# Deep Q-Networks for Event Summarization

Francisco Javier Arceo, Chris Kedzie

Columbia University in the City of New York

December 13, 2016

#### **Event Summarization**

Motivation

#### Deep Q-Networks

Review

States

Actions

Rewards

Policy

Algorithm

#### Experiments

Simulation

Benchmarking

#### Resources

## Event Summarization

The primary goal of Event Summarization is to summarize an event over time.

### Event Summarization

The primary goal of Event Summarization is to summarize an event over time.

- ▶ As crises unfold and many articles are generated about a given event, it is beneficial to have a meaningful summary about the incident.
- Given the seemingly countless number of news and media outlets, reading and summarizing all of this content is impractical.
- ▶ Instead, it is useful to have a system that evaluates these articles automatically and returns the most valuable information.



### Extractive Event Summarization

We can structure this problem analytically and find methods to solve Event Summarization (or at least try to).

### Extractive Event Summarization

We can structure this problem analytically and find methods to solve Event Summarization (or at least try to).

- ▶ Extractive streaming summarization takes as input a short text description of a topic or an event known as a query, a document stream, and a time ordered set of sentences relevant to the query.
- ▶ The algorithm sequentially examines each sentence from the document stream and includes the candidate sentence into the summary when novel and important information is detected.

### Previous Work in Extractive Event Summarization

- ▶ Recent research in text retrieval has focused on extractive algorithms to identify important sentences from a large set of documents for summarizing articles from different events in the news (e.g., [1], [4], [2], and [3]).
- ▶ [4] 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 have been shown to fall short of simple heuristic algorithms [2], which may be due to the limited ability of traditional n-gram models to capture the rich and often idiosyncratic structure of language.



# Deep Q-Networks for Event Summarization

This leads us to explore Deep Q-Networks (DQN) for 3 reasons:

- 1. We can define an architecture that propagates information end-to-end about the inputs, actions, and rewards
- 2. The representation and interaction between the stream, summary, and query can be learned jointly
- 3. RNN-LSTM embeddings can learn a more robust semantic representation than classical n-gram models

# Deep Q-Networks: A Brief Introduction

- ▶ Q-Learning is a Reinforcement Learning framework that finds an optimal policy by taking an input state, set of possible actions available, and returning the action with the highest reward.
- ▶ When the state-action space becomes intractable, deterministic algorithms are no longer feasible for an optimal policy and researchers instead train a model to learn the policy.
- ▶ Deep Q-Networks [5] are an increasingly popular framework for learning these policies from end-to-end.



### States

- ▶ A state  $s(x_t, \tilde{y}_t, d)$  at time t is a function of the stream  $X_d$ ,  $\forall d \in D$ , which represents the candidate sentence,  $x_t$  to include in the summary, the state of the current summary  $\tilde{y}_t$ , and the query d.
- ▶ For brevity we use  $s_t$  where the dependence on  $x_t$ ,  $\tilde{y}_t$ , and d is assumed.

#### Actions

We define our set of actions as  $\mathcal{A} := \{select, skip\}$  where select corresponds to adding the current sentence  $x_t$  to the predicted summary  $\tilde{y}_t$  and incrementing the current system time, or skip where only t is incremented without changing the current summary. Simply,

$$\tilde{y}_{t+1} = \begin{cases} \tilde{y}_t \cup \{x_t\}, & \text{if } a_t = select \\ \tilde{y}_t, & \text{if } a_t = skip. \end{cases}$$
(1)

#### Reward

The reward r for a given action a at time t is measured by the change in ROUGE-1F1 score of the predicted summary  $\tilde{y}_t$  measured against a gold standard summary Y.

$$ROUGE-R(\tilde{y}, Y) = \frac{\sum_{g \in Y} \min \left( count(g, \tilde{y}), count(g, Y) \right)}{\sum_{g \in Y} count(g, Y)} \quad (2)$$

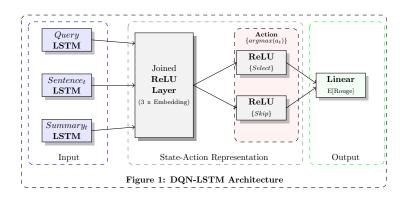
$$\text{ROUGE-P}(\tilde{y}, Y) = \frac{\sum_{g \in \tilde{Y}} \min\left(\text{count}(g, \tilde{y}), \text{count}(g, Y)\right)}{\sum_{g \in \tilde{Y}} \text{count}(g, \tilde{Y})} \quad (3)$$

$$r_t = \text{ROUGE-F1}(\tilde{y}_t, Y) - \text{ROUGE-F1}(\tilde{y}_{t-1}, Y).$$
 (4)

