Deep Q-Networks for Event Summarization

Francisco Javier Arceo
fja2114@columbia.edu

Chris Kedzie
kedzie@cs.columbia.edu

Columbia University in the City of New York

December 13, 2016

Agenda

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.

State, Action, and Rewards

- The state $s(x_t, \tilde{y}_t, d)$ at time t is a function of the candidate sentence, x_t to include in the summary, the state of the current summary $\tilde{y}_t$, and the query d.
- The set of actions as $\mathcal{A} := \{select, skip\}$, which corresponds to

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

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

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,

$$\tilde{y}_{t+1} = \begin{cases} \tilde{y}_t \cup \{x_t\}, & \text{if } a_t = select \\ \tilde{y}_t, & \text{if } a_t = skip. \end{cases} \quad (2)$$

Reward

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

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

$$\text{ROUGE-P}(\tilde{y}, Y) = \frac{\sum_{g \in \tilde{Y}} \min\left(\text{count}(g, \tilde{y}), \text{count}(g, Y)\right)}{\sum_{g \in \tilde{Y}} \text{count}(g, \tilde{Y})} \tag{4}$$

$$r_t = \text{ROUGE-F1}(\tilde{y}_t, Y) - \text{ROUGE-F1}(\tilde{y}_{t-1}, Y). \tag{5}$$

Policy

- We define our Q-Learner as an RNN-LSTM that is trained by randomly exploring the state-space using an ϵ-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

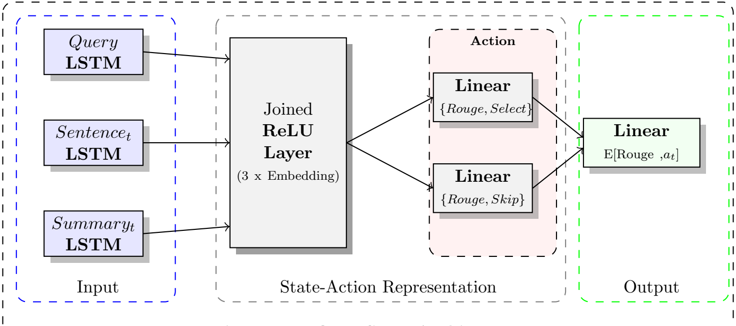
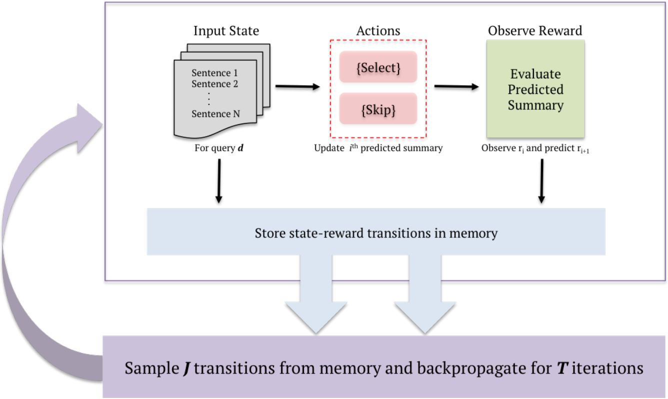


Figure 1: DQN-LSTM Architecture

*ReLU(x) = max(0, x)

Overview of Algorithm

DQN-LSTM for Event Summarization Training Procedure



DQN-LSTM for Event Summarization Training Procedure

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

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

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

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

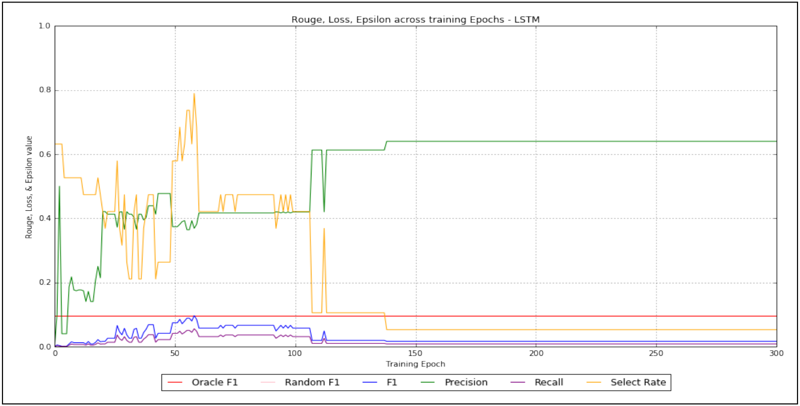
Small Problems

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

Model learned to select the single best sentence.



DQN-LSTM for 1 Query and 20 Sentences: Recall

Model learned to select all sentences.

DQN-LSTM for 1 Query and 20 Sentences: F1

Model was able to maximize F1.

What did we learn?

The experiments are extremely useful in understanding the implications of the specification of the network. We found that

1. Maximizing Precision is easier than maximizing Recall.
2. Choosing either Precision or Recall yields obvious pathological results; i.e., selecting the single best sentence or selecting all sentences.
3. Optimizing F1 requires much longer training to choose the optimal strategy.

References and Links

Thank you

- GitHub Repository

Fernando Diaz, Bhaskar Mitra, and Nick Craswell.
Query expansion with locally-trained word embeddings.
In *Proceedings of the 54th Annual Meeting of the Association for Computational.* ACL–Association for Computational Linguistics.

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.

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.

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.

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