Paywall Policy Learning in Digital News Media

Heidar Davoudi, Zana Rashidi, Aijun An, Morteza Zihayat, and Gordon Edall

Abstract—Subscription-based online newspapers usually offer non-subscribed users a certain number of free articles in a period of time, and then directs them to a page (called paywall) asking for subscription. This approach (also known as *metered* or *fixed* paywall) does not consider the user's reading history nor the articles that the user may read in the future, and consequently, it may disengage many potential subscribers. To that end, we propose adaptive paywall mechanisms to make optimal paywall decisions (i.e., showing the article or the paywall) by balancing the benefit of showing the article against that of presenting the paywall. We define the notions of *utility* and *cost* which are used to define an objective function for the optimal paywall decision problem. We propose the *Lookahead* policy (LAP) and *QPaywall* policy (QP) as two data-driven approaches to solve the adaptive paywall problem. While the LAP method makes paywall decisions *on the fly* by simulating trajectories of article requests using Monte Carlo sampling, the QP approach is based on reinforcement learning and learns a neural network-based action-value (Q) function for this purpose. We compare advantages of the proposed approaches and discuss the practical considerations of using them in a real environment. Empirical studies on a real dataset from a major newspaper in Canada show that the proposed methods outperform several baseline approaches in terms of various business objectives.

Index Terms—Adaptive paywall, digital news media, sequential decision optimization, reinforcement learning

1 Introduction

Advertisement and digital subscription are two main revenue sources for digital news media agencies. While the ad-based model (also known as a *ad-supported free content* model) offers articles for free and generates revenue by displaying advertisements, the subscription-based model offers the reader access to articles through a paid subscription. However, advertisement revenues may not be sufficient to sustain existing forms of news production as they are not stable due to no long-term relationship between an advertiser and its customers. Therefore, the number of online news providers seeking for sustainable relationships with customers is increasing. Currently, 78 percent of U.S. newspapers with circulations over 50,000 are using a digital subscription model [1]. To support digital subscriptions, news agencies (e.g., The New York Times, The Globe and Mail) use a mechanism called *paywall* to request readers to subscribe the newspaper. Traditionally, a paywall model offers a certain number of free articles within a period of time (e.g., a month), and then redirects visitors to the subscription page (i.e., *paywall*) to continue reading articles. The main goal of this mechanism is to persuade users to subscribe the newspaper in order to increase revenue.

- *H. Davoudi is with the Faculty of Science, Ontario Tech University, Oshawa, ON L1G 0C5, Canada. E-mail: heidar.davoudi@uoit.ca.*
- *Z. Rashidi and A. An are with the Department of Electrical Engineering and Computer Science, York University, Toronto, ON M3J 1P3, Canada. E-mail: {zrashidi, aan}@cse.yorku.ca.*
- *M. Zihayat is with the Ted Rogers School of Information Technology Management, Ryerson University, ON M5B 2K3 Toronto, Canada. E-mail: mzihayat@ryerson.ca.*
- *G. Edall is with The Globe and Mail, Toronto, ON M5A 1L1, Canada. E-mail: GEdall@globeandmail.com.*

Manuscript received 1 Feb. 2019; revised 2 Dec. 2019; accepted 11 Jan. 2020. Date of publication 24 Jan. 2020; date of current version 10 Sept. 2021. (Corresponding author: Aijun An.) Recommended for acceptance by Y. Xia. Digital Object Identifier no. 10.1109/TKDE.2020.2969419

However, user acquisition is not easy in the news domain as users usually have many options in selecting news sources. In fact, most of the time, we cannot observe a direct relationship between the number of paywalls presented to readers and the number of subscriptions. In other words, an increase in the number of times that the paywall is presented to readers may not necessarily result in the raise in the number of subscriptions. Moreover, the number of paywalls are much higher than the number of subscriptions (see Fig. 1). In fact, the traditional paywall mechanism, which is based on the total number of articles read in a period of time, is not only misaligned with the goal of increasing revenue, but also it may turn away many potential subscribers. This is due to the fact that blocking a user from reading articles at a wrong time may disengage the user too early while allowing a non-potential subscriber to read too much content for free.

