Deep Q-Learning for Event Summarization

Francisco Javier Arceo
fja2114@columbia.edu

Chris Kedzie
kedzie@cs.columbia.edu

November 1, 2016

Abstract

We present a streaming extraction policy for massive document summarization learned using a deep reinforcement learning recurrent neural network architecture. The rewards of our network are evaluated through manually reviewed summaries, which allow us to specify an end-to-end framework to optimize our extractive policy. By using raw text from our articles and mapping words into a higher embedding dimension through our recurrent neural network, we are able to learn a more robust representation than traditional n-gram models. We benchmark our model against random decisions, bag-of-words, and bag-of-bigrams models.[1]

1 Introduction

Crisis informatics is becoming an increasingly popular area of study in machine learning with more recent research focusing on extractive summarization (e.g., [7] and [5]) of multi-document summarization. The approach by [7] has shown that it is possible to select relevant sentences from a massive number of documents on the web to create summaries with meaningful content by adapting classifiers to maximize search policies. These systems operate in a streaming fashion and are capable of evaluating each sentence within each article to decide whether or not to include or ignore the sentence.

Unfortunately, these systems have still been shown to fall short of heuristic algorithms [citation needed], which may be due to inadequate capturing of the rich structure and often idiosyncratic information by traditional n-gram language models.

[1] `https://github.com/franciscojavierarceo/DQN-Event-Summarization`

[1] have shown that natural language processing can be mapped to a higher order dimension to represent more powerful features for various language modeling tasks. These embeddings have proven incredibly powerful on a variety of different natural language processing tasks [citations needed].

In this paper we show that a deep recurrent neural network with a long short term memory (LSTM) is able to successfully learn an extractive summary policy by encoding the action, state, and reward into a Q-Learning model, similar to that of [3]. We show that on a variety of different metrics, our specification is able to reach state-of-the-art performance.

2 Related Work

3 Extractive Streaming Summarization

In the extractive streaming summarization task, we are given as input a query, e.g. a short text description of a topic or an event, and document stream, a time ordered set of sentences relevant to the query. Starting with an initially empty summary, an extractive, streaming summarization algorithm is intended to examine each sentence in order and when new and important (relative to the query) information is identified, add that sentence to the summary.

Implicit to this problem is the notion of system time – the summarization algorithm can only examine sentences that occur in the stream before the current system time. Advancing the system time gives the algorithm access to more sentences, although in practice the stream is sufficiently large enough that choices have to be made about how much history can be kept in memory. For many domains, e.g. crisis informatics, it is preferable for a summarization algorithm to identify important information as early as possible, and so the objective function should penalize a large disparity between the time a piece of information is first available to the algorithm and the system time at which that information is actually added to the summary.

Previous work in this area has either incremented the system time in fixed increments (e.g. an hour) [8, 7] or operated in a fully online setting [2, 6]. In both cases explicitly modeling the current state of the summary, the stream, and their relationship with the query is quite complicated and exhibits non-linear dynamics that are difficult characterize in traditional feature based models.

Additionaly, the structured nature of the sentence selection task (sentence selection is highly dependent on the current summary state) suggests that imitation or reinforcement learning are necessary to obtain parity between training and testing feature distributions.

This leads us to explore deep Q networks (DQN) for two reasons. First, both the representation and mode of interaction between the stream, summary, and query can be learned. Second, the learned Q function (plus random noise) controls the state space that is explored, ensuring more consistency between train and test distributions than for example naive imitation learning (possibly – I'm not totally happy with this sentence).

3.1 Problem Definition

DQN learns by using an a ϵ-greedy search policy to generate a sequence of state, action, next state, reward 4-tuples using the current learned Q-network to evaluate candidate actions. These tuples are sampled from and used to estimate the true Q function.

States In our setup a state $s(x_t, \tilde{y}_t, q)$ is a function of the stream X observed up to the current system time t, the state of the current summary $\tilde{Y}$ at system time t, and the query q. For brevity we will use $s(t, q)$ where the dependence on $x_t, \tilde{Y}_t$ is assumed. $s(t, q)$ is itself three recurrent neural networks, one for encoding the summary, the stream, and the query respectively.

