Information Processing and Management 51 (2015) 252–272

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

A reinforcement learning formulation to the complex question answering problem

Yllias Chali [a], Sadid A. Hasan [b,*], Mustapha Mojahid [c]

[a] University of Lethbridge, Alberta, Canada
[b] Philips Research North America, Briarcliff Manor, NY, USA
[c] IRIT, Toulouse, France

ARTICLE INFO

Article history:
Received 23 July 2012
Received in revised form 26 December 2014
Accepted 5 January 2015

Keywords:
Complex question answering
Multi-document summarization
Reinforcement learning
Reward function
User interaction modeling

ABSTRACT

We use extractive multi-document summarization techniques to perform complex question answering and formulate it as a reinforcement learning problem. Given a set of complex questions, a list of relevant documents per question, and the corresponding human generated summaries (i.e. answers to the questions) as training data, the reinforcement learning module iteratively learns a number of feature weights in order to facilitate the automatic generation of summaries i.e. answers to previously unseen complex questions. A reward function is used to measure the similarities between the candidate (machine generated) summary sentences and the abstract summaries. In the training stage, the learner iteratively selects the important document sentences to be included in the candidate summary, analyzes the reward function and updates the related feature weights accordingly. The final weights are used to generate summaries as answers to unseen complex questions in the testing stage. Evaluation results show the effectiveness of our system. We also incorporate user interaction into the reinforcement learner to guide the candidate summary sentence selection process. Experiments reveal the positive impact of the user interaction component on the reinforcement learning framework.

1. Introduction

The increasing demand for access to different types of information available online have led researchers to a renewed interest in a broad range of Information Retrieval (IR) related areas such as question answering, topic detection and tracking, summarization, multimedia retrieval, chemical and biological informatics, text structuring, and text mining. The existing document retrieval systems cannot satisfy the end-users' information need to have more direct access into relevant documents. Question Answering (QA) systems can address this challenge effectively (Strzalkowski & Harabagiu, 2006). The human variant of the QA systems were used effectively over the years. One such system is the popular QA system in Korea, the Korean Naver's Knowledge iN search,[1] which allows users to ask almost any question and get answers from other users (Chali, Joty, & Hasan, 2009). Another widely known QA service is Yahoo! Answers[2] which is a community-driven knowledge market website launched by Yahoo!. As of December 2009, Yahoo! Answers had 200 million users worldwide and more than

* Corresponding author.
E-mail addresses: chali@cs.uleth.ca (Y. Chali), sadid.hasan@philips.com (S.A. Hasan), mustapha.mojahid@irit.fr (M. Mojahid).

[1] http://kin.naver.com/.
[2] http://answers.yahoo.com/.

http://dx.doi.org/10.1016/j.ipm.2015.01.002
0306-4573/

1 billion answers.[3] Furthermore, Google launched a QA system[4] in April 2002 that was based on paid editors. However, the system was closed in December 2006. The main limitation of these QA systems is that they rely on human expertise to help provide the answers. Our goal is to automate this process such that computers can do the same as professional information analysts. This research is a small step towards such an ambitious goal.

QA research can handle different types of questions: fact, list, definition, how, why, etc. Some questions, which we call *simple* questions, are easier to answer. For example, the question: "Who is the prime minister of Canada?" asks for a person's name. This type of question (i.e. factoid) requires small snippets of text as the answer. On the other hand, *complex* questions often require multiple types of information. For example, the question: "How was Japan affected by the earthquake?" suggests that the inquirer is looking for information in the context of a wider perspective. Multi-document summarization techniques can be applied to treat these questions successfully (Chali, Hasan & Joty, 2011; Chali, Joty, et al., 2009; Harabagiu, Lacatusu, & Hickl, 2006).

Multi-document summarization can be used to describe the information of a document collection in a concise manner (Wan, Yang, & Xiao, 2007a). Some web-based systems are already utilizing the potential of this technology. For example, Google News[5] and the Newsblaster[6] system automatically collect, cluster, categorize, and summarize news from several sites on the web, and help users find news of their interest. In the last decade, complex questions have received much attention from both the Question Answering (QA) and Multi-document Summarization (MDS) communities (Carbonell, Harman, Hovy, Maiorano, & Prange, et al., 2000). Typically, complex QA evaluation systems including the 2004 AQUAINT Relationship QA Pilot,[7] the 2005 Text Retrieval Conference (TREC) Relationship QA Task,[8] and the TREC definition[9] return unstructured lists of candidate answers in response to a complex question. The MDS evaluations (including the 2005, 2006 and 2007 Document Understanding Conference (DUC[10])) task systems with returning paragraph-length answers to complex questions that are responsive, relevant, and coherent. Complex question answering in the form of a query-focused multi-document summarization task is useful in the domain of document management and search systems. For example, it can provide personalized news services for different users according to the users' unique information need (Wan et al., 2009). Moreover, users can obtain the news about a single event from different sources in the form of a summary containing multiple perspectives at the same time.

This paper is concerned with automatic answering of complex questions. We define the complex questions as the kind of questions whose answers need to be obtained from pieces of information scattered in different documents. Our experiments and evaluations were mainly influenced by the specific scenario proposed by the DUC (2005–2007) tasks. In fact, DUC proposes a query-focused summarization task whose features have allowed us to simulate our experiments with complex question answering. Hence, the considered complex questions are the type of questions that request information such as an elaboration about a topic, description about an event or entity, illustration about an opinion, and definition or discussion about an aspect or term or procedure. We use an extractive[11] multi-document summarization approach to perform the complex question answering task.

Effective complex question answering can aid to the improvement of the search systems. When a user searches for some information, the traditional search engines usually offer a listing of sources through which the user has to continue navigating until the desired information need is satisfied. Moreover, the available search systems lack a way of measuring the level of user satisfaction which could have been used to enhance the search policy in real time. User satisfaction can be observed effectively by monitoring user actions (e.g., copy-pasting, printing, saving, emailing) after the search results are presented. A user study can reveal the relationship between user satisfaction and retrieval effectiveness (Al-Maskari, Sanderson, & Clough, 2007). Zaragoza, Cambazoglu, and Baeza-Yates (2010) performed a quantitative analysis about what fraction of the web search queries (posed to the current search engines) can lead to satisfactory results. Computation of user satisfaction, as well as improvement to the search policy, is a difficult task to perform in real time. This motivates us to propose a reinforcement learning formulation to the complex question answering task so that the system can learn from user interaction to improve its accuracy according to user's information need.

Formally, the complex question answering problem can be mapped to a reinforcement learning framework as follows: given a set of complex questions, a collection of relevant documents[12] per question, and the corresponding human-generated summaries (i.e. answers to the questions), a reinforcement learning model can be trained to extract the most important sentences to form summaries (Chali, Hasan & Imam, 2011). Our main motivation behind the proposal of the reinforcement learning formulation is in fact to enable learning from human interaction in real time as we believe that the incorporation of user feedback into the learning process can lead to a robust system that produces more accurate summaries to increase the level of user satisfaction. However, for simplicity during the learning stage, we assume that initially there is no actual user interaction

[3] http://yanswersblog.com/index.php/archives/2009/12/14/yahoo-answers-hits-200-million-visitors-worldwide/.
[4] http://answers.google.com/.
[5] http://news.google.com.
[6] http://newsblaster.cs.columbia.edu/.
[7] http://trec.nist.gov/data/qa/add_QAresources/README.relationship.txt.
[8] http://trec.nist.gov/data/qa/2005_qadata/qa.05.guidelines.html.
[9] http://trec.nist.gov/overview.html.
[10] http://duc.nist.gov/.
[11] An approach where a subset of the sentences from the original documents are chosen as the candidate (i.e. machine-generated) summary.
[12] In this paper, we focus more on the summarization aspects to answer complex questions. The information retrieval phase for question answering falls outside the scope of this work. Hence, we assume the given set of documents as relevant for the given questions.

provided to the system rather the importance of a candidate document sentence can be verified by measuring its similarity with the given human-made abstract summary sentences using a reward function. This assumption relies on the intuition that the users are fully satisfied with the abstract summaries. The original document sentences that are mostly similar to the abstract summary sentences are assigned good reward values. In reinforcement learning, the learner is not aware of which actions (*sentence selection* in our case) to take, rather it must discover which actions deliver the most reward by trying them (Sutton & Barto, 1998).

Real-time user interaction can help QA systems evolve by improving their policy automatically as time passes. Toward this end, we treat the complex question answering task as an interactive problem. Supervised learning techniques are alone not adequate for learning from interaction (Sutton & Barto, 1998). These techniques require a huge amount of human-annotated training data and it is often impossible to collect training examples of all desired kinds in which the agent has to act. Instead, a reinforcement learning approach can be used to sense the state of the environment and take suitable actions that affect the state. We assume that a small amount of supervision is provided in the form of a reward function that defines the quality of the executed actions. In the training stage, the reinforcement learner repeatedly defines action sequences, performs the actions, and observes the resulting reward. The learner's goal is to estimate a policy that maximizes the expected future reward (Branavan, Chen, Zettlemoyer, & Barzilay, 2009).

In this paper, we present a reinforcement learning framework for answering complex questions. As noted before, we simplify our formulation by assuming no real time user interaction by considering that the human generated abstract summaries are the gold-standard and the users (if they were involved) are satisfied with them. The proposed system tries to produce automatic summaries that are as close as the abstract summaries. The relationship between these two types of summaries is learned and the final weights are used to output the machine generated summaries for the unseen data. We employ a modified linear, gradient-descent version of Watkins' $Q(\lambda)$ algorithm (Sutton & Barto, 1998) to estimate the parameters of our model. Experiments on the DUC benchmark datasets demonstrate the effectiveness of the reinforcement learning approach. We also extend this work by proposing a model that incorporates user interaction into the reinforcement learner to guide the candidate summary sentence selection process. Evaluation results indicate that the user interaction component further improves the performance of the reinforcement learning framework (Chali, Hasan, & Imam, 2012). The rest of the paper includes related work, our reinforcement learning formulation, feature space, user interaction modeling, experiments with results, and finally, conclusion with some future directions.

2. Related work

We perform the complex question answering task using an extractive multi-document summarization approach within a reinforcement learning setting. Extractive summarization is simpler than abstract summarization as the process involves assigning scores to the given document sentences using some method and then picking the top-ranked sentences for the summary. Although this kind of summary may not be necessarily smooth or fluent, extractive summarization is currently a general practice due to its simplicity (Jezek & Steinberger, 2008). Over the years, various extraction-based techniques have been proposed for generic multi-document summarization. In recent years, researchers have become more interested in query-focused (i.e. topic-biased) summarization. The leading systems in the DUC[13] and TAC[14] tracks focus on the complex question answering task through multi-document summarization.

