

DQN-LSTM for Event Summarization

Chris Kedzie

kedzie@cs.columbia.edu

Francisco Javier Arceo

fja2114@columbia.edu

October 10, 2016

Abstract

I will insert something here¹

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 select or skip 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 idiosyncractic information by traditional n-gram models in language modeling. [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 (DQN) with a long short term memory (LSTM) is able to successfully learn an extractive summary policy by encoding the action, state, and reward into our DQN-LSTM similar to that of [4]. We show that on a variety of different metrics, our specification is able to reach state-of-the-art performance.

¹ <https://github.com/franciscojavierarceo/DQN-Event-Summarization>

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 [3, 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 to characterize in traditional feature based models.

Additionally, 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 Algorithm

Algorithm 1 Streaming DQN-LSTM

Input: $\{x_q, \pi_q^*\}_{\forall q \in \mathcal{Q}}$, number of iterations N , mixture parameter ϵ

Output: $\hat{\pi}$

```
1: The notation is not very clear – WORK IN PROGRESS
2: Initialize experience memory  $\Gamma$ 
3: Initialize action-value function  $Q$  with random weights
4:  $i \leftarrow \emptyset$ 
5: for  $n \in \{0, 1, \dots, N - 1\}$  do
6:   for  $q \in \mathcal{Q}$  do
7:     for  $t \in \{0, 1, \dots, T - 1\}$  do
8:       Execute  $\pi_i$   $t$  times and reach  $s_t$ 
9:       for  $a \in \mathcal{A}(s_t)$  do
10:        With probability  $\epsilon$  set  $\pi^0 = \pi_q^*$ , else  $\pi_i$ 
11:        Compute  $c_t(a)$  by executing  $\pi^0$ 
12:         $\Gamma \leftarrow \Gamma \cup \{[\phi(s_t), a, c_t(a)]\}$ 
13:      end for
14:    end for
15:    Sample random minibatches of transitions  $\gamma_j$  from  $\Gamma$ 
16:    Set  $y_j = r_j$  for terminal  $\phi_{j+1}$ 
17:    Perform gradient step on  $(y_j - Q(\phi_j, a_j; \theta))^2$ 
18:     $i \leftarrow i + 1$ 
19:  end for
20: end for
```

Algorithm 2 Streaming DQN-LSTM

Input: $\{x_q, \pi_q^*\}_{\forall q \in \mathcal{Q}}$, number of iterations N , mixture parameter ϵ

Output: $\hat{\pi}$

```
1: The notation is not very clear – WORK IN PROGRESS
2: Initialize  $\pi_0, \Gamma$ 
3: Initialize action-value function  $\mathcal{Q}$  with random weights
4:  $i \leftarrow \emptyset$ 
5: for  $n \in \{0, 1, \dots, N - 1\}$  do
6:   for  $q \in \mathcal{Q}$  do
7:     for  $t \in \{0, 1, \dots, T - 1\}$  do
8:       Execute  $\pi_i$   $t$  times and reach  $s_t$ 
9:       for  $a \in \mathcal{A}(s_t)$  do
10:        With probability  $\epsilon$  set  $\pi^0 = \pi_q^*$ , else  $\pi_i$ 
11:        Compute  $c_t(a)$  by executing  $\pi^0$ 
12:         $\Gamma \leftarrow \Gamma \cup \{[\phi(s_t), a, c_t(a)]\}$ 
13:      end for
14:    end for
15:    Sample random minibatches of transitions  $\gamma_j$  from  $\Gamma$ 
16:    Set  $y_j = r_j$  for terminal  $\phi_{j+1}$ 
17:    Perform gradient step on  $(y_j - Q(\phi_j, a_j; \theta))^2$ 
18:     $i \leftarrow i + 1$ 
19:  end for
20: end for
```

3.2 Problem Definition

DQN learns by using an 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 $s(X_{\leq t}, \tilde{Y}_{\leq t}, q)$ 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 $X_{\leq t}, \tilde{Y}_{\leq t}$ 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,

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

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

$$\begin{aligned}\tilde{Y}_{\leq t+1} &= \tilde{Y}_{\leq t} \\ t &= t + 1.\end{aligned}$$

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

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

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

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 occurrences of ngram g in Y .

Similarly, ROUGE-N Precision is calculated as

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

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

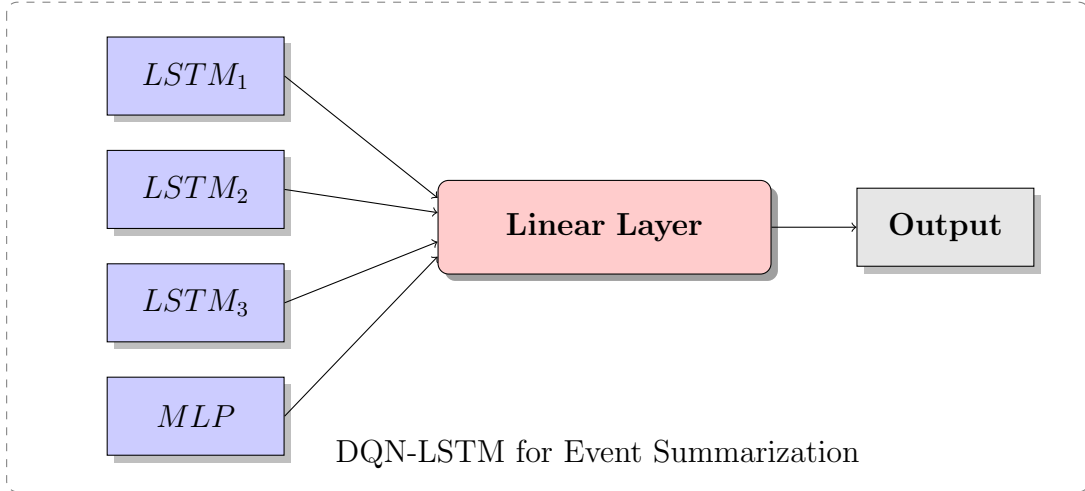
$$\text{ROUGE-NF1}(\tilde{Y}, Y) = \frac{2 \times \text{ROUGE-NP}(\tilde{Y}, Y) \times \text{ROUGE-NR}(\tilde{Y}, Y)}{\text{ROUGE-NP}(\tilde{Y}, Y) + \text{ROUGE-NR}(\tilde{Y}, Y)}$$

The reward r at time t is then:

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

TODO: Think about how to incorporate a time penalty into the reward.

Metric	Model	Performance
ROUGUE-NF1	DQN-LSTM	X
SOMETHING ELSE	DQN-BOW	X



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] Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683*, 2012.

- [3] 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.
- [4] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527*, 2015.
- [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. *arXiv preprint arXiv:1605.03664*, 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] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016.
- [10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.