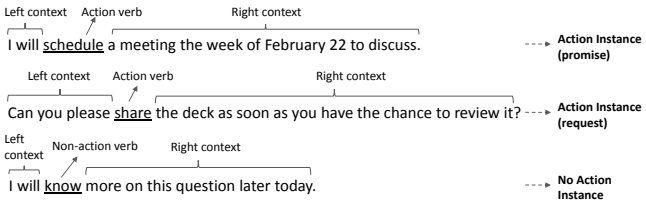


Figure 2: Action instances and action verbs.

(Request). Request R is a tuple $R = (X, Y, A, T)$, where X is requesting (assigning) Action A to Y to perform, optionally by the due date/time T.

In a conversation, the sender of a message is usually the requester (assigner) for a request. And we define a promise as follows:

(Promise). Promise P is a tuple $P = (X, Y, A, T)$, where X is committing to perform Action A for party Y, optionally by the due date/time T.

Similarly, in a conversation, the sender of a message is usually the committer of a promise. Note that depending on the action type there may be additional parameters of interest to be extracted for the given action from the text. For instance, for a "send file" action, the desired parameter is "File Name" and "File Type", and for a "calendar invite" action, additional parameters include a "List of Required Participants, List of Optional Participants, Time of the Event, Venue/Place, etc."

Let us illustrate the problem formulation with some examples of actionable statements in human conversations, Figure 2 shows two actionable statements and a sentence that does not contain any action. We refer to an occurrence of an instance that belongs to one of the two action classes in a natural language sentence (referred to as sentence here onwards) as an *action instance*. In Figure 2, the first two sentences contain a promise and a request, respectively, and hence contain one action instance each. We consider an action instance to be focused around a verb and its left and right context in the sentence. An actionable statement is identified as a combination of verbs and *action patterns*. In the sequel, an action pattern is defined as a collection of a specific ordering of tokens and POS tags, and other features derived from them that are related to the verb. Let V be all the verbs, T be the list of all tokens except V, and G be the list of POS tags[1] of T in a sentence S. An action pattern AP in sentence S is defined as follows.

(Action pattern). An action pattern is a three-tuple item, $AP = (v, d, f)$, where $v \in V$ and d is an ordered sequence of extracted and constructed features f. $f \in T \cup G$.

When an action pattern specifies an action instance (i.e. *positive action pattern* for one of the classes of interest), the associated verb is called an *action verb*. Note that the same tokens and tags can appear in two action instances that belong to different classes (e.g., one a promise and the other a request) for which the action patterns differ from each other because the order of features are different). Hence, identifying/learning tokens and tags as individual elements does not help to correctly identify action instances and the respective action classes. In general, a sentence can have more than one action pattern when it has more than one verb (i.e., more than one action instance). For instance, consider the sentence "Please review the document and I will set up a meeting next week." It has two verbs (review and set up) that contribute to two different action instances. The first is a request and the second is a promise. It is important to recognize these two action instances separately rather than identifying the whole sentence as an actionable statement. This is one reason that we are interested in extracting and learning patterns with respect to each verb. In doing so, we can better assist users by tagging relevant action items for each action instance in a sentence for efficient processing and information presentation (e.g., Figure 1 (a)). Given the above, the problem of actionable statement identification is defined as follows.

(Actionable statement identification). Given a sentence S, a set of pre-defined action classes $\{c_1, c_2, ...c_k\} \in C$, and a set of action patterns $\{ap_1, ap_2, ...ap_n\} \in AP$ in S, the problem of actionable statement identification is to correctly select class $c_j \in C$ for each $ap_i \in AP$.

[1]Includes inferred or computed tags.

3. ACTIONABLE STATEMENTS: LEARNING AND IDENTIFICATION

First, we extract features from sentences focusing on each verb. During learning, the proposed method learns action verbs and action patterns. Learning action verbs is important as they are a major discriminator in the identification of action instances (i.e., certain verbs are not actionable in some domains). The learned model consists of different types of action verbs (independent and enclosed) and action patterns that specify valid actionable statement forms for each action class. To predict whether each verb in a given sentence defines an action instance in one of the supported classes, we extract the set of features related to the verbs and use the learned model (consisting of action verbs and action patterns models) to predict the class of each verb in that sentence. The learned model can be updated from continuous user feedback and training. Next, we outline our approach in the actionable statement identification process.

3.1 Feature Space Definition

We define a set of feature candidates related to each verb which are used to learn various action patterns that identify action instances. Feature candidates consist of POS tags, semantic role features of the verb of interest [16, 5], and language token types, including:

1. Pronoun (PRON) token, auxiliary (AUX) token, personal named entities (NER).
2. Mood of the verb.
3. Whether the verb is an action verb (learned separately).
4. Whether the verb encloses other action verb/s.
5. The number of enclosed action verbs > 0.
6. The tense of the verb.