Other notable extraction-based summarization systems are as follows. Nastase (2008) expands a query by using the encyclopedic knowledge in Wikipedia and introduce a graph to generate the summary. Daumé III and Marcu (2006) present BAYESUM ("Bayesian summarization"), a sentence extraction model for query-focused summarization. On the other hand, Wan, Yang, and Xiao (2007b) propose a manifold-ranking method to make uniform use of sentence-to-sentence and sentence-to-topic relationships whereas the use of multi-modality manifold-ranking algorithm is shown in Wan et al. (2009). Other than these, topic-focused multi-document summarization using an approximate oracle score has been proposed in Conroy, Schlesinger, and O'Leary (2006) based on the probability distribution of unigrams in human summaries. In our proposed approach, we represent each sentence of a document as a vector of feature-values. We incorporate query-related information into our model by measuring the similarity between each sentence and the user query (i.e. the given complex question). We exploit some features such as title match, length, cue word match, named entity match, and sentence position to measure the importance of a sentence. We use a number of features to measure the query-relatedness of a sentence considering n–gram overlap, LCS, WLCS, skip-bigram, exact-word, synonym, hypernym/hyponym, gloss and Basic Element (BE) overlap, and syntactic information (See details in Section 6). These features have been adopted from several related works in the problem domain (Chali, Joty, et al., 2009; Edmundson, 1969; Litvak, Last, & Friedman, 2010; Schilder & Kondadadi, 2008; Sekine & Nobata, 2001). We also considered a dynamic feature to lower the redundancy in the extract summary using the Maximal Marginal Relevance (MMR) model (Carbonell & Goldstein, 1998). This feature helps the learner to understand which document sentence is less similar to the sentences that are already present in the candidate answer space (i.e. the current summary pool). We use the relevant novelty metric that was previously shown in Goldstein, Mittal, Carbonell, and Kantrowitz (2000). This metric measures relevance

[13] http://duc.nist.gov/pubs.html.
[14] http://www.nist.gov/tac/publications/index.html.

and novelty independently and provides a linear combination of them. A document sentence has a higher chance to be selected if it is both relevant to the given query and useful for a summary, while having minimal similarity to the previously selected sentences.

In the field of natural language processing, reinforcement learning has been extensively applied to the problem of dialogue management where the systems converse with a human user by taking actions that emit natural language utterances (Litman, Kearns, Singh, & Walker, 2000; Roy, Pineau, & Thrun, 2000; Scheffler & Young, 2002; Singh, Kearns, Litman, & Walker, 1999). The state space defined in these systems encodes information about the goals of the user and what they say at each time step. The learning problem is to find an optimal policy that maps states to actions through a trial-and-error process of repeated interaction with the user. Branavan et al. (2009) presented a reinforcement learning approach for mapping natural language instructions to sequences of executable actions. Recently, a related problem of automatic text summarization has been modeled using a reinforcement learning framework by Ryang and Abekawa (2012). Our approach is significantly different from their approach in a number of ways. Their formulation is only applicable to generic summarization while our system considers a problem that is a kind of query-focused multi-document summarization. Moreover, the reward function of their model is not designed to take user feedback into account whereas the reward function of our reinforcement learning framework is specially designed for considering user feedback as the principle way of improving the search policy in real time. A more recent work has explored alternate reinforcement learning algorithms, reward functions and feature sets to show promising results for the multi-document summarization task (Rioux, Hasan, & Chali, 2014).

As noted before, in our first experiment, we do not directly interact with a user making the cost of interaction lower. However, experiments in the complex interactive Question Answering (ciQA) task[15] at TREC-2007 demonstrate the significance of user interaction in this domain. The technique of user modeling in an interactive QA system is not new (Hickl & Harabagiu, 2006; Webb & Strzalkowski, 2006). An adaptive, open-domain, personalized, interactive QA system called YourQA[16] is an example of a deployed system where a QA module interacts with a user model and a dialogue interface (Quarteroni & Manandhar, 2009). Motivated by the effect of user interaction shown in the previous studies (Harabagiu, Hickl, Lehmann, & Moldovan, 2005; Lin, Madnani, & Dorr, 2010; Sakai & Masuyama, 2004; Wang, Huber, Papudesi, & Cook, 2003; Wu, Scholer, & Turpin, 2008; Yan, Nie, & Li, 2011), we propose an extension to our reinforcement learning model by incorporating user interaction into the learner and argue that the user interaction component can provide a positive impact in the candidate summary sentence selection process (Chali et al., 2012).

We compare our system with a SVM-based model. In the field of natural language processing, SVMs are applied to text categorization and syntactic dependency structure analysis. These approaches are reported to have achieved higher accuracy than previous approaches (Joachims, 1998). SVMs were also successfully applied to part–of–speech tagging (Giménez & Màrquez, 2003). Single document summarization systems using SVMs demonstrated good performance for both Japanese (Hirao, Isozaki, Maeda, & Matsumoto, 2002) and English documents (Hirao, Sasaki, Isozaki, & Maeda, 2002). Hirao, Suzuki, Isozaki, and Maeda (2003) showed effectiveness of their multiple document summarization system employing SVMs for sentence extraction. A fast query-based multi-document summarizer called *FastSum* used SVM regression[17] to rank the summary sentences where the goal was to estimate the score of a sentence based on a given feature set (Schilder & Kondadadi, 2008). However, our SVM model treats the task as a classification problem where the classifier is trained on data pairs, defined by feature vectors and corresponding class labels. The need for a large amount of data to train a SVM-based system often makes it harder to use in practice. For this reason, we use an unsupervised summarization model to evaluate our proposed reinforcement system. A k-means clustering algorithm is used to build the unsupervised system (Chali, Joty, et al., 2009).

3. Problem formulation

We formulate the complex question answering problem by estimating an action-value function (Sutton & Barto, 1998). We define the value of taking action a in state s under a policy π (denoted $Q^{\pi}(s,a)$) as the expected return starting from s, taking the action a, and thereafter following policy π:

$$Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma_k r_{t+k+1} | s_t = s, a_t = a\right\} \quad (1)$$

Here, E_{π} denotes the expected value given that the agent follows policy π, R_t is the *expected return* that is defined as a function of the reward sequence, $r_{t+1}, r_{t+2}, \ldots$, where r_t is the numerical reward that the agent receives at time step, t. We call Q^{π} the action-value function for policy π. γ stands for the discount factor that determines the importance of future rewards. We try to find out the optimal policy through policy iteration. Once we get the optimal policy (π^*) the agent chooses the actions using the Maximum Expected Utility Principle (Russel & Norvig, 2003). We show our reinforcement

[15] http://www.umiacs.umd.edu/j~immylin/ciqa/.
[16] http://www.cs.york.ac.uk/aig/projects/yourqa/.
[17] Regression tasks tend to estimate the functional dependence of a dependent variable on a set of independent variables.

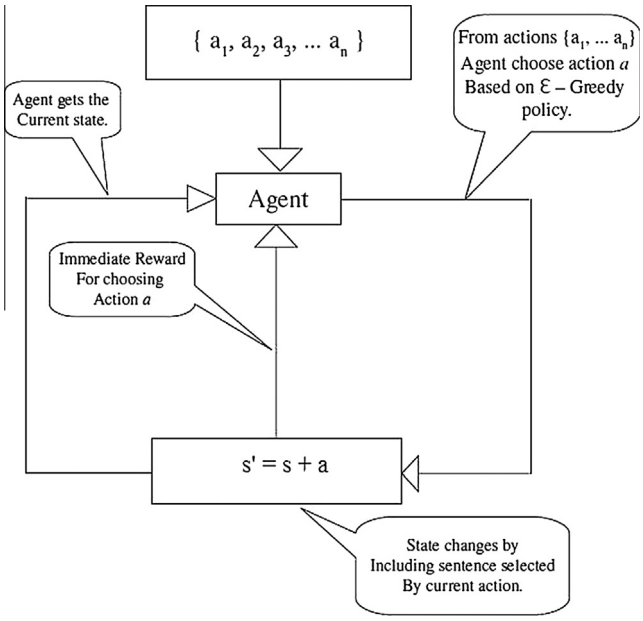

Fig. 1. Reinforcement learning framework.

learning framework in Fig. 1. The figure demonstrates that the agent can choose an action from its action space and performs that action at a certain time. This causes the current state of the agent to change. On the other hand, the agent receives an immediate reward for choosing the action which in turn contributes in updating its policy for determining the next action.

3.1. Environment, state & actions

Given a complex question q and a collection of relevant documents $D = \{d_1, d_2, d_3, \ldots, d_n\}$, the task is to find an answer (extract summary). The *state* is defined by the current status of the answer pool, which represents the sentences that are currently residing in the answer space. Note that, the current state depends on a particular action that is chosen to include a sentence into the answer pool based on a set of features described in Section 6. Initially, there is no sentence in the answer pool, i.e., the initial state s_0 is empty. In each iteration, a sentence from the given document collection is selected and added to the answer pool that in turn changes the *current state*. The environment is described by the state space. In each state, there is a possible set of actions that could be operated on the environment where a certain *action* denotes *selecting a particular sentence* (using the policy function of Eq. (1)) from the remaining document sentences that are not yet included in the answer pool (i.e. candidate extract summary).

3.2. Reward function

In the training stage of the reinforcement learning framework, for each complex question we are given a relevant document collection along with a set of human generated abstract summaries (see details in Section 7.2) as answers to the question. We consider these summaries (i.e. answers) as the gold-standard and assume that the users are satisfied with them. We utilize these summaries to calculate the immediate rewards. In a certain *state*, after taking an action a (i.e. selecting a sentence), we compute the immediate reward, r using the following formula:

$$r = w \times relevance(a) - (1 - w) \times redundancy(a) \tag{2}$$

where $relevance(a)$ is the textual similarity measure between the selected sentence and the abstract summaries, $redundancy(a)$ is the similarity measure between the selected sentence and the *current state* (that includes the already chosen set of sentences) of the answer pool, and w is the weight parameter that denotes the importance of relevance and redundancy. By including redundancy in the immediate reward calculation we discourage redundancy in the final extract summary. In our experiments, the value of w is kept to 0.5 to provide *equal importance* to both relevance and redundancy. We measure the textual similarity using ROUGE (Recall-Oriented Understudy for Gisting Evaluation) (Lin, 2004).