Actions The set of possible actions at each time step is $\mathcal{A} = \{select, skip\}$ where *select* corresponds to adding the current sentence x_t to the summary and incrementing the current system time,

$$\begin{aligned} \tilde{y}_{t+1} &= \tilde{y}_t \cup \{x_t\} \\ t &= t + 1 \end{aligned}$$

or *skip* where only t is incremented without changing the current summary

$$\begin{aligned} \tilde{y}_{t+1} &= \tilde{y}_t \\ t &= t + 1. \end{aligned}$$

Reward The reward for a given action will be measured by relative gain in ROUGE-2 F1 score of the predicted summary $\tilde{y}_t$ measured against a gold standard summary Y.

When only one gold summary reference is used, ROUGE-N Recall is calculated as

$$\text{ROUGE-NR}(\tilde{y}, Y) = \frac{\sum_{g \in \text{ngrams}(Y,N)} \min\left(\text{count}(g, \tilde{y}), \text{count}(g, Y)\right)}{\sum_{g \in \text{ngrams}(Y,N)} \text{count}(g, Y)}$$

where $\text{ngrams}(Y, N)$ returns the set of ngrams of order N in the summary Y and $\text{count}(g, Y)$ is the count of occurences of ngram g in Y.

Similarly, ROUGE-N Precision is calculated as

$$\text{ROUGE-NP}(\tilde{Y}, Y) = \frac{\sum_{g \in \text{ngrams}(Y,N)} \min\Big(\text{count}(g, \tilde{Y}), \text{count}(g, Y)\Big)}{\sum_{g \in \text{ngrams}(\tilde{Y},N)} \text{count}(g, \tilde{Y})}$$

and the F_1 is simply the harmonic mean of the two:

$$\text{ROUGE-NF1}(\tilde{Y}, Y) = \frac{2 \times \text{ROUGE-NP}(\tilde{Y}, Y) \times \text{ROUGE-NR}(\tilde{Y}, Y)}{\text{ROUGE-NP}(\tilde{Y}, Y) + \text{ROUGE-NR}(\tilde{Y}, Y)}$$

The reward r at time t is then:

$$r_t = \text{ROUGE-NF1}(\tilde{y}_t, Y) - \text{ROUGE-NF1}(\tilde{y}_{t-1}, Y)$$

3.2 Architecture

The architecture for our DQN-LSTM is nearly identical to that of [9] with modifications where appropriate. We learn the parameters using stochastic gradient descent with RMSprop [4], where we minimize the loss function below:

$$\mathcal{L}_i(\theta_i) = \text{E}_{\hat{s},\hat{a}}[\big(y_i - Q(\hat{s}, \hat{a}; \theta_i)\big)^2] \tag{1}$$

where

$$y_i = \text{E}_{\hat{s},\hat{a}}[r_i + \gamma \text{max}_{a'} Q(s', a'; \theta_{i-1}) | s', a']. \tag{2}$$

The updates are computed using the gradient update for $L_i(\theta_i)$

$$\Delta L_i(\theta_i) = \text{E}_{\hat{s},\hat{a}}[2\big(y_i - Q(\hat{s}, \hat{a}; \theta_i)\big)\Delta_{\theta_i} Q(\hat{s}, \hat{a}; \theta_i)]. \tag{3}$$