For the above features, and in particular in text parsing, part-of-speech labeling, and dependency parsing, we rely on state-of-the-art NLP parsers. In particular, we use IBM BigInsights SystemT [5] which has an associated query language called AQL (Annotation Query Language), which is used to get advanced text annotations, including POS tags, semantic role labels, dependencies, etc. Using AQL, the tokens of pronoun, auxiliary, and the NER tag are selected from the left context of the verb. If there are more than one of these available in the left context, the closest one to the verb is taken. The mood of a verb changes based on the place it appears in the sentence and how it is used in conjunction with prepositions (possible moods are: normal, imperative, infinitive, etc.). The notion of an enclosed verb in the list of features above refers to a verb

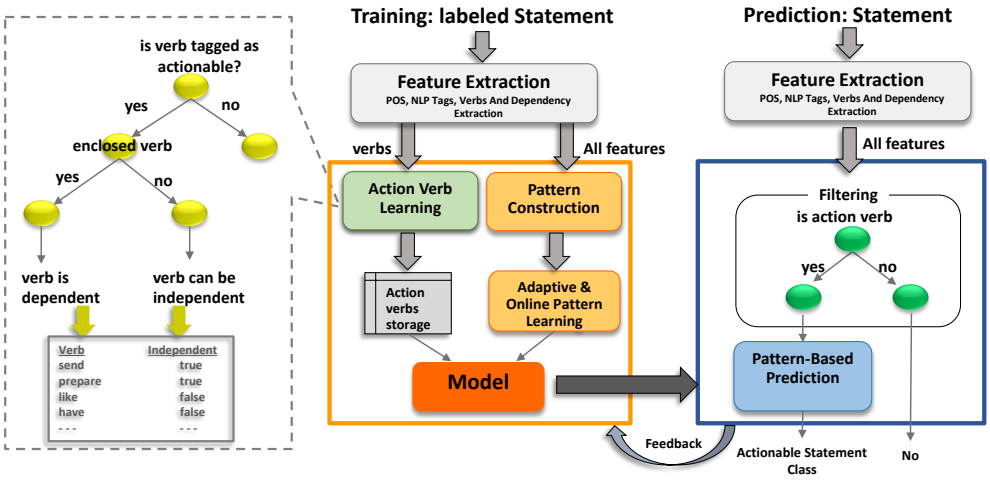


Figure 3: Learning and prediction of actionable statements.

that is not the dominating verb for the action instance but states "what/how" actions for the action instance. For example, in the sentence "Please schedule a meeting to discuss," the verb "discuss" (enclosed verb) is enclosed by the verb "schedule." When a verb is enclosing one or more verbs and at least one of them is an action verb, then it has > 0 enclosed action verbs. Finally, the tense of the verb is used as another feature (past-tense verbs do not specify an action instance).

3.2 Learning Action Verbs and Action Patterns

The learning in the proposed approach consists of two important components: (i) learning action verbs and (ii) learning action patterns. Figure 3 depicts the learning process of actionable statements.

3.2.1 Learning Action Verbs

Identifying a verb as an action verb or not is important. Recall that not all verbs are considered actionable in certain work contexts (domains), although they may appear in patterns that are identical to those of action patterns. For example, consider the following sentence fragments. Example 1: "I will rest"; example 2: "I will meet." Both sentences have similar tokens except for the verbs, but the first one is not considered an action instance because the action described by the verb is not important for the work environment. It should be noted that having an action verb in a sentence is a necessary condition but not sufficient to make it an actionable statement (e.g., "You could schedule."). Therefore, action verbs are learned from the training samples as follows. If a verb is tagged as an action type and is independent (not enclosing verbs), we record it as '<verb,true>' (independent or singular). If the verb encloses other verbs and is tagged as an action type, we record it as '<verb,false>' (dependent or compound). If there is a single instance where a compound verb acts as singular, we update the verb as singular. This helps the system to identify complex patterns where the verb participates in defining an action but indirectly. For example, in the sentence "I would like to schedule", "like" is a dependent action verb. Algorithm, 1 shows the proposed method for learning different forms and language constructs that action verbs appear in.

Algorithm 1 Learning Action Verbs

```
1:  input: String label, String verb, Set enclosed_verbs
2:  output: updated verb Map
3:  initialize or get existing Map<Verb, Boolean>
4:  if label ∈ Type then
5:      if |enclosed_verbs| > 0 then
6:          if Map does not contain verb then
7:              Map ← <verb, false>
8:          end if
9:      else
10:         Map ← <verb, true>
11:     end if
12: end if
```

Compound action verbs can be complex in statements and be found through following a chain of enclosed verbs until a singular action verb is reached. In case of compound verbs as "like", we record that the verb "like" can appear in action instances in the enclosed form. In the algorithm, if the label of the verb is of class promise or request, and it is not enclosed with any other verb, then we record the verb with the value "true" (line 10). That means, the verb appeared in a positive sample for a class and does not depend on another verb. If the labeled verb (to a class) encloses other verbs and there is no record of the verb as an independent, singular form, we record the verb with the value "false" (line 7). This means that the verb cannot be used independently (alone) in action instances. Note that we update an action verb as dependent if we have not ever discovered it in an independent example (line 6).

3.2.2 Building and Learning Action Patterns

We first extract all available features among the candidate features for a given action verb in a labeled sample, the patterns are formed based on the observed order of tags and tokens in the statement. In forming a pattern, the key decision point is how generic and specific patterns could be defined. Very specific patterns (e.g., all token-level patterns for an actionable statement) lead to over fitting and therefore low recall. On the other hand, learning at all POS tag levels and other syntactic and semantic linguistic features (such as verb mood, etc.) will lead to overgeneralized patterns and

therefore a low precision. We decided to form hybrid action patterns in which a combination of tag-level and token-level features and semantic (role label) features are present.