3.3. Function approximation

In many tasks such as the one to which we apply reinforcement learning, most of the states encountered will never have been experienced before. This occurs when the state or action space is very large. As in our case the number of states and actions are infinite, the approximate action-value function is represented as a parameterized functional form with parameter vector, $\vec{\theta}_t$. Our approximate action-value function is a linear function of the parameter vector, $\vec{\theta}_t$. Corresponding to every state-action pair (s, a), there is a column vector of features, $\vec{\varphi}_s = (\varphi_s(1), \varphi_s(2), \ldots, \varphi_s(n))^T$ with the same number of components as $\vec{\theta}_t$. The approximate action-value function is given by:

$$Q_t(s,a) = \vec{\theta}_t^T \vec{\varphi}_s = \sum_{i=1}^{n} \theta_t(i)\varphi_s(i) \tag{3}$$

3.4. Markov Decision Process (MDP)

Our environment has the Markov property, that is, given the current state and action we can predict the next state and expected reward. For our problem formulation, given the current state s if we take an action a, the next state will be $s' = s + a$, since our action is to choose a sentence from the given document collection and adding it into the extract summary pool. Given any state and action, s and a, the transition model is defined by:

$$\rho^a_{ss'} = P_r\{s_{t+1} = s' | s_t = s, a_t = a\} \tag{4}$$

$\rho^a_{ss'}$ will be 1 when $s' = s + a$. For all other states, the transition probability will be 0. Similarly, given any current state and action, s and a, together with any next state, s', the expected value of the next reward is:

$$R^a_{ss'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\} \tag{5}$$

4. Reinforcement learning

We consider our task as an infinite horizon discounted[18] sequential decision making problem that finds a parameter vector $\vec{\theta}$ to maximize $Q(b, a)$ from Eq. (3). Policy gradient algorithms tend to estimate the parameters θ by performing a stochastic gradient ascent. The gradient is approximated by interacting with the environment, and the resulting reward is used to update the estimate of θ. Policy gradient algorithms optimize a non-convex objective and are only guaranteed to find a local optimum (Branavan et al., 2009). We use a modified linear, gradient-descent version of Watkins' $Q(\lambda)$ algorithm with ϵ-greedy policy to determine the best possible action i.e. to select the most important sentences. We use the ϵ-greedy policy (meaning that most of the time this policy chooses an action that has maximal estimated action value, but with probability ϵ an action is selected at random) to balance between exploration and exploitation during the training phase. We empirically set the value of $\epsilon = 0.1$ during our experiments. We note that, 90% of the time our algorithm chooses an action with the best action-value and 10% of the time it chooses an action randomly.

The steps of our reinforcement learning algorithm are shown in Algorithm 1. Here, φ is a vector of feature-values (See details in Section 6) that is used to represent each document sentence and $\vec{\theta}$ is the vector of weights for the feature vector that the system will learn. γ is the discount factor that is used to calculate the reward of a state-action pair. The discount factor determines the importance of future rewards. We kept the initial value of γ as 0.1. The value of γ decreases by a factor of iteration counts. As long as the initial policy selects greedy actions, the algorithm keeps learning the action-value function for the greedy policy. However, when an exploratory action is selected by the behavior policy, the eligibility traces[19] for all state-action pairs are set to zero. The eligibility traces are updated in two steps. In the first step, if an exploratory action is taken, they are set to 0 for all state-action pairs. Otherwise, the eligibility traces for all state-action pairs are decayed by γ^λ. In the second step, the eligibility trace value for the current state-action pair is incremented by 1 while accumulating traces. The original version of the Watkins' $Q(\lambda)$ algorithm uses a linear, gradient-descent function approximation with binary features. However, since we deal with a mixture of real-valued and boolean features (See Section 6), we modified the algorithm to induce a different update for the eligibility traces. In the second step of eligibility trace update, we increment the value by the corresponding feature score. The addition of a random jump step avoids the local maximums in our algorithm. The parameter λ defines how much credit we give to the earlier states. α is the step size parameter for the gradient descent method that is reduced by a factor of 0.99 as learning converges towards the goal.

To the best of our knowledge, the proposed formulation with the modified version of the Watkins' $Q(\lambda)$ algorithm is unique in how it represents the complex question answering task in the reinforcement learning framework.

[18] Although fundamentally the summarization task is a finite horizon problem, we consider an approximation by formulating our task as having an infinite horizon.

[19] An eligibility trace is a temporary record of the occurrence of an event, such as the visiting of a state or the taking of an action (Sutton & Barto, 1998).

Algorithm 1. Modified Watkins' $Q(\lambda)$ algorithm

Input: α, $\vec{\theta}$, $\vec{e}$, λ, γ, δ, φ, ϵ, num of iteration T

Output: A vector $\vec{\theta}$ of learned weights

Initialize: $\vec{\theta}$ to $\vec{0}$, α to 0.01, γ to 0.1, λ to 0.9

$\vec{e} = \vec{0}$

$s, a \leftarrow$ initial state and action of episode

$\varphi \leftarrow$ set of features present in s, a

for *each* $i = 1 \ldots T$ **do**

  if s *is not terminal* **then**

    for $i \in \varphi$ **do**

      $e(i) \leftarrow e(i) + \varphi(i)$

    end

    Take action a, observe reward r, and next state, s

    $\delta \leftarrow r - \sum_i \varphi(i)\theta(i)$

    for $a \in A(s)$ **do**

      $\varphi \leftarrow$ set of features present in s, a

      $Q_a \leftarrow \sum_i \varphi(i)\theta(i)$

    end

    $\delta \leftarrow \delta + \gamma max_a Q_a$

    $\theta \leftarrow \theta + \alpha\delta\vec{e}$

    $\alpha \leftarrow 0.99 * \alpha$

    if *probability* $\leq 1 - \epsilon$ **then**

      for $a \in A(s)$ **do**

        $Q_a \leftarrow \sum_i \varphi(i)\theta(i)$

      end

      $a \leftarrow argmax_a Q_a$

      $\vec{e} \leftarrow \gamma\lambda\vec{e}$

    else

      $a \leftarrow$ a random action $\in A(s)$

      $\vec{e} \leftarrow 0$

    end

  end

end

return $\vec{\theta}$

5. Modeling user interaction

In the basic reinforcement learning model for answering complex questions (discussed in the previous section), we have a set of possible actions in each state. Note that *state* refers to the current status (content) of the answer space while *action* refers to *choosing a candidate sentence*. Initially, there is no sentence in the answer pool. So, the initial state is empty. In each iteration, a new sentence is selected based on the learned Q function from the document collection and added to the answer pool that in turn changes the state. We propose an extension to this model and add user interaction in the reinforcement learning loop in order to facilitate the candidate selection process. For a certain number of iterations during the training stage, the user is presented with the top five candidate sentences based on the learned Q function. The user can also see the complex question being considered and the current status (content) of the answer space (i.e. state). The task of the user at this point is to select the best candidate among the five to be added to the answer space. In the basic reinforcement learning model, the first candidate was selected to be added automatically as it was having the highest similarity score. In this

```
7.04925667663e-07,
0.00120311214399,
0.00207844076797,
0.00231916497637,
0.000581781612264,
0.000600707473091,
0.00232913277055,
0.0,
0.00395316588888,
0.00189000356609,
0.00435198103838,
0.003707038634,
0.00416111283109,
0.00482962916116,
0.00774013468505,
-0.00633759221965,
Human interaction starts:

Topic: home-schooling pros and cons
Question: What are the advantages and disadvantages of home schooling? Is the trend growing or declining?
Current Summary:

Top five candidates:
1:An editorial last Friday in The Wall Street Journal said cases like Ms. Maple 's extend `` beyond the issue of home schooling to the fundamental rights of
families to raise their children the way they see fit . ''
2:Bush 's home schooling message met with applause Friday , but he has departed from conservatives on key education points .
3:With the emergence of home schooling as a viable choice for teen-agers , enrollment has increased almost 50 percent in the last decade , according to James
 Schiefelbein , the Nebraska school 's principal .
4:The California initiative , Proposition 38 , would give parents $ 4,000 for every school-age child that could be spent for any form of currently sanctioned
 education _ including parochial , secular private , charter school , home schooling .
5:`` The state of New Jersey has no laws regarding home schooling , and I would like to keep it that way , '' he said .


Select a sentence to add to current summary: 2
End of human interaction in iteration 1
Human interaction starts:

Topic: home-schooling pros and cons
Question: What are the advantages and disadvantages of home schooling? Is the trend growing or declining?
Current Summary:Bush 's home schooling message met with applause Friday , but he has departed from conservatives on key education points .
```

Fig. 2. User interaction in reinforcement learning.

way, there was a chance that a potentially unimportant sentence could be chosen that is not of user's interest. However, in the extended reinforcement learning model, the user interaction component enables us to incorporate the human viewpoint and thus, the judgment for the best candidate sentence is supposed to be perfect. Extensive experiments on DUC-2006 benchmark datasets support this claim. In Fig. 2, we show an example of how exactly the user interaction component works.

The values at the top of the figure represent the set of initial weights for the considered features. The outcome of the reinforcement learner is a set of weights that are updated through several iterations until the algorithm converges. The currently considered topic is shown next, followed by the complex question, current summary (i.e. answer) and the top five candidate sentences. At this point, the user selects a sentence to add to the answer space and the feature weights are updated based on this response. This process runs for three iterations for each topic during training. In the remaining iterations, the algorithm selects the sentences automatically and continues updating the weights accordingly.

6. Feature space

We represent each sentence of a document as a vector of feature-values. We divide the features into two major categories: static and dynamic. Static features include two types of features, where one declares the importance of a sentence in a document and the other measures the similarity between each sentence and the user query. We use one dynamic feature that measures the similarity of already selected candidate with each remaining sentences. The dynamic feature is used to ensure that there is no redundant information present in the final summary.

6.1. Static features: importance

6.1.1. Position of sentences

Sentences that reside at the start and at the end of a document often tend to include the most valuable information. We manually inspected the given document collection and found that the first and the last 3 sentences of a document often qualify to be considered for this feature. We assign the score 1 to them and 0 to the rest.

6.1.2. Length of sentences

Longer sentences contain more words and have a greater probability of containing valuable information. Therefore, a longer sentence has a better chance of inclusion in a summary. We give the score 1 to a longer sentence and assign the score 0 otherwise. We manually investigated the document collection and set a threshold that a longer sentence should contain at least 11 words. The empirical evidence to support the choice of this threshold is based on the direct observation of our datasets.

6.1.3. Title match

If we find a match such as exact word overlap, synonym overlap or hyponym overlap between the title and a sentence, we give it the score 1, otherwise 0. The overlaps are measured by following the same procedure as described in Sections 6.2.5, 6.2.6, and 6.2.7, respectively. We use the WordNet[20] (Fellbaum, 1998) database for the purpose of accessing synonyms and hyponyms.

6.1.4. Named Entity (NE)

The score 1 is given to a sentence that contains a Named Entity class among: PERSON, LOCATION, ORGANIZATION, GPE (Geo-Political Entity), FACILITY, DATE, MONEY, PERCENT, TIME. We believe that the presence of a Named Entity increases the importance of a sentence. We use the *OAK* System (Sekine, 2002), from New York University for Named Entity recognition.[21] The accuracy of the NE tagger used in the OAK system was reported to be of 72% recall and 80% precision (Sekine & Nobata, 2004).