To that end, we raise the following question: *How many and what articles a particular user should be allowed to read before a paywall?* We suggest that the availability of vast amount of users' interaction data stored by online newspapers, and advanced techniques in data mining and machine learning can provide an appropriate data-driven answer to this question. Intuitively, the answer to this question should not be the same for all users as they may have different potentials for subscription. Moreover, the answer should consider business objectives even if they are in conflict with one another. For example, allowing readers to read more articles leads to more display of ads, increasing the ad-based revenue. However, from the subscription point of view, this is not desirable as offering too much free content makes subscription unnecessary from the reader's point of view.

We model the problem of finding the optimal paywall time as a sequential decision-making process (or a sequential decision problem), in which a decision whether to show a paywall needs to be made at each time point during a reading session and once the paywall is presented, the session

1041-4347 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: Ontario Tech University. Downloaded on November 10,2026 at 19:57:42 UTC from IEEE Xplore. Restrictions apply.

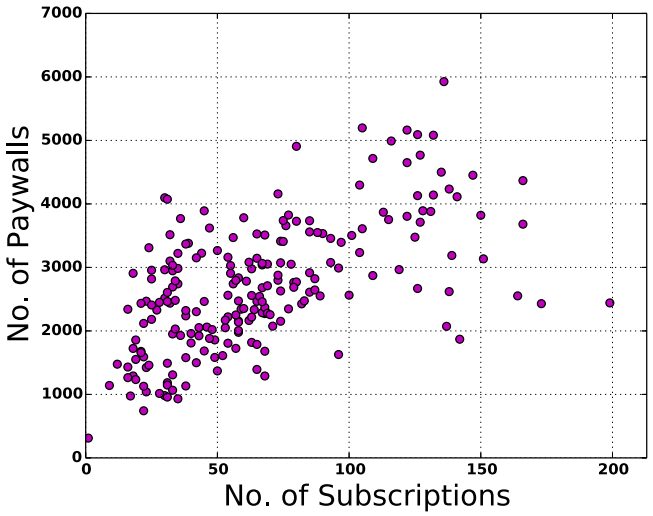


Fig. 1. The number of subscriptions versus the number of paywalls in The Globe and Mail dataset for the period 2014-01 to 2014-07. There is a weak correlation (i.e., $\rho = 0.59$) between the two numbers.

terminates. To define an objective function for this problem, we introduce the notion of *utility*, and *cost*. The utility of an article represents the effectiveness/usefulness of an article in achieving a business objective (e.g., user engagement, which can increase the subscription possibility). The *cost* of an article represents the amount of resources allocated to prepare the article (e.g., money paid to the author). Furthermore, we argue that the behavior of users (e.g., visited articles in a session) in the past is an important factor in the decision-making process. Traditional paywall models (called *fixed paywall* or *metered paywall*) block users from reading articles after visiting a certain number of articles (e.g., two articles) without considering the above or any other factors.

Fig. 2 shows four different scenarios demonstrating how the utility and cost can be used to make smarter decisions. Given a fixed paywall policy with the limit of two articles, all users in this figure would be redirected to the paywall after visiting the second article. Assume that we do not use the fixed paywall but consider what they would like to read next. In the first scenario, $user_1$ clicks on a_8. Since a_8 has a low cost, we can show it to the user and take advantage of other benefits (e.g., displaying ads when showing a_8). However, $user_1$ might not be a good target for subscription as she only reads the articles whose utility is low (e.g., can be found somewhere else), so once she clicks on the next article (i.e., a_4), which has a high cost, the paywall is presented. In the second scenario, $user_2$ can visit article a_3 as its high cost can be justified by its high utility, and then receives the paywall after visiting a_9, where the next article has low utility and high cost (i.e., a_{11}). In the third scenario, $user_3$ visits high utility articles (i.e., a_5, a_7), which may be articles visited by subscribers before their subscription, therefore she is more likely to become a subscriber. In this case, we show the paywall (after a_7) since the next article (e.g., a_8) has a high cost and low utility. Note that this user is more likely to become a subscriber and showing the paywall at the right time might persuade her to subscribe. This example shows that different reading behaviors may require different paywall strategies, and making decisions based on utility and cost model can serve as an effective approach to make a smart decision per user visit.