We define an action pattern as a set of features including a sequence of tokens, tags (including features 1-5), verb-centered features (e.g, feature 6) and semantic features. During learning, the proposed method takes an order-preserving combination of tokens and tags (features 1-5) as one feature (called signature feature) as they need to be considered together. We take the order of the tokens and tags as they appear in the sentences for as feature 1 (e.g., the order of tags and tokens of feature 1 for the verb in the phrase "John, could you do it?" is "NER,could,you"). Our model accommodates the addition of new features in training for all potential tokens and tags on the left-hand side of the action verb since all such features are considered candidate features in forming the signature feature [2].

We adapt a Winnow-based multi-class classifier for learning action patterns. We make use of the SNOW [4] representation, which is similar to a one-layer neural network but differs from how it updates weights. The weight updating process is mistake-driven and does both promotion and demotion of weights. The network has target nodes and each input feature has weights associated with each target node. We have the target nodes set as promise, request, and other. A positive prediction means the algorithm predicts a target node (having the highest weight aggregation of features) and negative prediction otherwise. Let t, α, and β be the target node, promotion constant, and demotion constant, respectively. Promotion and demotion parameters are positive constants, $\alpha > 1$ and $0 < \beta < 1$. Let $A_t = \{A_1, .., A_m\}$ be the set of *active features*[3] for a given example linked to the target node t.

The algorithm promotes weights of the target node t as shown in Equation 1 when it predicts negative, where $\sum_{i \in A_t} w_{t,i}$ is not the highest value and the provided label matches t.

$$\forall_i, w_{t,i} = w_{t,i} \cdot \alpha \tag{1}$$

It demotes weights of the target node t as shown in Equation 2 when it predicts positive, where $\sum_{i \in A_t} w_{t,i}$ is the highest among the target nodes and the label does not match t.

$$\forall_i, w_{t,i} = w_{t,i} \cdot \beta \tag{2}$$

All the other weights remain unchanged, including those of features that are not present in the current sample. In the prediction stage, we take the highest aggregated weighted class as the prediction (winner takes all). In rare but possible occasions of a tie, we make the decision based on a separate support score (separately maintained and explained later) to select either the promise or request classes.

3.3 Actionable Statements Identification

As Figure 3 depicts, our actionable statement identification (prediction) is a two-step process. In the first step, we check whether the verb has been learned as an action verb (either independent or dependent) and in the second step, we perform the pattern matching-based identification. This two-stage identification process helps our approach to avoid having to keep track of every verb in the action patterns, which would explode the number of action patterns it has to learn. If the verb is an independent or dependent action verb, we do the prediction by applying the SNOW algorithm. Otherwise, we determine it is not an actionable statement. Note that our model is able to learn the set of action verbs from the training samples (i.e., bootstrapped). In our implementation, we bootstrapped the dictionary of action verbs with few hundreds of seed action verbs taken from the domain and existing users of an enterprise email client, which is expanding through the continuous learning process.

[2]Inclusion of other new features may require new model update operations to be defined, outside of the set of candidate features for inclusion in signature patterns

[3]features present in the in-hand statement that is being processed.

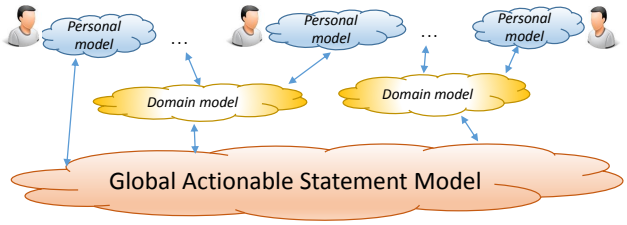


Figure 4: Hierarchy of models for personalized and domain-specific learning of actionable statements.

3.4 Domain-Specific and Personalized Model Learning

Previous sections described how we build a single global model of actionable statements from training data. In general, action verbs and action patterns may differ from one industry domain or business sector to another (such as healthcare vs. finance). Different users may also consider certain statements actionable or not, depending on their work environment and preferences. We describe a framework for enabling the learning of domain-specific and personalized models for actionable statements. This is performed through defining hierarchical and layered models for actionable statements, e.g. with the three levels of global, domain-specific, and personal models. Figure 4 shows an illustration of the hierarchy of models.

3.4.1 Learning personalized models

The personalized model learning is managed through the user feedback mechanism. During initial training, the system is bootstrapped with the global model which is learned using the set of initial training data (see Figure 3). If personalized learning is not enabled, all users feedback will be assessed and applied to the global model. When it is enabled, the user feedback, which translates into learning new action pattens (asc described in Section 3.2.2) to be included or excluded from the model for the given user, is stored in their personalized models. We define a personal feedback promotion mechanism to surface actionable statement patterns for which there are enough statistical support from different users.