6.1.5. Cue word match

The probable relevance of a sentence is affected by the presence of pragmatic words such as "significant", "impossible", "in conclusion", "finally" etc. We use a cue word list[22] of 228 words. We give the score 1 to a sentence having any of the cue words and 0 otherwise.

6.2. Static features: query-related

6.2.1. n–gram Overlap

n–gram overlap measures the overlapping word sequences between the candidate document sentence and the query sentence where n stands for the length of the n-gram ($n = 1, 2, 3, 4$). We measured the recall based n–gram scores for a sentence S and a query Q using the following formula (Chali, Joty, et al., 2009):

$$NgramScore(S, Q) = \frac{\sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{gram_n \in S} Count(gram_n)}$$

where n stands for the length of the n-gram ($n = 1, 2, 3, 4$) and $Count_{match}(gram_n)$ is the number of n-grams co-occurring in the query and the candidate sentence.

6.2.2. LCS

Given two sequences S_1 *and* S_2, the longest common subsequence (LCS) of S_1 *and* S_2 is a common subsequence with maximum length. We use this feature to calculate the longest common subsequence between a candidate sentence and the query. We used the LCS-based F-measure to estimate the similarity between a document sentence S of length m and a query sentence Q of length n as follows (Lin, 2004):

$$R_{lcs}(S, Q) = \frac{LCS(S, Q)}{m} \tag{6}$$

$$P_{lcs}(S, Q) = \frac{LCS(S, Q)}{n} \tag{7}$$

$$F_{lcs}(S, Q) = (1 - \alpha) \times P_{lcs}(S, Q) + \alpha \times R_{lcs}(S, Q) \tag{8}$$

where $LCS(S, Q)$ is the length of a longest common subsequence of S and Q and α is a constant that determines the importance of precision and recall. We set the value of α to 0.5 to give equal importance to precision and recall.

6.2.3. WLCS

Weighted Longest Common Subsequence (WLCS) improves the basic LCS method by remembering the length of consecutive matches encountered so far. Given two sentences X and Y, the WLCS score of X and Y can be computed using the similar dynamic programming procedure as stated in Lin (2004). The WLCS-based F-measure between a query and a sentence can be calculated similarly as described in Section 6.2.2.

6.2.4. Skip-bigram

Skip-bigram measures the overlap of skip-bigrams between a candidate sentence and a query sentence. Skip-bigram counts all in-order matching word pairs while LCS only counts one longest common subsequence. The skip bi-gram score between the document sentence S of length m and the query sentence Q of length n can be computed as follows (Chali, Joty, et al., 2009):

[20] WordNet (http://wordnet.princeton.edu/) is a widely used semantic lexicon for the English language. It groups English words (i.e. nouns, verbs, adjectives and adverbs) into sets of synonyms called synsets, provides short, general definitions (i.e. gloss definition), and records the various semantic relations between these synonym sets. We utilize the first-sense synsets. We use the WordNet version 3.0 in this research.

[21] We do not consider coreference resolution in this work.

[22] We constructed the cue word list from a list of transition words available at http://www.smart-words.org/transition-words.html.

$$R_{skip_2}(S,Q) = \frac{SKIP_2(S,Q)}{C(m,2)} \tag{9}$$

$$P_{skip_2}(S,Q) = \frac{SKIP_2(S,Q)}{C(n,2)} \tag{10}$$

$$F_{skip_2}(S,Q) = (1-\alpha) \times P_{skip_2}(S,Q) + \alpha \times R_{skip_2}(S,Q) \tag{11}$$

where $SKIP_2(S,Q)$ is the number of skip bi-gram matches between S and Q and α is a constant that determines the importance of precision and recall. We set the value of α as 0.5 to state equal importance to precision and recall. C is the combination function. We call the Eq. (11) skip bigram-based F-measure.

6.2.5. Exact-word overlap

This is a measure that counts the number of words matching exactly between the candidate sentence and the query sentence. Exact-word overlap can be computed as follows:

$$Exact\ word\ overlap\ score = \frac{\sum_{w_1 \in WordSet} Count_{match}(w_1)}{\sum_{w_1 \in WordSet} Count(w_1)} \tag{12}$$

where WordSet is the set of important[23] words in the sentence and $Count_{match}$ is the number of matches between the WordSet and the important query words.

6.2.6. Synonym overlap

This is the overlap between the list of synonyms of the important words extracted from the candidate sentence and the query related[24] words. We use the WordNet (Fellbaum, 1998) database for this purpose (Chali, Joty, et al., 2009). Synonym overlap can be computed as follows:

$$Synonym\ overlap\ score = \frac{\sum_{w_1 \in SynSet} Count_{match}(w_1)}{\sum_{w_1 \in SynSet} Count(w_1)} \tag{13}$$

where SynSet is the synonym set of the important words in the sentence and $Count_{match}$ is the number of matches between the SynSet and *query related words*.

6.2.7. Hypernym/hyponym overlap

This is the overlap between the list of hypernyms and hyponyms (up to level 2 in WordNet) of the nouns extracted from the sentence and the query related words. This can be computed as follows:

$$Hypernym/hyponym\ overlap\ score = \frac{\sum_{h_1 \in HypSet} Count_{match}(h_1)}{\sum_{h_1 \in HypSet} Count(h_1)} \tag{14}$$

where HypSet is the hyponym/hyponym set of the nouns in the sentence and $Count_{match}$ is the number of matches between the HypSet and *query related words*.

6.2.8. Gloss overlap

Our systems extract the glosses for the proper nouns from WordNet. Gloss overlap is the overlap between the list of important words that are extracted from the glossary definition of the nouns in the candidate sentence and the query related words. This can be computed as follows:

$$Gloss\ overlap\ score = \frac{\sum_{g_1 \in GlossSet} Count_{match}(g_1)}{\sum_{g_1 \in GlossSet} Count(g_1)} \tag{15}$$

Where GlossSet is the set of important words (i.e. nouns, verbs and adjectives) taken from the gloss definition of the nouns in the sentence and $Count_{match}$ is the number of matches between the GlossSet and query related words.

6.2.9. Syntactic feature

The first step to calculate the syntactic similarity between the *query* and the *sentence* is to parse them into syntactic trees using a syntactic parser (Chali, Joty, et al., 2009). We use the Charniak parser[25] (Charniak, 1999) for this purpose. Once we build the syntactic trees, our next task is to measure the similarity between the trees. For this, every tree T is represented by an m dimensional vector $v(T) = (v_1(T), v_2(T), \ldots, v_m(T))$, where the i-th element $v_i(T)$ is the number of occurrences of the i-th tree fragment in tree T. The tree fragments of a tree are all of its sub-trees which include at least one production with the restriction that no production rules can be broken into incomplete parts. The similarity between two syntactic trees can be computed using the *tree kernel* function (Collins & Duffy, 2001). The *TK* (tree kernel) function gives the similarity score between

[23] Henceforth important words are the nouns, verbs, adverbs and adjectives.

[24] To establish the query related words, we took a query and created a set of related queries by replacing its important words by their first-sense synonyms using WordNet.

[25] Available at ftp://ftp.cs.brown.edu/pub/nlparser/.

the *query* and the *document sentence* based on their syntactic structures. The tree kernel of the two syntactic trees, T_1 and T_2 is actually the inner product of the two m-dimensional vectors, $v(T_1)$ and $v(T_2)$ (Moschitti & Basili, 2006; Moschitti, Quarteroni, Basili, & Manandhar, 2007; Zhang & Lee, 2003):

$$TK(T_1, T_2) = v(T_1) \cdot v(T_2)$$

6.2.10. Basic Element (BE) overlap

Basic Elements are defined as follows (Hovy, Lin, Zhou, & Fukumoto, 2006):

- the head of a major syntactic constituent (noun, verb, adjective or adverbial phrases), expressed as a single item, or
- a relation between a head-BE and a single dependent, expressed as a triple: (head|modifier| relation).

We extract BEs for the sentences (or query) by using the BE package distributed by ISI.[26] We compute the Likelihood Ratio (LR) for each BE following Hovy, Lin, and Zhou (2005). We sort the BEs based on LR scores to produce a BE-ranked list. The ranked list contains important BEs at the top which may or may not be relevant to the complex question. We filter out the BEs that are not related to the query and get the BE overlap score (Hovy et al., 2005).

6.3. Dynamic feature

For each sentence that is selected for the summary pool, we measure its similarity with the remaining non-selected sentences using ROUGE. The similarity value is encoded into the feature space of the non-selected sentences as the dynamic feature. The purpose of this feature is to ensure that the next sentence to be chosen into the summary is significantly different from the sentence that is already there. In other words, from the dynamic feature of a sentence we can understand whether the sentence can add any new information into the summary or not. The dynamic feature is updated each time a new sentence is added to the summary. We use the Maximal Marginal Relevance (MMR)[27] method (Carbonell & Goldstein, 1998) to balance this feature with query relevance. We give equal importance to query relevance and redundancy reduction such that the selected sentence can add some valuable as well as new information to the summary. The concept of dynamic similarity feature has been successfully utilized in other relevant applications such as pattern recognition (Liu, Yan, Lu, & Ma, 2006).

7. Evaluation framework and results

7.1. Task overview

Over the past few years, complex questions have been the focus of much attention in both the automatic question–answering and Multi Document Summarization (MDS) communities. Typically, current complex QA evaluations including the 2004 AQUAINT Relationship QA Pilot, the 2005 Text Retrieval Conference (TREC) Relationship QA Task, and the TREC definition (and others) require systems to return unstructured lists of candidate answers in response to a complex question. However, MDS evaluations (including the 2005, 2006 and 2007 Document Understanding Conference (DUC)) have tasked systems with returning paragraph-length answers to complex questions that are responsive, relevant, and coherent. The DUC conference series is run by the National Institute of Standards and Technology (NIST) since 2001 whose aim is to further progress in summarization and enable researchers to participate in large-scale experiments. This paper deals with the query-focused multi-document summarization task as defined in the Document Understanding Conference, DUC-2007. The task is as follows:

"Given a complex question (topic description) and a collection of relevant documents, the task is to synthesize a fluent, well-organized 250-word summary of the documents that answers the question(s) in the topic".

For example, given the topic description (from DUC 2007):

```
<topic>
    <num>D0703A</num>
        <title> steps toward introduction of the Euro </title>
        <narr>
        Describe steps taken and worldwide reaction prior to the introduction of the Euro on January
  1, 1999.
        Include predictions and expectations reported in the press.
        </narr>
</topic>
```

[26] BE website:http://www.isi.edu/cyl/BE.

[27] A sentence has high marginal relevance if it is both relevant to the query and contains minimal similarity to previously selected sentences.

and a collection of relevant documents, the task of the summarizer is to build a summary that answers the question(s) in the topic description. We consider this task[28] and apply a reinforcement approach to generate topic-oriented 250-word extract summaries.

