A Reinforcement Learning Framework for Answering Complex Questions

Yllias Chali
University of Lethbridge
4401 University Drive W,
Lethbridge, AB, Canada
chali@cs.uleth.ca

Sadid A. Hasan
University of Lethbridge
4401 University Drive W,
Lethbridge, AB, Canada
hasan@cs.uleth.ca

Kaisar Imam
University of Lethbridge
4401 University Drive W,
Lethbridge, AB, Canada
imam@uleth.ca

ABSTRACT

Scoring sentences in documents given abstract summaries created by humans is important in extractive multi-document summarization. In this paper, we use extractive multi-document summarization techniques to perform complex question answering and formulate it as a reinforcement learning problem. We use a reward function that measures the relatedness of the candidate (machine generated) summary sentences with abstract summaries. In the training stage, the learner iteratively selects original document sentences to be included in the candidate summary, analyzes the reward function and updates the related feature weights accordingly. The final weights found in this phase are used to generate summaries as answers to complex questions given unseen test data. 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. selecting the most important sentences. We compare the performance of this system with a Support Vector Machine (SVM) based system. Evaluation results show that the reinforcement method advances the SVM system improving the ROUGE scores by $\sim 28\%$.

Author Keywords

Complex Question Answering, Multi-document Summarization, Reinforcement Learning, Reward Function, Support Vector Machines

ACM Classification Keywords

H.3.m Information Storage and Retrieval: Miscellaneous; I.2.7 Artificial Intelligence: Natural Language Processing

General Terms

Algorithms, Performance, Experimentation

INTRODUCTION

Given a collection of document sentences and their abstract summaries (created by human), we can make use of a learner that tries to find the most important sentences that can be extracted to produce system generated summaries. The importance of a sentence can be verified by measuring its similarity with the abstract summary sentences using a reward function. The most similar sentences are given good reward values and assigned bad values, otherwise. This is reinforcement learning where the task is to learn what to do — how to map situations to actions — so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them [9]. Although search engines are proved to be adequate as a tool for finding documents on the web, certain limitation exists in what the search engine does with the query. For instance, they provide no significant way of measuring whether a user is satisfied with the answer (search) or not, and hence, it cannot improve its policy dynamically in real time. This becomes a main motivation of applying the reinforcement approach to our domain. We can treat complex question answering as an interactive problem since we think that if real-time user feedback can be provided in terms of reward, the answering systems might evolve substantially by improving automatically as time passes. Supervised learning techniques are not adequate for learning from interaction, moreover, it requires a huge amount of human-annotated training data. In interactive problems, it is often impractical to obtain training examples of desired behaviour that are both correct and representative of all the situations in which the agent has to act. So, the strategy here is to use a reinforcement approach that can sense the state of the environment to some extent and is able to take actions that affect the state. We assume that a little amount of supervision is provided in the form of a reward function that defines the quality of executed actions. During training, the learner repeatedly constructs action sequences for a set of given documents, executes those actions, and observes the resulting reward. The learner's goal is to estimate a policy that maximizes future expected reward [1]. In this paper, we present a novel reinforcement learning framework for answering complex questions using an extractive multi-document summarization approach. We simplify our formulation by assuming no real time user interaction. We treat the fact that the human generated abstract summaries are the gold-standards and users (if they were involved) are satisfied with these summaries. Thus, our approach tries to produce automatic summaries that are as close as the abstract summaries. Then, the correspondence between these two types of summaries is learned and the final weights are used to output machine generated

IUI'11, February 13–16, 2011, Palo Alto, California, USA.

summaries from the unseen data. We employ a modified linear, gradient-descent version of Watkins' $Q(\lambda)$ algorithm [9] to estimate the parameters of our proposed model. Extensive experiments demonstrate that our reinforcement method can outperform the well-known supervised learning technique — Support Vector Machines (SVM). The rest of the paper includes the problem formulation in terms of reinforcement learning, feature space, evaluation framework with results, and finally, the conclusion.

PROBLEM FORMULATION

Almost all reinforcement learning algorithms are based on estimating value functions — functions of states (or of state-action pairs) that estimate how good it is for the agent to be in a given state [9]. We formulate the complex question answering problem by estimating an action-value function. 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:

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

Here, E_{π} denotes the expected value given that the agent follows policy π and t is any time step. 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 [7].