The users feedback is collected in two forms: (i) *incorrect actionable statement identification*: flagging an identified action as not being a request or a promise, and (ii) *missing actionable statements*: flagging a statement (and thereby the verb within a statement) as being a promise or request that is not captured. The user feedback is assessed against the matching action pattern in the model if it exists, and leads to the promotion or demotion operation on the features weights. If not exist, new patterns are created and added to personal models.

In order to track the support for user feedback, we define two parameters associated with each action pattern: (1) a support score s, which keeps track of the number of times that the same feedback (existing pattern in the global model or a new pattern) has received positive support from the users and (2) the list of users who have supported the pattern. We apply a threshold γ on s for each pattern to identify which model level they belong to. For a given action

pattern ap, when $s(ap) < \gamma$, the pattern will stay active only for the users who support it. And, if $s(ap) \geq \gamma$, it is promoted to the higher-level model in the hierarchy (the global or domain-specific model, depending on the configuration). If the feedback is to demote a given pattern ap in the global model for which after demotion still $s(ap) \geq \gamma$, it remains active in the global model, but it would be recorded in the personal model of the user who demoted it as an excluded pattern. This personalized model allows the user to keep his/her personal patterns in his/her personalized model when there is no system-wide or domain-wide agreement.

3.4.2 Learning domain-specific models

The learning of a domain-specific model can be done through a training phase with samples labeled with the specific domain name. This creates known domain models in the hierarchy. Domain models can also be adapted through user feedback by users who belong to that domain. Domain configuration in the system is associated with a domain support score threshold $\sigma << \gamma$. When domains are enabled for a specific user, action patterns with a support score s below σ are kept in personal models, if $\sigma < s(ap) < \gamma$ it belongs to a domain model, and if $s(ap) \geq \gamma$ the pattern is stored in the global model.

3.4.3 Actionable statement identification in domain-specific and personalized statement models

During the actionable statement identification process, priority is given to patterns in the personal, then domain, and finally global models. This enables considering personalized models and domain models over the patterns in the global model. Note that every user may not have a personal model (which occurs when they have not provided any feedback or when their feedback is applied to patterns on the global and domain levels).

4. CONVERSATION AUTO-TRIAGE

Human conversations are carried out with various social collaboration tools with Web or mobile application interfaces. Each new update in a conversation thread (group chat session, text, email, or social feed) can potentially create a notification for a user subscribed to those conversations. The goal of conversation auto-triage is to identify selected updates (messages) within conversation threads that are of interest or importance, and therefore selectively create notifications and increase the productivity of knowledge workers by enabling them to focus on the most important conversations first. In order to understand the criteria for the triage of conversations by users, we organized two separate user studies; each started with a survey and followed by a design workshop with the goal of identifying the criteria and prioritizing them through voting. The user studies were divided into 60 internal enterprise employees and 53 external survey participants. The users had a diverse set of roles, including knowledge workers, managers, business planners, project managers, and administrative assistants, among others. The survey questions were focused around their habits and behavior when dealing with social collaboration tools from emails, chat, social media and how they organize their work. The design workshop then proceeded with a select number of users from both groups representing different enterrprise user roles with the agenda of finding and prioritizing the most common factors for conversation prioritization.

The surveys and user studies led to the following top categories of criteria: (1) the importance of the sender of an update (message) for the receiver, (2) the existence of an ask (action) targeted to the receiver, (3) the urgency of the action, and (4) a commitment/promise made to the receiver. Based on the combination of these factors, we defined 4 classes of priorities: (i) Act Now, (ii) Read/Act Promptly, (iii) Act Later, and (iv) Others; with the following initial, default conditions: (i) Act Now: messages from important people which have an action with immediate timeline signal (e.g., by the end of the day); (ii) Read/Act Promptly: messages from important people, or messages with an action for you with an immediate timeline from any person; (iii) Act Later: any message with an actionable statement not covered by the prior two categories, and messages with an explicit mention of the recipient but no action; all the other update feeds would be classified to Other priority class. Figure 5 shows an example of a Web-based email interface in which the conversation auto-triage for emails has been demonstrated.

Technically, we prepare the features needed for auto-triage as follows. The first feature is the detection of people important to the user: there is a list of important people maintained for a user. This list is a dynamic list compiled from user interactions and considering organizational groups/projects and hierarchies. Also, depending on the organization structure and collaboration tool, other people are added. For example, the manager of the user is automatically added to the list of important people. In the case of email, we analyze the past interactions of the user with conversations in the emails over a specified period of time (e.g., last month) and people to whom the user has replied promptly after receipt are added to this list. This list can also be supplemented externally if the collaboration tools (e.g., email clients) provides their own list of important people.

The other set of features are extracted from the metadata of the message, e.g., From (sender), To/CC (receivers), Subject, Date/Time of an email. The set of action-specific features are extracted from the content of the message. We have developed an action information extraction method to extract metadata attributes of an action such as the intended target user (if explicitly mentioned), object, etc. We also provide a template-based method for the identification of certain types of action verb, e.g., Send/Share file/document, and Calendar/Invite related requests. For such action types, we define templates that include parameters of interests to be extracted in the form of rules written in AQL and an extensible dictionary associated to each parameter. The template-based extraction method looks beyond immediate sentences that contain the action verb and uses the following three techniques: (i) co-reference resolution, (ii) looking for missing parameter values in the neighboring sentences, and (iii) looking through the conversation threads on the same topic (email chains).