7.2. Corpus for training and testing

The DUC-2006 and DUC-2007 document sets came from the AQUAINT corpus, which is comprised of newswire[29] articles from the Associated Press and New York Times (1998–2000) and Xinhua News Agency (1996–2000). In Table 1, we present the description of the datasets used in our experiments. We use the DUC-2006 data to learn a weight for each of the features (described in Section 6) and then use these weights to produce extract summaries for the document clusters of DUC-2007. We also use the given abstract summaries for each topic as the training data. In DUC-2006, each topic (including a complex question) and its document cluster were given to 4 different NIST assessors, including the developer of the topic. Each assessor created a 250-word summary of the document cluster that satisfies the information need expressed in the topic. These multiple reference summaries were used in the training stage to calculate the numeric rewards.

7.3. Systems for comparisons

7.3.1. Baseline system

We report the evaluation scores of one baseline system (used in DUC-2007) in each of the tables in order to show the level of improvement our system achieved. The baseline system generates summaries by returning all the leading sentences (up to 250 words) in the ⟨*TEXT*⟩ field of the most recent document(s).

7.3.2. SVM settings

We compare the performance of our reinforcement learning approach with a SVM-based technique to answer complex questions. A support vector based approach requires a huge amount of training data during the learning stage. Here, typically, the training data includes a collection of sentences where each sentence is represented as a combination of a feature vector and corresponding class label (+1 or −1). We use the same corpus (Section 7.2) for training and testing during the SVM experiments. We generate a training data set by automatically annotating (using ROUGE similarity measures) 50% of the sentences of each document cluster as positive and the rest as negative. The choice of 50% is based on the fact that SVM can learn well from a balanced (equal proportion of positives and negatives) set of examples (Chali & Hasan, 2012). The annotation follows a similar procedure to Chali, Hasan, and Joty (2009). The same feature set (Section 6) is used to represent the document sentences as feature vectors except the dynamic feature. The dynamic feature seemed to be inappropriate for the SVM setting. However, to reduce the redundancy in the system-generated summaries, we use the MMR-approach during the summary generation process.

During the training step, we used the third-order polynomial kernel with the default value of C.[30] We used the SVM^{light}[31] (Joachims, 1999) package. We performed the SVM training experiments using WestGrid[32] to mitigate computation time. We used the *Cortex* cluster which is comprised of shared-memory computers for large serial jobs or demanding parallel jobs.

In the multi-document summarization task at DUC-2007, the required summary length was 250 words. In our SVM setup, we used $g(x)$, the normalized distance from the hyperplane to x to rank the sentences (Chali, Hasan, et al., 2009). Then, we chose the top N sentences as the candidate summary sentences. Initially, the top-ranked sentence is added to the summary and then we perform an MMR-based computation to select the next sentence that is equally valuable as well as new (balancing the query relevance and redundancy factor). We continue this task until the summary length of 250-words is reached.

7.3.3. K-means clustering

The k-means algorithm follows a simple way to cluster a given data set through a pre-specified number of clusters k. There are several approaches (such as "iK-Means" by Mirkin (2005), Hartigan's method (Hartigan & Wong, 1979) etc.) to estimate the number of clusters. These methods may also give incorrect number of clusters. However, in our task, we simply assume that we have two clusters: 1. *Query-relevant cluster* that contains the sentences which are relevant to the user-questions, and 2. *Query-irrelevant cluster* that contains the sentences that are not relevant to the user-questions.

The k-means algorithm defines clusters by the center of mass of their members (Manning & Schutze, 2000). According to Chali, Joty, et al. (2009), we start with a set of initial cluster centers that are chosen randomly and go through several iterations of assigning each object to the cluster whose center is the closest. After all objects have been assigned, we recompute

[28] For simplicity, our system does not attempt to address fluency in this research.

[29] Although we perform experiments on the newswire articles, we speculate that our feature space is also capable to take other types of datasets like *Yahoo! Answers* into consideration.

[30] C is the trade-off parameter of SVM. We experimented with different kernels and found that the third-order polynomial kernel with the default value of C performs best.

[31] http://svmlight.joachims.org/.

[32] http://westgrid.ca/.

Table 1
Description of the datasets.

Characteristics	Training data	Testing data
Conference	DUC-2006	DUC-2007
Number of clusters	50	25
Number of topics with associated complex questions	50	25
Number of documents per clusters	25	25

the center of each cluster as the centroid or mean ($\boldsymbol{\mu}$) of its members. We use the squared Euclidean distance as the distance function. Once we have learned the means of the clusters using the K-means algorithm, our next task is to rank the sentences according to a probability model. A Bayesian model is used for this purpose:

$$P(q_k|\boldsymbol{x},\Theta) = \frac{p(\boldsymbol{x}|q_k,\Theta)P(q_k|\Theta)}{p(\boldsymbol{x}|\Theta)} = \frac{p(\boldsymbol{x}|q_k,\Theta)P(q_k|\Theta)}{\sum_{k=1}^{K} p(\boldsymbol{x}|q_k,\Theta)p(q_k|\Theta)} \tag{16}$$

where q_k is a cluster, $\boldsymbol{x}$ is a feature vector representing a sentence and Θ is the parameter set of all class models. We set the weights of the clusters as equiprobable (i.e. $P(q_k \mid \Theta) = 1/K$). We calculated $p(\boldsymbol{x} \mid q_k, \Theta)$ using the gaussian probability distribution. The gaussian probability density function (pdf) for the d-dimensional random variable $\boldsymbol{x}$ is given by:

$$p_{(\boldsymbol{\mu},\boldsymbol{\Sigma})}(\boldsymbol{x}) = \frac{e^{\frac{-1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}}{\sqrt{2\pi}^d\sqrt{det(\boldsymbol{\Sigma})}} \tag{17}$$

where $\boldsymbol{\mu}$, the mean vector and $\boldsymbol{\Sigma}$, the covariance matrix are the parameters of the gaussian distribution. We get the means ($\boldsymbol{\mu}$) from the K-means algorithm and calculate the covariance matrix using the unbiased covariance estimation procedure (Chali, Joty, et al., 2009):

$$\widehat{\boldsymbol{\Sigma}_j} = \frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T \tag{18}$$

7.4. Evaluation and analysis

7.4.1. Automatic evaluation: ROUGE

Similar to DUC-2006, in DUC-2007, each of the four assessors created a 250-word summary of the document cluster that satisfies the information need expressed in the topic statement. These multiple "reference summaries" were used in the evaluation of our system-generated summary[33] content. We considered the widely used evaluation measures Precision (P), Recall (R) and F-measure for our evaluation task. *Recall* is defined as the ratio of the number of units (sentences/words) of the system-generated summaries in common with the reference summaries to the total number of units in the *reference* summary while *precision* is the ratio of the number of units of system-generated summaries in common with the reference summaries to the total number of units in the *system-generated* summaries. F-measure combines precision and recall into a single measure to compute the overall performance. We evaluate the system generated summaries using the automatic evaluation toolkit ROUGE (Lin, 2004) which has been widely adopted by DUC. ROUGE parameters were set as that of DUC-2007 evaluation setup. We report the scores of the two official ROUGE metrics of DUC, ROUGE-2 (bigram) and ROUGE-SU (skip bigram). All the ROUGE measures are calculated by running ROUGE-1.5.5 with stemming but no removal of stopwords.

ROUGE run-time parameters:
ROUGE-1.5.5.pl -2 -1 -u -r 1000 -t 0 -n 4 -w 1.2 -m -l 250 -a

Tables 2 and 3 show the ROUGE precision and recall scores of the reinforcement system, the supervised SVM system, and the unsupervised k-means system. In Table 4, we compare the ROUGE-F scores of the baseline system, SVM system, k-means system, and reinforcement system. From here, we find that the **reinforcement** system improves the ROUGE-2 and ROUGE-SU scores over the **baseline** system by 32.9% and 21.1%, respectively. On the other hand, the **reinforcement** system outperforms the supervised **SVM** system demonstrating improvements to the ROUGE-2 and ROUGE-SU scores by 28.4% and 2.7%, respectively besides performing very closely to the unsupervised **k-means** system.

Statistical significance: An approximate result to identify which differences in the competing systems' scores are significant can be achieved by comparing the 95% confidence intervals for each mean. In Table 4, we show the 95% confidence intervals of all the systems to report significance for doing meaningful comparison. ROUGE uses a randomized method named bootstrap resampling to compute the confidence intervals. Bootstrap resampling has a long tradition in the field of statistics (Efron & Tibshirani, 1994). We use 1000 sampling points in the bootstrap resampling.

Discussion: The performance gain of the reinforcement learning method was achieved from the reinforcement learning algorithm as its major goal was to learn the optimal policy of selecting the best possible action from the available action space such that the machine generated summary has the closest match with the given gold standard summaries. Two systems can

[33] The summaries are truncated to 250 words by ROUGE if the summary length reaches over the 250 word limit due to the inclusion of a complete sentence.

Table 2
ROUGE measures: precision.

Systems	ROUGE-2	ROUGE-SU
SVM	0.0707	0.1477
K-means	0.1072	0.1742
Reinforcement	0.0878	0.1417

Table 3
ROUGE measures: recall.

Systems	ROUGE-2	ROUGE-SU
SVM	0.0641	0.1209
K-means	0.0779	0.1348
Reinforcement	0.0849	0.1319

Table 4
Performance comparison: F-score with confidence intervals.

Systems	ROUGE-2	Confidence intervals	ROUGE-SU	Confidence intervals
Baseline	0.0649	0.0608–0.0688	0.1127	0.1084–0.1167
SVM	0.0672	0.0570–0.0787	0.1329	0.1218–0.1444
K-means	0.0902	0.0662–0.0953	0.1520	0.1241–0.1594
Reinforcement	0.0863	0.0740–0.0968	0.1365	0.1236–0.1478

be judged as significantly different if one of the two criteria becomes true: (1) their confidence intervals for the estimates of the means do not overlap at all, or (2) the two intervals overlap but neither contains the best estimate for the mean of the other (Schenker & Gentleman, 2001). Analyzing the reported confidence intervals of different systems from Table 4, we see that the **reinforcement** system significantly outperforms the **baseline** system according to the first criterion. We also see that the confidence intervals of the **SVM**, **reinforcement** and **k-means** systems do overlap. However, according to the second criterion, we find that the **reinforcement** system is significantly better than the **SVM** system in terms of ROUGE-2 scores. On the other hand, there is no significant difference between the **reinforcement** system and the **k-means** system if we interpret the confidence intervals by considering the both criteria. Moreover, the inclusion of the dynamic similarity feature into the feature space contributed in minimizing the redundancy of the automatically generated summaries which also provided a positive impact on the system performance. We claim that the performance of the proposed system would further improve if the reward function could consider syntactic and semantic similarities between a selected summary sentence and the abstract summaries. In our experiments, the use of human interaction with the considered SVM and K-means setup was not seemed appropriate. However, the SVM model implicitly receives a kind of feedback from the users as we use an automatic annotation strategy to build the training data set by calculating the textual similarities of the document sentences with the given human generated summaries. Details of this approach are available at Chali and Hasan (2012).