Environment, State & Actions

For complex question answering problem we are given a complex question q and a collection of documents $D = \{d_1, d_2, d_3, \ldots, d_n\}$, we have to find out an answer (extract summary). The state is defined by the current status of the answer space. Initially, there is no sentence in the answer pool. So, the initial state s_0 is empty. In each iteration, we add a sentence from the document to the answer pool that in turn changes the state. In each state we have a set of actions. Actions are defined by selecting a sentence from the remaining document sentences that are not included so far in the extract summary.

Reward Function

During training for each complex question, we had abstract summaries generated by human as answers. After taking each action a, we computed the immediate reward, r using the function: $r = relevance(a) - 0.5 * redundancy(a)$. Formally, $relevance(a)$ is the textual similarity measure between the selected sentence and the abstract summaries, whereas $redundancy(a)$ is the similarity measure between the selected sentence and the current state (set of sentences already chosen). By including redundancy in the immediate reward calculation we discourage redundancy in the final extract summary. We measure the textual similarity using ROUGE [6].

Function Approximation

In many tasks such as the one to which we apply reinforcement learning, most states encountered will never have been experienced exactly before. This will almost always be the case when the state or action spaces include continuous variables or complex sensations. 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: $Q_t(s, a) = \vec{\theta_t^T}\vec{\varphi_s} = \sum_{i=1}^{n} \theta_t(i)\varphi_s(i)$

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 next 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 document collection and adding it into the extract summary pool. Given any state and action, s and a, the transition model is defined by: $\rho^{a}_{ss'} = P_r\left\{s_{t+1} = s^{'} | s_t = s, a_t = a\right\}$

$\rho^{a}_{ss'}$ 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:

$$R^{a}_{ss'} = E\left\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s^{'}\right\} \quad (2)$$

We viewed our problem as an infinite horizon sequential decision making problem. For calculating the reward of a state-action pair, we used the discount factor γ. We kept the initial value of γ as 0.1. The value of γ decreases with the increase of iteration counts.

Reinforcement Learning

Our reinforcement learning problem finds the parameter vector $\vec{\theta}$ that maximize $Q(s, a)$. 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 [1]. 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. As long as the initial policy selects greedy

actions, the algorithm keeps learning the action-value function for the greedy policy. But when an exploratory action is selected by the behavior policy, the eligibility traces [9] for all state-action pairs are set to zero. In particular, these are updated in two steps. If an exploratory action is taken, they are set to zero for all state-action pairs. Otherwise, the eligibility traces for all state-action pairs are decayed by $\gamma\lambda$. 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, 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. We reduced the step size, α by 0.99 as learning converges toward the goal. Thus, our formulation with its modified version is unique in how it represents the complex question answering task in the reinforcement learning framework. It is to be noted that, in some respects, our task is more easily amenable to reinforcement learning since we do not directly interact with a human user making the cost of interaction even lower. We address the complexity of our underlying environment and the state space by developing a modified linear, gradient-descent algorithm that learns efficiently while exploring a small subset of the states with ϵ-greedy policy.

FEATURE SPACE

We represent each sentence of a document as a vector of feature-values (φ). 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 [3, 4, 8]. We use a total of eighteen features as static features. To measure importance of a sentence, we consider its position, length, and match with title, certain named entity and cue words. For query-related features, we consider n–gram overlap, LCS, WLCS, skip-bigram, exact-word, synonym, hypernym/hyponym, gloss and Basic Element (BE) overlap, and syntactic features. To lower redundancy in the extract summary, for each sentence that is selected we measure its similarity with the remaining non-selected sentences using ROUGE. We use the Maximal Marginal Relevance (MMR) method [2] to balance this feature with query relevance.

EVALUATION FRAMEWORK AND RESULTS

Task Overview

This paper deals with the query-focused multi-document summarization task as defined in the Document Understanding Conference, DUC-2007. The task is defined as *"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"*. We use 50 topics of DUC-2006 data to learn the weights respective to each feature and then use these weights to produce extract summaries for the first 25 topics (subset of the given 45 topics) of the DUC-2007 data.