Using the featured extracted from the content of metadata of a conversation we trained an SVM classifier for the auto-triage of conversations. The basic method is formulated as a traditional classification problem and the use of SVM classifier, we enable collecting feedback on the classification on each conversation. We use the user feedback in two ways: (i) consistency checking of the criteria in the feedback with what the system is trained on. In some cases, where the analysis may reveal a conflict in class labels for two or more training instances with the same set of criteria but conflicting class labels. This is used to do conflict resolution in the training dataset. (ii) building personalized classification logic for features that are not captured in the classification problem. When feedback is on features that are not fed into the classifier (e.g., keywords that the user considers important for her), it leads to the generation of personal rule-based classification model that is retained in a personal layer. This personal auto-triage model is used to promote/demote the classes returned by the classifier, e.g. for conversations that the user would like to be brought into her attentions or those that are not.

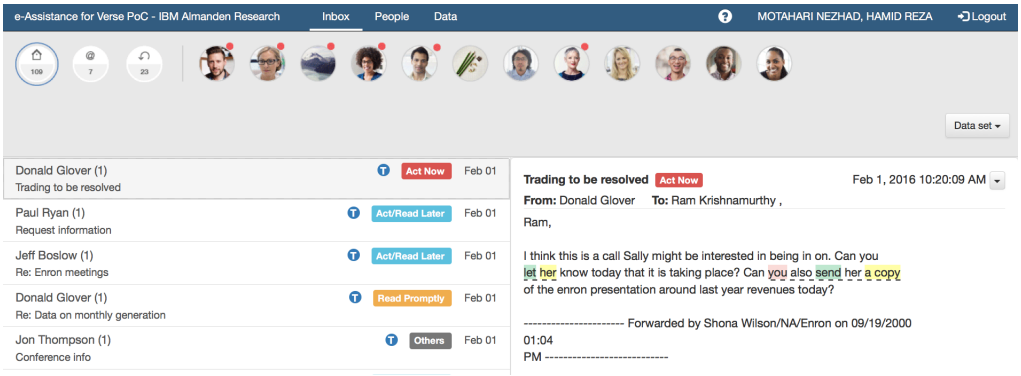


Figure 5: `eAssistant` PoC for Email conversations

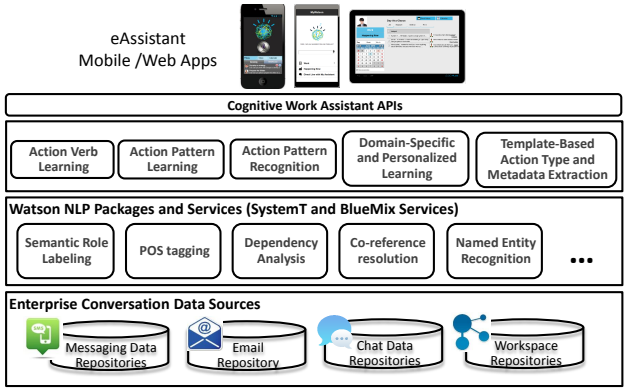


Figure 6: Layered architecture overview of `eAssistant`.

5. IMPLEMENTATION AND EVALUATION

In this section, we present implementation details of `eAssistant` including its layered architecture and evaluation results.

5.1 Implementation

`eAssistant` is implemented as a set of APIs that offer cognitive assistance for actionable statement identification for both text and rich conversations and conversation auto-triage on rich conversations in emails (e.g., IBM Verse [4]), chats, and social collaboration tools (e.g., IBM Watson Workspace [5]). An overview of the layered architecture of `eAssistant` is shown in Figure 6. For linguistic analysis, `eAssistant` uses SystemT [5] libraries (part of the IBM BigInsights platform), as an advanced NLP parser, for PoS tagging, named entity recognition (NER), semantic role labeling, and co-reference resolution. Semantic role labeling helps in identifying the thematic and semantic role that each token plays related to a verb, including subject, object, mood, etc. [16]. PoS tags and named entities are used as features in forming signature feature of action patterns. Co-reference resolution facilitates linking sentences, mainly related to human mentions/entities, and finding values of template parameters in the template-based action type. We implemented our temporal signal identification by defining AQL queries for mentions of time and dates as well as phrases like ASAP, today, and EOD. The `eAssistant` functionalities are exposed through a set of REST APIs. The `eAssistant` APIs are used in PoCs that integrate the functionalisties with email clients and also offered seprately in a PoC tool with a Web interface that is using AngularJS and Twitter Bootstrap. The `eAssistant` PoC is released as a pilot to select early adopter users.

[4] `http://www-03.ibm.com/software/products/en/ibm-verse`
[5] `https://workspace.ibm.com/`

5.2 Evaluation

We evaluated the proposed methods in `eAssistant` using a publicly available email dataset, i.e. Enron [12] email corpus, and a set of private corporate email data sets.