7.4.2. Manual evaluation

It might be possible to get state-of-the-art ROUGE scores although the generated summaries are bad (Sjöbergh, 2007). Therefore, we conduct an extensive manual evaluation in order to analyze the effectiveness of our approach. Two university graduate students judged the summaries for linguistic quality and overall responsiveness according to the DUC-2007 evaluation guidelines. "The given score is an integer between 1 (very poor) and 5 (very good) and is guided by consideration of the following factors: 1. Grammaticality, 2. Non-redundancy, 3. Referential clarity, 4. Focus, and 5. Structure and Coherence. They also assigned a content responsiveness score to each of the automatic summaries. The content score is an integer between 1 (very poor) and 5 (very good) and is based on the amount of information in the summary that helps to satisfy the information need expressed in the topic narrative."[34]

Table 5 presents the average linguistic quality and overall responsive scores of all the systems. From these results, we can see that the reinforcement system does not perform well compared to the baseline system in terms of linguistic quality. This fact is understandable since our approach did not consider any post-processing and sentence-ordering algorithms to fine-tune the system-generated summaries by ignoring the fluency component of the system task. However, in terms of overall content responsiveness, the reinforcement system outperformed all other systems indicating a better accuracy in meeting the user-requested information need. The differences between the systems were computed to be statistically significant[35] at $p < 0.05$ except for the difference between the **k-means** and the **reinforcement** system in terms of linguistic quality.

[34] http://www-nlpir.nist.gov/projects/duc/duc2007/quality-questions.txt.
[35] We tested statistical significance using Student's t-test and a p value less than 0.05 was considered significant.

Table 5
Average linguistic quality and overall responsiveness scores for all systems.

Systems	Linguistic quality	Overall responsiveness
Baseline	4.24	1.86
SVM	3.48	3.20
K-means	3.30	3.45
Reinforcement	3.32	3.80

Table 6
Effective features.

Final weight	Associated feature
0.012837	Basic element overlap
0.007994	Syntactic feature
0.007572	Length of sentences
0.006483	Cue word match
0.005235	Named entity match
0.002201	2–gram overlap
0.002182	Title match
0.001867	Skip–bigram
0.001354	WLCS
0.001282	1–gram overlap

7.4.3. Most effective features for reinforcement learning

After the training phase, we get the final updated weights corresponding to each feature. The smallest weight value indicates that the associated feature can be eliminated because it does not contribute any relevant information for *action* selection. From this viewpoint we can infer that − weights reflect the effectiveness of a certain feature. Table 6 shows the top ten final feature weights (ranked by higher effectiveness) for this problem domain that we find after the training experiment. The table shows that the Basic Element overlap feature is the most effective feature followed by the syntactic feature and the sentence length feature. On the other hand, 1–gram overlap has the lowest weight value denoting the fact that this feature has little impact on the reinforcement learner.

7.4.4. Experiments with user interaction

System Description. The major objective of this experiment is to study the impact of the user interaction component in the reinforcement learning framework. To accomplish this purpose, we use the first 30 topics at most[36] from the DUC-2006 data to learn the weights respective to each feature and then use these weights to produce extract summaries for the next 15 topics (test data).

We follow six different ways of learning the feature weights by varying the amount of user interaction incorporated and the size of the training data: **(1) SYS_0_20**, **(2) SYS_10_20**, **(3) SYS_20_0**, **(4) SYS_20_10**, **(5) SYS_30_0**, and **(6) SYS_30_30**. The numbers in the system titles indicate how many user-interaction and non-user-interaction topics each system included during training, respectively. For example, the first system is trained with 20 topics of the DUC-2006 data without user interaction. Among these systems, the sixth system is different as it is trained with the first 30 topics of the DUC-2006 data without user interaction. The learned weights that are found from the **SYS_30_0** experiment are used as the initial weights of this system. This means that the **SYS_30_30** system is trained with 60 topics in a pseudo-manner (30 topics with interaction from **SYS_30_0** and 30 topics without interaction).

The outcomes of these systems are sets of learned feature weights that are used to generate extract summaries (i.e. answers) for the last 15 topics (test data) of the DUC-2006 data set. After the six learning experiments, we get six sets of learned feature weights which are used to generate six different sets of summaries for the test data (15 topics). We evaluate these six versions of summaries for the same topics and analyze the effect of user interaction in the reinforcement learning framework.

Evaluation. We report the two official ROUGE metrics of DUC-2006 in the results: ROUGE-2 (bigram) and ROUGE-SU (skip bigram). In Table 7, we compare the ROUGE-F scores of all the systems. In our experiments, the only two systems that were trained with 20 topics are **SYS_0_20** and **SYS_20_0** (the one has 20 unsupervised, the other has 20 supervised). From the results, we see that the **SYS_20_0** system improves the ROUGE-2 and ROUGE-SU scores over the **SYS_0_20** system by 0.57%, and 0.47%, respectively. Again, we see that the **SYS_20_10** system improves the ROUGE-2 and ROUGE-SU scores over the **SYS_10_20** system (both systems had 30 topics but where **SYS_20_10** had more human-supervised topics) by 0.96%, and 8.56%, respectively. We also find that the **SYS_30_0** system improves the ROUGE-2 and ROUGE-SU scores over the **SYS_20_10** system (both systems had 30 topics with **SYS_30_0** having more human supervision) by 0.25%, and 0.80%, respectively. The

[36] We build several reinforcement systems by varying the training data size.

Table 7
Performance comparison: F-scores.

Systems	ROUGE-2	ROUGE-SU
SYS_0_20	0.052252	0.118643
SYS_10_20	0.059835	0.114611
SYS_20_0	0.052551	0.119201
SYS_20_10	0.060409	0.124420
SYS_30_0	0.060560	0.125417
SYS_30_30	0.060599	0.125729

Table 8
95% confidence intervals for different systems.

Systems	ROUGE-2	ROUGE-SU
SYS_0_20	0.040795–0.063238	0.110603–0.126898
SYS_10_20	0.046216–0.073412	0.114425–0.134460
SYS_20_0	0.041366–0.063316	0.111324–0.127472
SYS_20_10	0.046718–0.073785	0.114423–0.134463
SYS_30_0	0.046364–0.074779	0.114820–0.134460
SYS_30_30	0.050021–0.075493	0.117726–0.134321

results show a clear trend of improvement when human interaction is incorporated. We can also see that the **SYS_30_30** system is performing the best since it starts learning from the learned weights that are generated from the outcome of the **SYS_30_0** setting. This denotes that the user interaction component has a positive impact on the reinforcement learning framework that further controls the automatic learning process efficiently (after a certain amount of interaction has been incorporated). In Table 8, we report the 95% confidence intervals for ROUGE-2 and ROUGE-SU to show the significance of our results.

We also conduct an extensive manual evaluation of the systems. Two university graduate students judged the summaries for linguistic quality and overall responsiveness according to the DUC-2007 evaluation guidelines. Table 9 presents the average linguistic quality and overall responsive scores of all the systems. Analyzing these results, we can clearly see the positive impact of the user interaction component in the reinforcement learning framework. The improvements in the results are statistically significant ($p < 0.05$).

Discussion. The main goal of the reinforcement learning phase is to learn the appropriate feature weights that can be used in the testing phase. When the agent is in learning mode, in each iteration the weights get updated depending on the immediate reward it receives after selecting an action. To illustrate, when a user is interacting with the system (according to Section 5), in each iteration one sentence is chosen to be included into the summary space. The top five candidates vary based on the previously selected *action*. For each considered topic, the user provides feedback for three consecutive iterations while automatic learning continues in the remaining iterations. When the summary length reaches to 250 words, we obtain the weights learned from one topic. These weights become the initial weights for the next topic. The user again interacts with the system for three iterations and the process continues for a certain number of topics. Then, the agent starts automatic learning for the remaining topics.

The effect of the presence or absence of user feedback on the feature weights can be shown using the graphs in Fig. 3 (**SYS_20_0** experiment) and Fig. 4 (**SYS_0_20** experiment). To understand the feature weight change over iterations, we also present the weights from different stages of the **SYS_20_0** experiment in Fig. 5. Note that the **SYS_20_0** system is trained with 20 topics of the DUC-2006 data with user interaction. We draw this graph using the *gnuplot*[37] graphing utility. To connect all data points smoothly, we used the *"smooth csplines"* option. The labels in the Y-axis of all the figures refer to the features[38] in the following order: (1) 1-gram overlap, (2) 2-gram overlap, (3) LCS, (4) WLCS, (5) exact word overlap, (6) synonym overlap, (7) hypernym/hyponym overlap, (8) sentence length, (9) title match, (10) named entity match, (11) cue word match, (12) syntactic feature, and (13) BE overlap. We also show the weights from different stages of the **SYS_0_20** experiment in Fig. 6. Note that the **SYS_0_20** system is trained with 20 topics of the DUC-2006 data without user interaction. This graph shows how automatic learning affects the weights during the same stages as shown in the previous graph. If we compare the corresponding graphs (with or without user interaction), we find that both the graphs show a similar kind of trend, i.e., at the end of the learning phase (end of topic-20), all the feature weights converge to zero except for 2-gram overlap and BE overlap. However, the main point to notice here is how quickly they converged. From the figures, we can see that the **SYS_20_0** system converged quickly for the important two features (2-gram overlap and BE overlap) by immediately lowering the values in iteration-2. We can also see that the hypernym/hyponym overlap feature gets an abrupt increase in its value during iteration-8 while

[37] http://www.gnuplot.info/.
[38] We include those features that have at least one non-zero weight value in any of the considered stages.

Table 9
Linguistic quality and responsiveness scores.