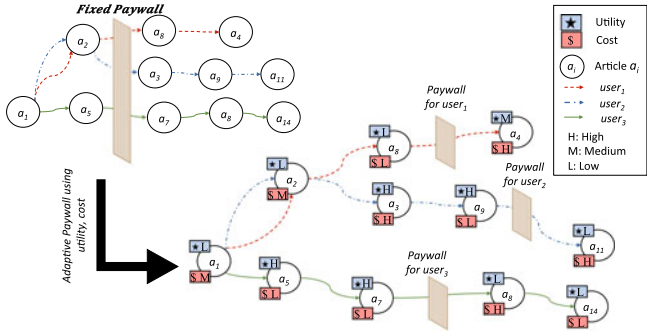


Fig. 2. Incorporating utility, cost and graph structure in paywall mechanisms.

Finding the optimal paywall time for a user is a challenging problem. First, while the concepts of utility and cost provide insights into the paywall decision problem, incorporating them into the optimal decision making process in a disciplined way is not a trivial task. Second, when making a decision at a time point (i.e., when the user clicks on an article), we only know the articles that the user has visited and the next article she is trying to read. The articles beyond the next one are unknown. Thus, when making an optimal decision at an early point, we need to consider the uncertainty as to what would happen later. Third, a proposed model is better to be flexible so that any objective can be plugged in as desired. Last but not least, finding the optimal paywall time is a real time decision making problem, therefore the proposed approach should be efficient to work in an online setting.

To address these challenges, we formulate the problem in a unified stochastic decision making framework by considering the utility and cost of articles. In this paper, we define the main components of the framework and present two effective approaches to solving the problem. The first approach (referred to as *lookahead paywall policy*) was first proposed in [2]. This paper extends [2] and proposes a reinforcement learning based method (referred to as *QPaywall*) to solve the problem. In particular, we use a neural network to learn an action-value function to estimate the value of making a certain decision in a state. Moreover, to facilitate this estimation, we propose a novel state space representation by leveraging article embeddings (i.e., low dimensional representations of articles) derived from a navigational graph. To the best of our knowledge, we are the first to formulate and tackle the paywall problem as an optimal decision problem and propose adaptive solutions to the problem. Our main contributions are as follows:

- We propose the new problem of *optimal paywall mechanism in digital news media*, and model the problem as a sequential stochastic decision optimization problem using the notion of utility and cost.
- We provide a theoretical analysis and design an effective lookahead policy to solve the paywall problem by considering the potential future decisions based on a historical article navigational graph.
- We propose a novel solution using reinforcement learning. In particular, we propose a novel state representation method using utility, cost and embeddings of articles derived from the navigational graph, and use a neural network to learn a policy function.

Authorized licensed use limited to: Ontario Tech University. Downloaded on November 10,2026 at 19:57:42 UTC from IEEE Xplore. Restrictions apply.

- We evaluate the proposed models using a real dataset obtained from a Canadian national newspaper and illustrate that our methods outperform some baselines consistently in terms of different business objectives.

The rest of the paper is organized as follows. Section 2 discusses the related work. The problem we tackle is defined in Section 3. We describe the proposed paywall model in Section 4. The proposed adaptive paywall policies are presented in Section 5. Section 6 presents an application of the proposed methods and its empirical evaluation, and finally Section 8 concludes the paper.