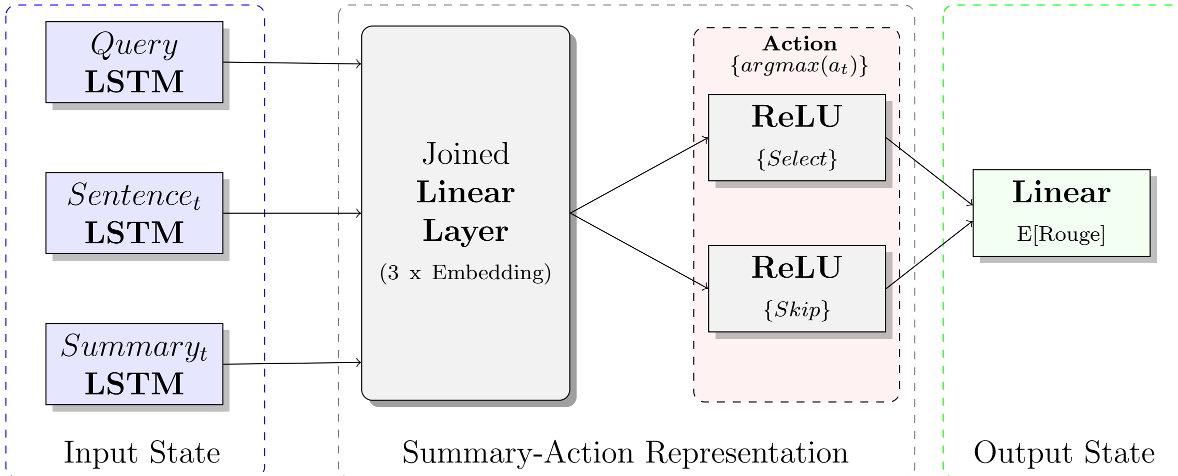

Figure 1: DQN-LSTM Architecture

Metric	Model	Performance
ROUGUE-1F1	DQN-LSTM	X
ROUGUE-1F1	DQN-BOW	X
ROUGUE-1F1	DQN-BiBOW	X
ROUGUE-1F1	Random	X
ROUGUE-2F1	DQN-LSTM	X
ROUGUE-2F1	DQN-BOW	X
ROUGUE-2F1	DQN-BiBOW	X
ROUGUE-2F1	Random	X

Algorithm 1 DQN-LSTM for Event Summarization Training Procedure

Input: $\{\mathcal{Q}$: Event queries, X_q: Input sentences, N: Number of epochs$\}$
Output : $\{\hat{\pi}$: extraction policy, $\tilde{Y}$: event summary for query $q\}$

Initialize memory $\Gamma = \{\emptyset\}_{q=1}^{|\mathcal{Q}|}$
Initialize summaries $\tilde{Y} = \{\emptyset\}_{q=1}^{|\mathcal{Q}|}$
Initialize action-value function $\hat{Q}$ with random weights
for $epoch = 1, .., N$ **do**
  for $q \in \mathcal{Q}$ **do**
    X_q= {Extract $t = 1, ..., T_q$ sentences for query q}
    $\tilde{Y}_q$= {Extract $t = 1, ..., T_q$ summaries for query q}
    for $(x_t, \tilde{y}_t) \in (X_q, \tilde{Y}_q)$, **do**
      Set $s_t = s(x_t, \tilde{y}_t, q)$
      $\forall a_t \in \mathcal{A}(s_t) = \{select, skip\}$ compute $\hat{Q}(s_t, a_t)$
      Select $a_t^* = \text{argmax}_{a_t} \hat{Q}(s_t, a_t)$
      if $random() < \epsilon$ **then**
        Set a_t^* to random action with $\Pr(a_t) = \frac{1}{|\mathcal{A}|}$
      end if
      if $a_t^* = \{select\}$ **then**
        Update $\tilde{y}_{t+1} = \tilde{y}_t \cup \{x_t\}$
      end if
      Execute action a_t^* and observe reward r_t and new state s_{t+1}
      Update $\Gamma_q = \Gamma_q \cup \{[s_t, a_t^*, r_t, s_{t+1}]\}$
    end for
  end for
  for $j = 1, ..., J$ transitions sampled from Γ **do**

$$\text{Set } y_j = \begin{cases} r_j & \text{if } s_{j+1} \text{ is terminal} \\ r_j + \gamma \text{max}_{a'} \hat{Q}(s_{j+1}, a'; \theta) & \text{if } s_{j+1} \text{ is non-terminal} \end{cases}$$

    Perform gradient step on $\mathcal{L}(\theta) = (y_j - \hat{Q}(s_j, a_j; \theta))^2$
  end for
end for

References

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155, 2003.

[2] Qi Guo, Fernando Diaz, and Elad Yom-Tov. Updating users about time critical events. In *European Conference on Information Retrieval*, pages 483–494. Springer, 2013.

[3] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*, 2015.

[4] Geoffrey Hinton, N Srivastava, and Kevin Swersky. Lecture 6a overview of mini–batch gradient descent.

[5] Chris Kedzie. Extractive and abstractive event summarization over streaming web text.

[6] Chris Kedzie, Fernando Diaz, and Kathleen McKeown. Real-time web scale event summarization using sequential decision making. *Proceedings of the Joint Conference on Artificial Intelligence*, 2016.

[7] Chris Kedzie, Kathleen McKeown, and Fernando Diaz. Predicting salient updates for disaster summarization. In *Proceedings of the 53rd annual meeting of the ACL and the 7th International Conference on Natural Language Processing*, pages 1608–1617, 2015.

[8] Richard McCreadie, Craig Macdonald, and Iadh Ounis. Incremental update summarization: Adaptive sentence selection based on prevalence and novelty. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 301–310. ACM, 2014.

[9] Karthik Narasimhan, Tejas D Kulkarni, and Regina Barzilay. Language understanding for textbased games using deep reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Citeseer, 2015.