Systems	Linguistic quality	Overall responsiveness
SYS_0_20	2.92	3.20
SYS_10_20	3.45	3.40
SYS_20_0	3.12	3.39
SYS_20_10	3.50	3.72
SYS_30_0	3.68	3.84
SYS_30_30	3.96	4.10

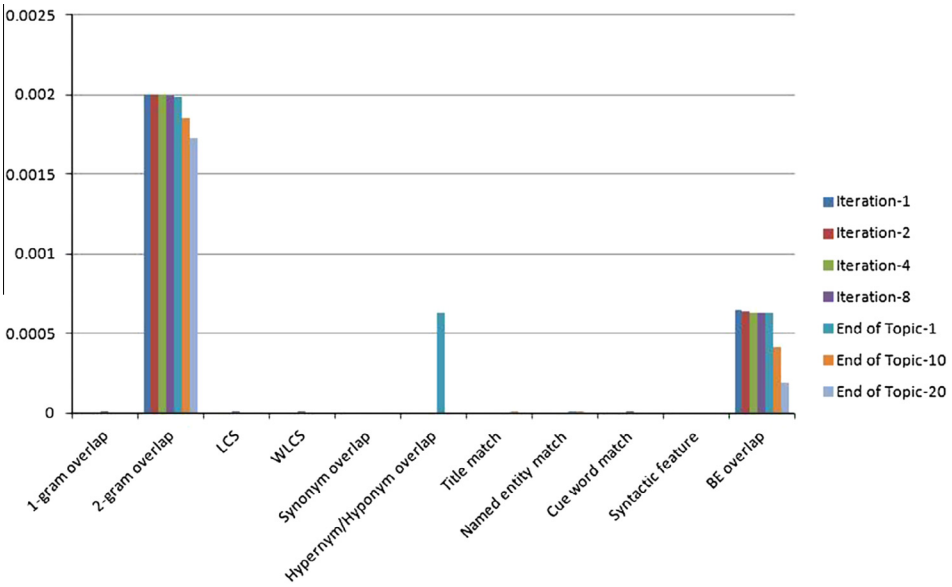

Fig. 3. Effect of user feedback on feature weights.

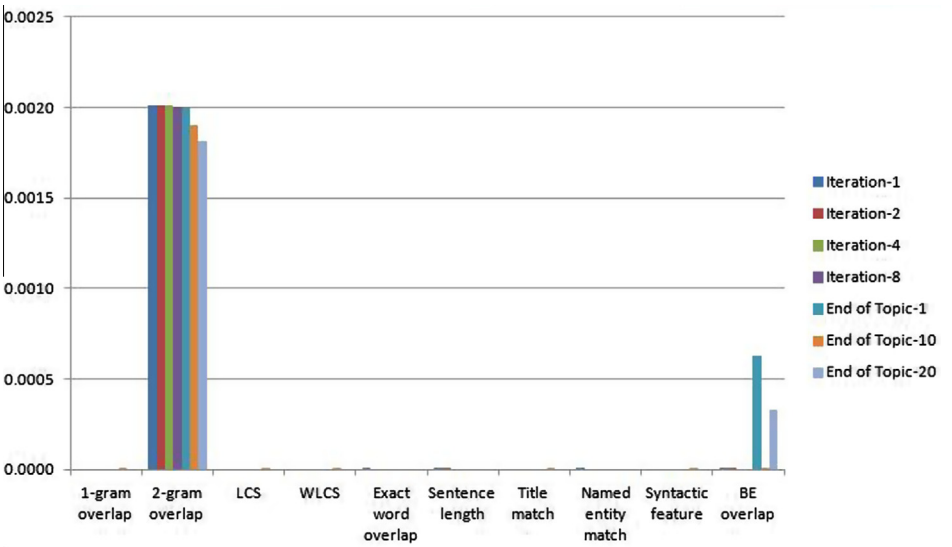

Fig. 4. Effect of fully automatic learning on feature weights.

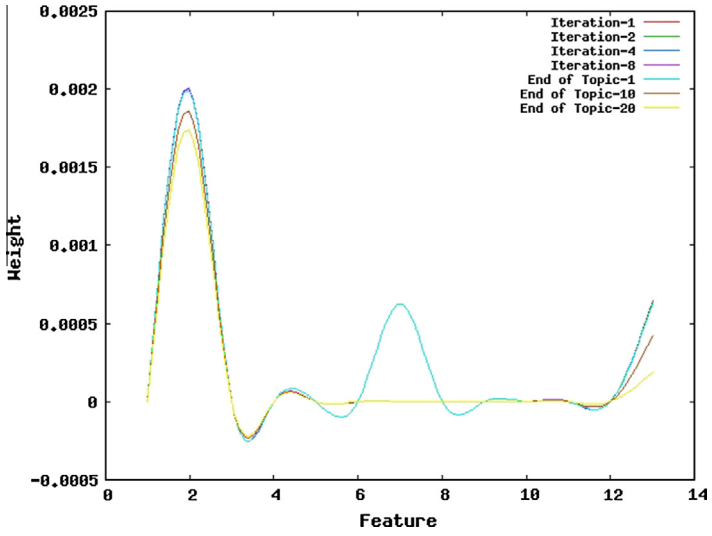

Fig. 5. Feature weight change over iterations (with user interaction).

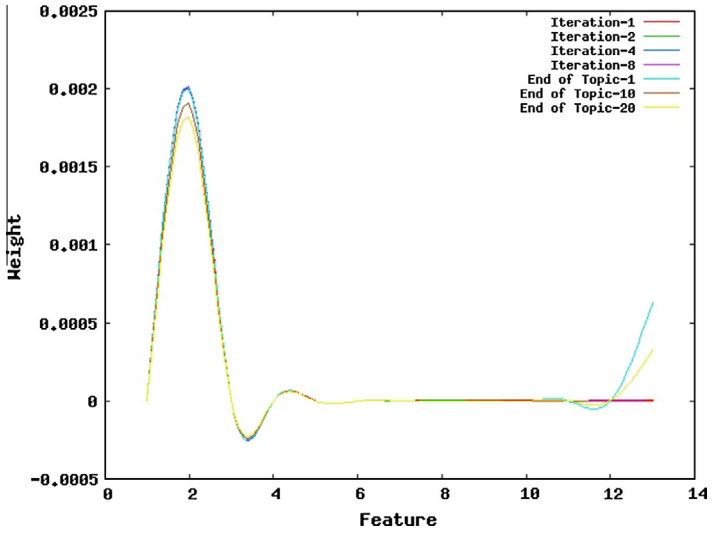

Fig. 6. Feature weight change over iterations (without user interaction).

the value is lowered later. This phenomenon indicates that the reinforcement system is responsive to the user interests and actions. On the other hand, from Figs. 4 and 6 we understand that the **SYS_0_20** system converges slowly by following a fixed pattern. From this investigation, we can conclude that our reinforcement system can learn quickly and effectively from the provided user feedback. In this experiment, we interacted with the system for three iterations for each topic. We claim that the learning performance will improve significantly if more user interaction is provided during the learning phase. The evaluations shown in Section 7.4.4 also support this claim.

8. Conclusion and future work

The main contribution of this paper is a reinforcement learning formulation of the complex question answering problem. We proposed a modified version of the Watkins' $Q(\lambda)$ algorithm to represent the complex question answering task in the reinforcement learning framework. The main motivation of applying a reinforcement approach in this domain was to enhance real-time learning by treating the task as an interactive problem where user feedback can be added as a reward. Initially, we simplified this assumption by not interacting with the users directly. We employed the human-generated abstract summaries to provide a small amount of supervision using reward scores through textual similarity measurement.

Later, we extended our model by incorporating a user interaction component to guide the candidate sentence selection process during the reinforcement learning phase.

We compared our reinforcement system with a baseline, a supervised (SVM) system, and an unsupervised (K-means) system. Extensive evaluations on the DUC benchmark data sets showed the effectiveness of our approach. The performance gain of the reinforcement learning method was achieved from the reinforcement learning algorithm as its major goal was to learn the optimal policy of selecting the best possible action from the available action space such that the machine generated summary has the closest match with the given gold standard summaries. We claim that the performance of the proposed system would further improve if the reward function could consider syntactic and semantic similarities between a selected summary sentence and the abstract summaries. The inclusion of the dynamic similarity feature into the feature space contributed in minimizing the redundancy of the automatically generated summaries which also provided a positive impact on the system performance. Furthermore, our experiments with the user interaction component revealed that the systems trained with user interaction can perform better. The evaluations also showed that the reinforcement system is able to learn automatically (i.e. without interaction) and effectively after a sufficient amount of user interaction is provided as the guide to candidate answer sentence selection.

The implications of our approach for interactive systems can be observed by the user modeling component as it helps the reinforcement framework to refine the answers based on user feedback. Another scenario of adding user interaction could be as follows: a user submits a question to the QA system and receives an extract summary for a list of relevant documents (obtained by a Web search engine) as the answer to the question. The user will then be asked to give a rating about his satisfaction which will be encoded as a reward in the reinforcement learning approach. The current answer to the complex question will be refined accordingly and the same process will be followed until the satisfaction level reaches to the maximum. This process is definitely time consuming, however, once the system receives a considerable amount of feedback about several complex questions, a reinforcement learning system could learn about the user's interests, choices etc. from this data. The learned model can be used to answer unseen complex questions efficiently. We plan to accomplish this goal in the future.

In this research, we kept the value of ϵ static through out the weight learning phase to denote a fixed probability of exploration. In future work, we will experiment on tuning the value of ϵ where we will start with a high value to ensure a greater amount of exploration and less exploitation while gradually decreasing ϵ to reduce exploration as time passes. In this work, we used ROUGE as a reward function to provide feedback to each chosen action. We plan to extend this research by using different textual similarity measurement techniques such as Basic Element (BE) overlap (Hovy et al., 2006), syntactic similarity measure (Moschitti & Basili, 2006), semantic similarity measure (Moschitti et al., 2007), and Extended String Subsequence Kernel (ESSK) (Hirao et al., 2003) as the reward functions.

Acknowledgments

We would like to thank the anonymous reviewers for their useful comments. The research reported in this paper was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada – discovery grant and the University of Lethbridge. This work was done when the second author was at the University of Lethbridge.

References

Al-Maskari, A., Sanderson, M., & Clough, P. (2007). The relationship between IR effectiveness measures and user satisfaction. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '07* (pp. 773–774). ACM.

Branavan, S., Chen, H., Zettlemoyer, L. S., & Barzilay, R. (2009). Reinforcement learning for mapping instructions to actions. In *Proceedings of the joint conference of the 47th annual meeting of the association for computational linguistics (ACL-IJCNLP 2009), Suntec, Singapore* (pp. 329–332).

Carbonell, J., & Goldstein, J. (1998). The Use of MMR, Diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval (SIGIR 1998), Melbourne, Australia* (pp. 335–336).

Carbonell, J., Harman, D., Hovy, E., Maiorano, S., Prange, J., & Sparck-Jones, K. (2000). Vision statement to guide research in question & answering (Q&A) and text summarization. National Institute of Standards and Technology (NIST) Draft Publication.

Chali, Y., Hasan, S. A., & Joty, S. R. (2009). Do automatic annotation techniques have any impact on supervised complex question answering? In *Proceedings of the joint conference of the 47th annual meeting of the association for computational linguistics (ACL-IJCNLP 2009), Suntec, Singapore* (pp. 329–332).

Chali, Y., & Hasan, S. A. (2012). Query-focused multi-document summarization: Automatic data annotations and supervised learning approaches. *Journal of Natural Language Engineering, 18*(1), 109–145.