2 Related Work

The paywall problem and pay models are considered in *journalism* as the critical issues and have thus been studied. However, to the best of our knowledge, there are few studies on data-driven approaches to address the paywall problem. In this section, we discuss the related work in journalism. Furthermore, since we consider the paywall problem as a sequential decision over time, we discuss the related areas in machine learning (i.e., *reinforcement learning*) and operations research (*approximate dynamic programming*).

2.1 Digital Journalism and Pay Models

Generating revenue has been a major issue for the news industry over the past decade [3]. While diminishing income due to decline in the advertisement revenue makes newspapers to start implementing paywall mechanisms, there are few studies on the analytical side to make this mechanism more effective. Most studies in the journalism community focus on the qualitative and quantitative investigation of features (e.g., age) influencing people to pay for subscription [4], [5], [6], [7]. For example, Fletcher *et al*. [4] utilized a survey data from six countries to study the factors driving users attitudes to pay for online news. They investigated different hypotheses and concluded that younger people and those who paid for print newspapers were more likely to pay for online news subscription. Similarly, Goyanes [5] studied the factors affecting willingness to pay for online newspapers by conducting a survey of 570 adults chosen at random. The regression analysis showed predictors such as demographic, other online purchases, and social media usage had relationships with tendency to pay for online news subscription. Davoudi *et at*. [7] suggested that subscription to an online newspaper is not instantaneous act but is an informed decision based on certain levels of engagement.

2.2 Reinforcement Learning

Markov Decision Process (MDP) is widely used in reinforcement learning to solve sequential decision problems. When the environment model is available, dynamic programming methods such as *value iteration* or *policy iteration* can be used to find the optimal policy [8]. For example, Shani *et al*. [9] considered the recommendation (in the bookstore domain) as a decision-making problem compared to the traditional prediction perspective. They designed a framework based on MDP and utilized the value iteration approach to make an optimal decision (i.e., whether to recommend an item to a user or not). However, due to the limitation on exploration they used some heuristic techniques to learn the policy.

However, the environment model is often not available. In such cases, Temporal Difference (TD) learning [10] techniques such as Q-learning [11] or SARSA [8] can be utilized to learn the policy from the environment. However, these model free approaches need a lot of interactions (i.e., exploration) with the environment before convergence and suffer from sparsity of reward signals in a highly stochastic environment. Another family of reinforcement learning methods is called *policy gradient* where we learn the optimal policy (a mapping from states to actions) directly to maximize the expected reward (e.g., by using a gradient decent method). One of the first algorithm in this category was called REINFORCE [12]. More recent methods utilized natural gradients to tackle the poor convergence property of policy gradient approaches [13]. Moreover, trust regions and second-order optimization have been used to address overconfidence in taking gradient steps [14]. *Actor-critic* algorithms [15] are the other class of reinforcement learning techniques which learn both the action-value function and the policy at the same time. For example, Mnih *et al*. [16] introduced an asynchronous actor-critic algorithm where parallel agents (actors) act in the environment and collect data with different exploration strategies while the critic (state-value function) provides feedback in order to stabilize training and facilitate faster convergence.

2.3 Deep Reinforcement Learning

The function approximation is used to handle large state spaces when using TD methods (e.g., Q-Learning). In particular, the first successful neural network as a function approximator was Tesauro's TD Gammon [17]. A more recent example is Deep Q-Networks (i.e., DQN) where a Convolutional Neural Network (CNN) is optimized to approximate the Q-values of state-action pairs using visual inputs (i.e., Atari game pixels) as a state representation [18]. The *double* Q-Learning approach tackles the overestimation problem in DQN by decoupling the greedy max operator when taking actions [19]. The other improvement over DQN is the *prioritized experience replay* [20] where samples from the replay buffer are prioritized according to their importance. The idea is that prioritizing significant experiences leads to more efficient learning. Recently, Dai *et al*. [21] used Q-Learning to solve combinatorial optimization problems such as the minimum vertex coverage over graphs. They used Structure2Vec [22], a graph embedding method, preserving structural proximity of nodes, to represent the state space. Moreover, they designed the reward function and cast the problem into a reinforcement learning framework.