# Policy

- ▶ We define our Q-Learner as an RNN-LSTM that is trained by randomly exploring the state-space using an  $\epsilon$ -greedy search over actions and states.
- ▶ We train our Q-Learner by iteratively updating the learned extraction policy through backpropagation of the observed reward for each action.
- ▶ We define an architecture similar to that of [6] and map our three inputs (query, sentence, and current predicted summary) into LSTM embeddings according to **Figure 1**

## Architecture of the Q-Learner



### DQN-LSTM for Event Summarization Training Procedure

```
Input: \{\mathcal{D}: \text{ Event queries, } X_d: \text{ Input sentences, } N: \text{ Number of epochs}\}
Output: \{\hat{Q}: \text{ extraction policy, } \tilde{Y}_d: \text{ event summary for query } d\}
```

```
1: Initialize extraction policy \hat{Q} with random weights
 2: Initialize memory and summary: \Gamma, \tilde{Y} = \{\emptyset\}_{d=1}^{|\mathcal{D}|}, \{\emptyset\}_{d=1}^{|\mathcal{D}|}
 3: for epoch = 1, ..., N do
 4:
           for query d \in \mathcal{D} do
                \tilde{X}_d, \tilde{Y}_d = \{\text{Extract } t = 1, ..., T_d \ (sentences_d, summary_d)\}
 5:
               for x_t, \tilde{y}_t \in X_d, \tilde{Y}_d do
 6:
                    Set s_t = s(x_t, \tilde{y}_t, d)
 7:
                    \forall a_t \in \mathcal{A}(s_t) \text{ compute } \hat{Q}(s_t, a_t) \text{ and select } a_t^* = \operatorname{argmax}_{a_t} \hat{Q}(s_t, a_t)
 8:
                    if random() < \epsilon then select a_t^* at random with Pr(a_t) = \frac{1}{|A|}
 9:
10:
                     Update \tilde{y}_{t+1} according to equation (1)
11:
                     Execute action a_t^* and observe reward r_t and new state s_{t+1}
12:
                     Update \Gamma_d = \Gamma_d \cup \{[s_t, a_t^*, r_t, s_{t+1}]\}
13:
           for j = 1, ..., J transitions sampled from \Gamma do
14:
```

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

Perform gradient step on  $\mathcal{L}(\theta) = (y_i - \hat{Q}(s_i, a_i; \theta))^2$ 15:

### Small Problems

Before evaluating the model on test data, it is illuminating to understand the in-sample training behavior of the architecture on a small-sample problem. Small sample experimentation allows us to

- 1. Verify that the model is able to solve our specified problem
- 2. Understand the number of iterations required to explore the state-space
- 3. Gain intuition about the impact of hyperparameters on the speed at which it takes to solve the problem

# DQN-LSTM for 1 Query and 20 Sentences: Precision



# DQN-LSTM for 1 Query and 20 Sentences: Recall

# DQN-LSTM for 1 Query and 20 Sentences: F1

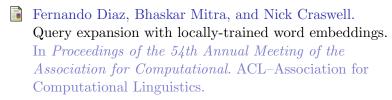
### Evaluation

We benchmark our model against (1) an Oracle that greedily chooses to include sentences if the increase in ROUGUE-F1 is positive, (2) random selection of sentences, and (3) a bag-of-words multilayer perceptron.

## Useful Links below

Thank you

▶ GitHub



Cristina Gârbacea and Evangelos Kanoulas.

The university of amsterdam (ilps. uva) at trec 2015 temporal summarization track.

Chris Kedzie and Kathleen McKeown.
Extractive and abstractive event summarization over streaming web text.

In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016, pages 4002–4003, 2016.



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.

Karthik Narasimhan, Tejas D Kulkarni, and Regina Barzilay.

Language understanding for textbased games using deep reinforcement learning. Event Summarization
Deep Q-Networks
Experiments
Resources

In Proceedings of the Conference on Empirical Methods in Natural Language Processing. Citeseer, 2015.