5.2.1 *Dataset*

To evaluate the actionable statements identification method, we extracted sentences from an email set taken from the Enron Corpus and other corporate email datasets. At the sentence level, there are 6,000 sentences in total in the dataset, consisting of 15,404 candidate action instances (with associated action verbs). To create a golden training/test set, we had a handful of domain experts manually tag each of these action candidates as either a promise, request, or neither. This dataset forms our human-labeled ground truth which is used for initial training and testing of the system.

5.2.2 *Experimental Setup*

Actionable statement identification is a classification problem at its core. To provide a basis for comparative analysis, while our approach is a white-box method, we experimented with formulating the problem with the state-of-the-art black-box methods and in particular an SVM classifier (SVM linear kernel, liblinear from scikit-learn library with the same set of features of). Among the black-box classifiers we experimented with for this problem, SVM performed others. This observation is also consistent with that of our prior work for using a black-box method in a similar context [11].

For evaluation, for both approaches of eAssistant and the SVM-based approach, we trained and tested the them by splitting the data into 70% for training and 30% for testing. The data set contained 15,404 action candidates from 6,000 sentences. The reported results below for both approaches are similar to that of a 10-fold cross validation on the same sets, when experiments done for each approach separately). In case of eAssistant, we used 1.5 and 0.8 for α and β, respectively (we performed a search within the space of possible α and β, using a small separate train/test set and these values led to best precision and recall). As metrics, we report on precision, recall, and F1 measure as shown in Equations 3, 4, and 5, respectively.

Approach	Precision	Recall	F1 measure
eAssistant	0.89	0.86	0.88
SVM	0.85	0.86	0.85

(a) Action verb level for three classes, trained on the same features

Approach	Precision	Recall	F1 measure
eAssistant	0.86	0.80	0.83
SVM	0.86	0.83	0.84

(b) Sentence level for three classes, trained on different features

Table 1: Comparison of results. The three classes are promise, request, and other.

	Precision	Recall	F1 measure	Support
Other	0.96	0.98	0.97	3847
Promise	0.84	0.80	0.82	209
Request	0.87	0.81	0.84	506

(a) `eAssistant`

	Precision	Recall	F1 measure	Support
Other	0.96	0.95	0.95	3874
Promise	0.87	0.86	0.86	206
Request	0.71	0.77	0.74	542

(b) SVM

Table 2: Precision, recall, and F1 breakdown at Action Class levels for test set. Support is the number of items.

$$Precision = \frac{\#\ of\ correctly\ identified\ action\ instances}{\#\ of\ identified\ action\ instances} \quad (3)$$

$$Recall = \frac{\#\ of\ correctly\ identified\ action\ instances}{\#\ of\ action\ instances} \quad (4)$$

$$F1 = 2 \cdot \frac{Precision\ \cdot\ Recall}{Precision\ +\ Recall} \quad (5)$$

5.2.3 Result analysis

We designed two separate experiments: (i) actionable statement identification at the verb-level. That is, we predict an actionable statement class for a candidate action verb in the dataset. (ii) actionable statement identification at the sentence-level. This is whether each of approaches can identify sentences that actionable, i.e., contain an actionable statement in them.

Table 1(a) reports the precision, recall, and F-measure for the verb-level experiment. For the sentence level, to make the evaluation setting stricter, we took sentences that have only one action instance within them. There are 5,701 such sentences in our dataset. For the SVM classifier, the frequent set of n-grams are used as features for the sentence-based evaluation, and in particular using a dictionary-based count of 1-2 grams and setting max_{df} to 0.8. Table 1(b) shows the results of the comparison.

As the results show, our pattern-based actionable sentence identification approach slightly outperformed SVM-based action identification in terms of precision for verb-level experiment, and their precisions are comparable at the sentence-level. In terms of recall, the performance of the two methods are the same at verb level, while at the sentence level, SVM-based approach performs slightly better arguably due to the fact that it is trained based on n-grams, which allows it to capture and distinguish phrases that makes a statement actionable. Table 2 shows the break-down of the performance of the two methods at the individual action class level for the verb-level action identification. As shown, `eAssistant` method achieves better precision and recall at identifying requests compared to the SVM-based approach, while it achieves comparable results on the Promise and Other classes.

Beyond outperforming at the accuracy level, Table 3 compares the two approaches on multiple dimensions. It shows the advantages of `eAssistant` method compared to the black-box method. In particular, for fine-grained action identification, the introduced `eAssistant` method is superior by exhibiting properties such as online learning, personalization, explainability (white-box), and domain adaptiveness all of which come as its out-of-the-box benefits.

Conversation auto-triage. We evaluated our our multi-dimensional auto-triage classification method on a larger email dataset with more than 26,000 emails. We annotated each of these emails using a rule-based conversation prioritization, using the criteria defined by our users in the workshop. Then, we conducted a similar 70% vs 30% data split, and trained an SVM classifier for auto-triage. It achieved 82% precision, and 85% recall. We had a supervised evaluation session with 4 users which provided us input on the auto-triage results, and submitted feedback for creating a personal rule-based auto-triage conditions for personalized conditions not captured by the auto-triage logic (based on keywords and context of the conversations). When using these additional criteria through the rule-based promotion of classes, the future emails covered by those conditions, which are misclassified by the classifier are correctly classified, which improves the overall prediction accuracy for each users. The user-defined conditions are translated into AQL-based rules that are applied as part of auto-triage pipeline.