2.4 Approximate Dynamic Programming

Approximate dynamic programming studies the sequential decision problem in a more general setting and with broader policy classes such as *myopic policy* (i.e., decision is based on the current state) or *lookahead policy* (i.e., decision is based on the predicted decision in the future) [23]. One related problem in this area is American price optioning [24], which has been studied extensively. American option allows option holders to exercise the option before the maturity date. This problem

Authorized licensed use limited to: Ontario Tech University. Downloaded on November 10,2026 at 19:57:42 UTC from IEEE Xplore. Restrictions apply.

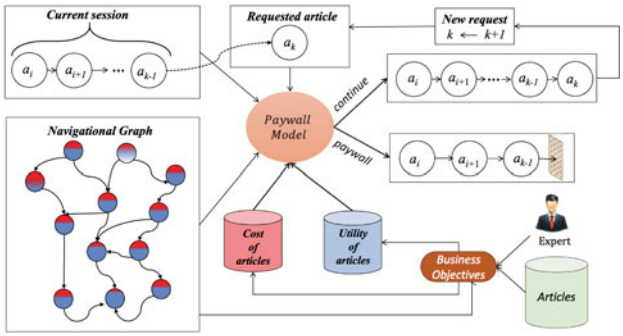


Fig. 3. The proposed adaptive paywall framework.

has been studied in the stock dynamics in a risk neutral world (i.e., a stochastic differential equation). Despite some similarities between this problem and the paywall problem, in the paywall problem we do not have such a dynamic instead we need to build the solution in a data-driven fashion.

3 Problem Definition

We first describe the main components of the proposed model, then define the problem. Fig. 3 shows the proposed framework for the adaptive paywall problem. The main components of the framework are: *utility*, *cost* and the *navigational graph* as well as the *paywall model*. The paywall model receives an article request, makes the decision, and changes the current state of the user session accordingly.

Definition 1 (Utility of Article). *The utility of article a_i, denoted as $\phi(a_i)$, measures the effectiveness/usefulness of the article in achieving a business objective (e.g., user* engagement *for increasing the subscription possibility).*

The utility of an article can be determined by domain experts or learned from historical navigational patterns in a data-driven fashion. For example, if a high percentage of the non-subscribed users reading an article subscribed to the newspaper later, the article has a high utility.

Definition 2 (Cost of Article). *The cost of article a_i, denoted as $\psi(a_i)$, specifies the amount of resources (e.g., time, monetary cost) allocated to produce it.*

The cost can be specified in different ways. For example, the amount of money which the newspaper has to pay to the author, number of pages, etc.

Definition 3 (Navigational Graph). *Navigational graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ is a directed graph, where $\mathcal{V}$ is a set of vertices representing articles, $\mathcal{E}$ is a set of directed edges where an edge e_{ij} from article a_i to a_j indicates a_j has been viewed right after a_i by at least one user, and weight $w_{ij} \in \mathcal{W}$ on e_{ij} represents the number/percentage of the users that read a_j right after reading a_i. Moreover, a_j is considered as an* adjacent neighbor *for a_i.*

The navigational graph encodes historical user navigation behaviours and is used in our model to estimate what article (s) a user is likely to visit next. Although the graph can be built based on all reading sessions that have occurred, we build the navigational graph based on the sessions made by subscribed users since they do not receive paywalls, and thus less bias is introduced.

A *session* is a group of activities (e.g., requesting and reading an article) that a user spends during one visit to the online newspaper. Traditional paywall mechanisms may consider the information in one or more sessions of an unsubscribed user in making a paywall decision (e.g., presenting the paywall if the total number of articles read by a user exceed a limit for a month). Since an unsubscribed user does not have an ID, user identification for each session (e.g., based on IP addresses or cookies) is necessary in order to consider multiple sessions of a user. However, since different users may use the same IP address and cookies can be blocked, tracking users across multiple sessions may be problematic. Thus, we focus on session-based paywall, although the proposed model can be applied to multiple sessions of a user.