System Settings

Reinforcement learning requires a trade off between exploration and exploitation. We used ϵ-greedy policy to balance between exploration and exploitation during the training phase. We set $\epsilon = 0.1$. So, our algorithm chooses an action with the best action-value 90% times and for 10% time it chooses an action randomly. 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 $\langle TEXT \rangle$ field of the most recent document(s). We compare the performance of our reinforcement learning approach with a SVM-based technique to answer complex questions. A Support Vector based approach requires 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 obtain a training data set by automatically annotating (using only ROUGE similarity measures) 50% sentences of each document set as positive and the rest as negative. During training step, we used the third-order polynomial kernel keeping the value of the trade-off parameter C as default. We used the SVM^{light} [5] package. We performed the SVM training experiments in the WestGrid (http://westgrid.ca/) for faster computation. We used the *Cortex* cluster which comprises some shared-memory computers for large serial jobs or demanding parallel jobs. Forcing summaries to obey a certain length constraint is a common set-up in summarization as in the multi-document summarization task at DUC-2007, the word limit was 250 words. In SVM systems, we used $g(x)$, the normalized distance from the hyperplane to x to rank the sentences. Then, we chose the top N sentences until the summary length is reached.

Results and Analysis

The multiple "reference summaries" given in DUC-2007 are used in the evaluation of our summary content. We evaluate the system generated summaries using the automatic evaluation toolkit ROUGE [6] which has been widely adopted by DUC. In Table 1, we show the ROUGE-F scores of the baseline system, SVM system and reinforcement system. 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 advances the SVM system improving the ROUGE-2 and ROUGE-SU scores by 28.4% and 2.7%, respectively. In table 2 and table 3, we report the 95% confidence intervals for ROUGE-2 and ROUGE-SU to show significance for meaningful comparison.

Most Effective Features

After the training phase, we get the final updated weights corresponding to each feature. A weight value close to zero indicates that the associated feature to this weight can be

Systems	ROUGE-2	ROUGE-SU
Baseline	0.0649	0.1127
SVM	0.0672	0.1329
Reinforcement	0.0863	0.1365

Table 1. Performance comparison: F-Score

Systems	ROUGE-2
Baseline	0.060870 - 0.068840
SVM	0.057032 - 0.078794
Reinforcement	0.074092 - 0.096803

Table 2. 95% confidence intervals: ROUGE-2

Systems	ROUGE-SU
Baseline	0.108470 - 0.116720
SVM	0.121819 - 0.144470
Reinforcement	0.123609 - 0.147870

Table 3. 95% confidence intervals: ROUGE-SU

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

Table 4. Effective features

eliminated because it does not contribute any relevant information for action selection. So, from this viewpoint we can infer that — weights reflect the effectiveness of a certain feature. Table 4 shows the top ten final feature weights (ranked by higher effectiveness) for this problem domain that we find after the training experiment.

CONCLUSION AND FUTURE WORK

In this paper, we presented a reinforcement formulation of the complex question answering problem. We proposed a modified version of the Watkins' $Q(\lambda)$ algorithm that is unique in how it represents 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. We simplified this assumption by not interacting with the users directly rather we employed the human-generated abstract summaries in order to provide a little amount of supervision using reward scores through textual similarity measurement. We compared the reinforcement system with a baseline and a SVM system. Evaluation results showed the effectiveness of applying the reinforcement approach for answering complex questions. In future, we plan to test several variants of our reinforcement learning approach using different parameterization of the algorithm, features, reward functions, and ROUGE parameters in order to analyze their concrete impact on the domain.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their useful comments. The research reported here was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada - discovery grant and the University of Lethbridge.

REFERENCES

1. S. Branavan, H. Chen, L. S. Zettlemoyer, and R. Barzilay. 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)*, pages 329–332, Suntec, Singapore, 2009.
2. J. Carbonell and J. Goldstein. 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)*, pages 335–336, Melbourne, Australia, 1998.
3. Y. Chali, S. R. Joty, and S. A. Hasan. Complex Question Answering: Unsupervised Learning Approaches and Experiments. *Journal of Artificial Intelligence Research*, 35:1–47, 2009.
4. H. P. Edmundson. New Methods in Automatic Extracting. *Journal of the Association for Computing Machinery (ACM)*, 16(2):264–285, 1969.
5. T. Joachims. Making large-Scale SVM Learning Practical. In *Advances in Kernel Methods - Support Vector Learning*, 1999.
6. C. Lin. 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*, pages 74–81, Barcelona, Spain, 2004.
7. S. Russel and P. Norvig. *Artificial Intelligence A Modern Approach, 2nd Edition*. Prentice Hall, 2003.
8. S. Sekine and C. A. Nobata. Sentence Extraction with Information Extraction Technique. In *Proceedings of the Document Understanding Conference (DUC 2001)*, New Orleans, Louisiana, USA, 2001.
9. R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction* . The MIT Press, Cambridge, Massachusetts, London, England, 1998.