5.2.4 Personalized learning and user feedback on actionable statement identification

We have used the eAssistant prototype in a pilot with 40+ users. It receives user feedback in the form of classification corrections and the highlighting missing (not captured) actions. In the evaluation period, we received more than 250 pieces of feedback for missing action types and nearly 20 correction cases. The analysis of the missing cases shows that more than 70% of the cases belong to introduction of new actionable statement classes (a notable class was “questions") that we have not covered in our base experiments, which we added as a new actionable statement class in learning and predictions. The remaining 30% of missing cases and the corrections led to learning patterns that are kept in the personal models. The benefit of this approach is the instant application of the feedback and a positive reward cycle for the user. The analysis of incorrect cases identified a number of verbs identified as Requests in short statements such as “find it attached" or “go for it" with an imperative verb and no left-hand context. Finding such cases led us to improve our method to keep track of verbs that are exceptions from a valid action patterns in the model (in this case for action pattern for imperative verbs and no left-hand context), and the evaluation of their right-hand side during prediction.

The actionable statement identification method described in the paper is integrated into the pipeline of a set of internal collaboration tools, and is under evaluation by alpha users. In particular, the APIs exposed by `eAsssistant` is consumed by different collaboration applications (email, messaging and social connections, etc) for actionable statement identification and made available to a large experimental user base for their evaluation and feedback.

Approach	Online	Adaptable	Explainable	Domain-specific	Personalizable
eAssistant	Yes	Yes	Yes	Yes	Yes
Classifier-based approaches (e.g.,[11])	No	No	No	No inherent support	No inherent support

Table 3: The multi-dimensional comparison of eAssistant approach with a black-box approach for actionable statement identification

6. RELATED WORK

The problem of action characterization and identification in textual interactions among humans (and in particular over emails) has been extensively studied (see [6] for a bibliography on this topic). This includes introducing theories and concepts such as speech acts [25], language action perspectives [26], frameworks for manual identification and management of activities (e.g., [2, 14, 9, 15]), activity modeling [21], learning and finding action items [20, 11, 22], and the classification and grouping of emails based on activities [17, 8, 13, 7, 24]. The contribution and novelty of the approach presented in this paper lies in the fact that it investigates and makes new contributions to the problem of learning and identifying actionable statements (expressed in complex language patterns) in human conversations in an adaptive, online, domain-aware, and personalized manner. In particular, we cast the problem of the understanding of work-focused human conversations into that of learning and recognizing action patterns and actionable statements from input-labeled samples, and introduce a method that is capable of continuously adapting the learned model as new samples become available and also based on feedback.

The approaches focusing on advanced email user interfaces for manual activity management (e.g. [2, 14, 9, 15]) are not directly comparable to our learning-based approach. We study the existing work in three main categories: (i) manual tagging- and annotation-based methods for identifying actions (e.g., those studied in [17, 13]), (ii) intelligent email systems leveraging machine learning [8, 13, 7, 24], and (iii) classification-based methods for action extraction from textual conversations [20, 11, 22]. The tools and techniques in the first category are not related to the work presented in this paper as they require human interaction in organizing the content. The second category of work (e.g., [8, 13, 7, 24]) offers machine learning methods to classify/identify emails with action statements in them. These approach the problem either as a supervised classification problem of whole email [13] or as an unsupervised clustering problem of grouping all emails with the same activity into the same cluster (e.g., [2]). These operate at the email level and do not focus on detecting individual actionable statements as we do in this paper. The last category of related work (e.g., [20, 11, 22] offers methods for the identification of actions in sentences as a whole. These methods approach the problem as a classical classification problem in which features are also defined in advance and updating the model requires re-training from the start. In addition, these methods are unable to pinpoint individual actions and do not identify multiple action instances within a sentence, if exists. Nevertheless, to provide a comparative basis at the sentence level, we did experiments to compare our approach with that of black box approaches as reported in the evaluation.

In contrast to these approaches, as also depicted in Table 3, our method: (i) is fine-grained in identification (identifies action instances at the level of verbs compared to sentences), (ii) is adaptive (based on feedback and the domain) and online (learns from feedback continuously), and (iii) can be personalized. Our proposed method also outperforms the black-box, SVM model in experiments and achieves comparable recall. Finally, we offer a novel, multi-dimensional, and action-based conversation auto-triage method that is a basis for offering cognitive assistance to improve the human user interaction experience in conversational agents and in Web or mobile applications. In the literature, the problems of conversation, and in particular email classification [3] and email prioritization [27] have been studied. However, these approaches mainly consider the textual content as a whole, and involved people, but to the best of our knowledge none of them are considering action, action metadata, and conversation characteristics.

7. CONCLUSION AND FUTURE WORK

In this paper, we investigated the problem of offering cognitive assistance in work environment, and in particular on the problem of the identification of actionable statements and auto-triage of human conversations. To the best of our knowledge, this is the first work that introduces an adaptive, online, and personalizable method for learning fine-grained actionable statements. We evaluated our implemented framework for its identification of actionable statements and compared it against state-of-the-art machine learning techniques. We showed that our approach is favorably comparable to the state-of-the-art and it is adaptable and teachable through feedback. The developed `eAssistant` prototype system is piloted with a set of selected early adopter enterprise users. As future work, we are investigating deep-learning based methods for conversation auto-triage. We are also planning to integrate and offer `eAssistant` as a cognitive conversational agent on top of productivity and social collaboration tools.