Chali, Y., Hasan, S. A., & Imam, K. (2011). A reinforcement learning framework for answering complex questions. In *Proceedings of the 16th international conference on intelligent user interfaces, IUI '11* (pp. 307–310). ACM.

Chali, Y., Hasan, S. A., & Imam, K. (2012). Improving the performance of the reinforcement learning model for answering complex questions. In *Proceedings of the 21st ACM conference on information and knowledge management (CIKM 2012)* (pp. 2499–2502). ACM.

Chali, Y., Hasan, S. A., & Joty, S. R. (2011). Improving graph-based random walks for complex question answering using syntactic, shallow semantic and extended string subsequence kernels. *Information Processing and Management (IPM), Special Issue on Question Answering, 47*(6), 843–855.

Chali, Y., Joty, S. R., & Hasan, S. A. (2009). Complex question answering: Unsupervised learning approaches and experiments. *Journal of Artificial Intelligence Research (JAIR), 35*, 1–47.

Charniak, E. (1999). A maximum-entropy-inspired parser. Technical report CS-99-12, Brown University, Computer Science Department.

Collins, M., & Duffy, N. (2001). Convolution kernels for natural language. In *Proceedings of neural information processing systems, Vancouver, Canada* (pp. 625–632).

Conroy, J. M., Schlesinger, J. D., & O'Leary, D. P. (2006). Topic-focused multi-document summarization using an approximate oracle score. In *Proceedings of the COLING/ACL on main conference poster sessions* (pp. 152–159).

Daumé III, H., & Marcu, D. (2006). Bayesian query-focused summarization. In *Proceedings of the 21st international conference on computational linguistics and the 44th annual meeting of the association for computational linguistics* (pp. 305–312).

Edmundson, H. P. (1969). New methods in automatic extracting. *Journal of the Association for Computing Machinery (ACM), 16*(2), 264–285.
Efron, B., & Tibshirani, R. J. (1994). *An introduction to the bootstrap*. CRC Press.
Fellbaum, C. (1998). *WordNet – An electronic lexical database*. Cambridge, MA: MIT Press.
Giménez, J., & Màrquez, L. (2003). Fast and accurate part-of-speech tagging: The SVM approach revisited. In *RANLP* (pp. 153–163).
Goldstein, J., Mittal, V., Carbonell, J., & Kantrowitz, M. (2000). Multi-document summarization by sentence extraction. *Proceedings of the 2000 NAACL-ANLPWorkshop on automatic summarization* (Vol. 4, pp. 40–48). ACL.
Harabagiu, S., Lacatusu, F., & Hickl, A. (2006). Answering complex questions with random walk models. In *Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR 2006)* (pp. 220–227).
Harabagiu, S., Hickl, A., Lehmann, J., & Moldovan, D. (2005). Experiments with interactive question-answering. In *Proceedings of the 43rd annual meeting on association for computational linguistics, ACL '05* (pp. 205–214). ACL.
Hartigan, J. A., & Wong, M. A. (1979). A k-means clustering algorithm. *Applied Statistics, 28*, 100–108.
Hickl, A., & Harabagiu, S. (2006). Enhanced interactive question-answering with conditional random fields. In *Proceedings of the interactive question answering workshop at HLT-NAACL 2006, IQA '06* (pp. 25–32). ACL.
Hirao, T., Isozaki, H., Maeda, E., & Matsumoto, Y. (2002). Extracting important sentences with support vector machines. In *Proceedings of the 19th international conference on computational linguistics, Taipei, Taiwan* (pp. 1–7).
Hirao, T., Sasaki, Y., Isozaki, H., & Maeda, E. (2002). NTT's text summarization system for DUC 2002. In *Proceedings of the document understanding conference, Philadelphia, Pennsylvania, USA* (pp. 104–107).
Hirao, T., Suzuki, J., Isozaki, H., & Maeda, E. (2003). NTT's multiple document summarization system for DUC 2003. In *Proceedings of the document understanding conference, Edmonton, Canada*.
Hovy, E., Lin, C. Y., & Zhou, L. (2005). A BE-based multi-document summarizer with query interpretation. In *Proceedings of the document understanding conference, Vancouver, BC, Canada*.
Hovy, E., Lin, C. Y., Zhou, L., & Fukumoto, J. (2006). Automated summarization evaluation with basic elements. In *Proceedings of the 5th conference on language resources and evaluation, Genoa, Italy*.
Jezek, K., & Steinberger, J. (2008). Automatic text summarization (the state of the art 2007 and new challenges). In *Znalosti* (pp. 1–12), ISBN: 978-80-227-2827-0.
Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning (ECML)*.
Joachims, T. (1999). Making large-scale SVM learning practical. In *Advances in kernel methods – support vector learning*.
Lin, C. (2004). ROUGE: A package for automatic evaluation of summaries. In *Proceedings of workshop on text summarization branches out, post-conference workshop of association for computational linguistics, Barcelona, Spain* (pp. 74–81).
Lin, J., Madnani, N., & Dorr, B. J. (2010). Putting the user in the loop: Interactive maximal marginal relevance for query-focused summarization. In *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics, HLT '10* (pp. 305–308). ACL.
Litman, D. J., Kearns, M. S., Singh, S., & Walker, M. A. (2000). Automatic optimization of dialogue management. In *Proceedings of COLING*.
Litvak, M., Last, M., & Friedman, M. (2010). A new approach to improving multilingual summarization using a genetic algorithm. In *Proceedings of the 48th annual meeting of the association for computational linguistics* (pp. 927–936). ACL.
Liu, Q., Yan, W., Lu, H., & Ma, S. (2006). Occlusion robust face recognition with dynamic similarity features. In *18th International conference on pattern recognition (ICPR)* (pp. 544–547).
Manning, C. D., & Schutze, H. (2000). *Foundations of statistical natural language processing*. The MIT Press.
Mirkin, B. (2005). *Clustering for data mining: A data recovery approach*. CRC, Boca Raton F1: Chapman and Hall.
Moschitti, A., & Basili, R. (2006). A tree kernel approach to question and answer classification in question answering systems. In *Proceedings of the 5th international conference on language resources and evaluation, Genoa, Italy*.
Moschitti, A., Quarteroni, S., Basili, R., & Manandhar, S. (2007). Exploiting syntactic and shallow semantic kernels for question/answer classificaion. In *Proceedings of the 45th annual meeting of the association of computational linguistics (ACL 2007), Prague, Czech Republic* (pp. 776–783).
Nastase, V. (2008). Topic-driven multi-document summarization with encyclopedic knowledge and spreading activation. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP-08)* (pp. 763–772).
Quarteroni, S., & Manandhar, S. (2009). Designing an interactive open-domain question answering system. *Natural Language Engineering, 15*(1), 73–95.
Rioux, C., Hasan, S. A., & Chali, Y. (2014). Fear the REAPER: A system for automatic multi-document summarization with reinforcement learning. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 681–690). Association for Computational Linguistics.
Roy, N., Pineau, J., & Thrun, S. (2000). Spoken dialogue management using probabilistic reasoning. In *Proceedings of ACL*.
Russel, S., & Norvig, P. (2003). *Artificial intelligence a modern approach* (2nd Edition). Prentice Hall.
Ryang, S., & Abekawa, T. (2012). Framework of automatic text summarization using reinforcement learning. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning, EMNLP-CoNLL 2012, Jeju Island, Korea* (pp. 256–265).
Sakai, H., & Masuyama, S. (2004). A multiple-document summarization system with user interaction. In *Proceedings of the 20th international conference on computational linguistics, COLING '04*. ACL.
Scheffler, K., & Young, S. (2002). Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of HLT*.
Schenker, N., & Gentleman, J. (2001). On judging the significance of differences by examining the overlap between confidence intervals. *The American Statistician, 55*, 182–186.
Schilder, F., & Kondadadi, R. (2008). FastSum: Fast and accurate query-based multi-document summarization. In *Proceedings of the 46th annual meeting of the association for computational linguistics on human language technologies: Short papers* (pp. 205–208). ACL.
Sekine, S. (2002). Proteus project OAK system (English sentence analyzer). <http://nlp.nyu.edu/oak>.
Sekine, S., & Nobata, C. A. (2001). Sentence extraction with information extraction technique. In *Proceedings of the document understanding conference (DUC 2001), New Orleans, Louisiana, USA*.
Sekine, S., & Nobata, C. (2004). Definition, dictionaries and tagger for extended named entity hierarchy. In *Proceedings of the forth international conference on language resources and evaluation*.
Singh, S. P., Kearns, M. J., Litman, D. J., & Walker, M. A. (1999). Reinforcement learning for spoken dialogue systems. In *Advances in NIPS*.
Sjöbergh, J. (2007). Older versions of the ROUGEeval summarization evaluation system were easier to fool. *Information Processing and Management, 43*, 1500–1505.
Strzalkowski, T., & Harabagiu, S. (2006). *Advances in open domain question answering*. Springer.
Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, Massachusetts, London, England: The MIT Press.
Wan, X., Yang, J., & Xiao, J. (2007a). Manifold-ranking based topic-focused multi-document summarization. In *Proceedings of the 20th international joint conference on artifical intelligence (IJCAI-07)* (pp. 2903–2908).
Wan, X., Yang, J., & Xiao, J. (2007b). Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proceedings of the joint conference of the 47th annual meeting of the association for computational linguistics (ACL-2007), Prague, Czech Republic* (pp. 552–559).
Wan, X., & Xiao, J. (2009). Graph-based multi-modality learning for topic-focused multi-document summarization. In *Proceedings of the 21st international jont conference on artifical intelligence (IJCAI-09)* (pp. 1586–1591).
Wang, Y., Huber, M., Papudesi, V. N., & Cook, D. J. (2003). User-guided reinforcement learning of robot assistive tasks for an intelligent environment. In *Proceedings of the IEEE/RJS international conference on intelligent robots and systems (IROS)*. Las Vegas, NV: IEEE.
Webb, N., & Strzalkowski, T. e. (2006). *Proceedings of the HLT-NAACL workshop on interactive question answering*. ACL.

Wu, M., Scholer, F., & Turpin, A. (2008). User preference choices for complex question answering. In *Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '08* (pp. 717–718). ACM.

Yan, R., Nie, J., & Li, X. (2011). Summarize what you are interested in: An optimization framework for interactive personalized summarization. In *Proceedings of the 2011 conference on empirical methods in natural language processing* (pp. 1342–1351). Edinburgh, Scotland, UK: ACL.

Zaragoza, H., Cambazoglu, B. B., & Baeza-Yates, R. (2010). Web search solved?: All result rankings the same? In *Proceedings of the 19th ACM international conference on information and knowledge management, CIKM '10* (pp. 529–538). ACM.

Zhang, A., & Lee, W. (2003). Question classification using support vector machines. In *Proceedings of the special interest group on information retrieval* (pp. 26–32). Toronto, Canada: ACM.