Acknowledgment

We thank Prof. Amit Sheth and Prof. Krishnaprasad Thrirunarayan from Kno.e.sis Center, Wright State University for their insightful comments on early draft of our paper.

8. REFERENCES

[1] P. Adams. The end of apps as we know them. `https://blog.intercom.com/the-end-of-apps-as-we-know-them/`, 2014. Accessed: 01/05/2017.

[2] V. Bellotti, N. Ducheneaut, M. Howard, and I. Smith. Taking email to task: the design and evaluation of a task management centered email tool. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 345–352. ACM, 2003.

[3] E. Blanzieri and A. Bryl. A survey of learning-based techniques of email spam filtering. *Artif. Intell. Rev.*, 29(1):63–92, Mar. 2008.

[4] A. Carlson, C. Cumby, J. Rosen, and D. Roth. The snow learning architecture. Technical report, UIUCDCS, 1999.

[5] L. Chiticariu, R. Krishnamurthy, Y. Li, S. Raghavan, F. R. Reiss, and S. Vaithyanathan. Systemt: an algebraic approach to declarative information extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 128–137. Association for Computational Linguistics, 2010.

[6] M. Dredze. Email research. `https://sites.google.com/site/emailresearchorg/bibliography`, 2016. Accessed: 01/05/2017.

[7] M. Dredze, T. Lau, and N. Kushmerick. Automatically classifying emails into activities. In *Proceedings of the 11th international conference on Intelligent user interfaces*, pages 70–77. ACM, 2006.

[8] M. Dredze, H. M. Wallach, D. Puller, T. Brooks, J. Carroll, J. Magarick, J. Blitzer, F. Pereira, et al. Intelligent email: Aiding users with ai. In *AAAI*, pages 1524–1527, 2008.

[9] Flow. `https://www.getflow.com`, 2016. Accessed: 01/05/2017.

[10] T. Garsiel and P. Irish. How browsers work: Behind the scenes of modern web browsers. *Google Project, August*, 2011.

[11] A. K. Kalia, H. R. Motahari-Nezhad, C. Bartolini, and M. P. Singh. Monitoring commitments in people-driven service engagements. In *Services Computing (SCC), 2013 IEEE International Conference on*, pages 160–167. IEEE, 2013.

[12] B. Klimt and Y. Yang. Introducing the enron corpus. In *CEAS*, 2004.

[13] N. Kushmerick, T. Lau, M. Dredze, and R. Khoussainov. Activity-centric email: A machine learning approach. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 1634. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[14] A. Lampert, R. Dale, and C. Paris. The nature of requests and commitments in email messages. In *Proceedings of the AAAI Workshop on Enhanced Messaging*, pages 42–47, 2008.

[15] MailPilot. `http://mindsense.co/mailpilot`, 2016. Accessed: 01/05/2017.

[16] L. Màrquez, X. Carreras, K. C. Litkowski, and S. Stevenson. Semantic role labeling: An introduction to the special issue. *Comput. Linguist.*, 34(2):145–159, June 2008.

[17] L. McDowell, O. Etzioni, A. Halevy, and H. Levy. Semantic email. In *Proceedings of the 13th international conference on World Wide Web*, pages 244–254. ACM, 2004.

[18] H. R. M. Nezhad. Cognitive assistance at work. In *AAAI Fall Symposium Series*. AAAI Publications, 2015.

[19] T. Petrocelli. Taming notifications before they run rampant. `http://www.cmswire.com/social-business/taming-notifications-before-they-run-rampant/`, 2015. Accessed: 01/05/2017.

[20] M. Purver, P. Ehlen, and J. Niekrasz. Detecting action items in multi-party meetings: Annotation and initial experiments. In *Machine Learning for Multimodal Interaction*, pages 200–211. Springer, 2006.

[21] A. Qadir, M. Gamon, P. Pantel, and A. H. Awadallah. Activity modeling in email. In *Proceedings of NAACL-HLT*, pages 1452–1462, 2016.

[22] A. Qadir and E. Riloff. Classifying sentences as speech acts in message board posts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 748–758. ACL, 2011.

[23] M. J. Rees. Evolving the browser towards a standard user interface architecture. In *Proceedings of the Third Australasian Conference on User Interfaces - Volume 7*, AUIC '02, pages 1–7, Darlinghurst, Australia, Australia, 2002. Australian Computer Society, Inc.

[24] S. Scerri, G. Gossen, B. Davis, and S. Handschuh. Classifying action items for semantic email. In *LREC*, 2010.

[25] J. R. Searle. *Speech acts: An essay in the philosophy of language*, volume 626. Cambridge university press, 1969.

[26] T. Winograd. Designing a new foundation for design. *Communications of the ACM*, 49(5):71–74, 2006.

[27] Y. Yang, S. Yoo, F. Lin, and I.-C. Moon. Personalized email prioritization based on content and social network analysis. *IEEE Intelligent Systems*, 25(4):12–18, July 2010.