Definition 4 (Session-Based Paywall). *In this model, the paywall decision (i.e., whether to present the paywall at a time point in a session) is made based on the information that the user provides in the session (historical records of this particular user).*

An example of session-based paywall is the traditional fixed/metered paywall that allows a user to read a fixed maximum number of articles (e.g., 2 articles) in a session. In this work, we propose an adaptive session-based paywall model in which the number of articles that a user can read depends on what the user has read/requested in the session and estimations of what the user might read in the future. Our adaptive paywall problem is defined as followed.

Problem Statement (*Adaptive Paywall*). Given a navigational graph of subscribed users and a session that an unsubscribed user started, the goal is to determine at what time point during the session the paywall is presented so that the following objective function is maximized:

$$\frac{\sum_{i=0}^{k} \phi(a_i)}{\sum_{i=0}^{k} \psi(a_i)}, \tag{1}$$

where $\phi(a_i)$ and $\psi(a_i)$ are the utility and cost of the ith article that the user reads. The paywall is presented when user requests $k+1'th$ article (a_{k+1}). Note that in the proposed solution, the denominator never goes to zero as the first article is always shown to user.[1] Although we define the objective function using the utility-to-cost ratio, it can be defined in different ways based on business objectives.

4 Proposed Paywall Model

The adaptive paywall problem is a sequential decision problem, that is, at each time step when the user requests an article in a session, a decision needs to be made regarding whether to allow the user to read the requested article or to present the paywall. Formulating and solving such a problem in a disciplined way is not a trivial task. The major challenge is the huge search space due to the high dimensionality of the problem. At each time step when the user requests a new article, in order to find an optimal solution, we need to look at not only what the user has read and requested, but also what the user would request or read in the future if the paywall is not presented at the current time, in order to compare with the values

1. Moreover, we assume small number (0.05) as the article cost for the case that another policy is chosen.

Authorized licensed use limited to: Ontario Tech University. Downloaded on November 10,2026 at 19:57:42 UTC from IEEE Xplore. Restrictions apply.

of the objective function at future time steps. However, what the user will request or read is uncertain at the current time. Considering all possibilities at each time step to find exact solutions is prohibitable due to the large number of articles and combinations of them over multiple time steps. Thus, we resort to approximate solutions.

4.1 Paywall Model Dynamics

One of the most important tasks in approximate dynamic programming (or reinforcement learning) is to design a model of the problem, that is, to design the components of the problem. However, despite the importance of this step, there is no standard approach to modeling the problem [23]. A good model can facilitate the design of policy for solving the problem and may also allow the change of the assumptions (e.g., how to combine utility and cost) based on business objectives and providing alternative solutions accordingly. The major components of a sequential stochastic decision problem are: state variable, decision variable/function, transition function, and contribution function. Given the main components of the model the objective function can be defined accordingly.

State Variable S_t. The state variable at time step t of a session is defined as $S_t = (\overline{u}_t, \overline{c}_t, a_t)$, where a_t is the *article requested* at time t by the user, and $\overline{u}_t$ and $\overline{c}_t$ are the sum of utilities and costs of visited and requested articles (including a_t) by time t in the session, respectively.

The state S_t encapsulates the accumulated information in a *session* by time t, which is used to make the decision at time t. By defining the state in this way, the next state will not depend on the states before this state, which makes the decision process satisfy the Markov property. Meanwhile, the state representation should contain *minimally* necessary information, otherwise it may make the problem computationally intractable. For example, listing all the articles the user has visited in the session so far in a state results in a much richer representation, but it causes the state space to grow exponentially and makes the problem computationally intractable.

Decision Function $X^\pi(S_t)$. The decision function determines the decision/action given state S_t using a policy π, where π is a function that maps S_t into a decision/action. In our problem, there are two possible decisions: presenting the paywall or not presenting it. The decision function is defined as follows:

$$x_t \triangleq X^\pi(S_t) \triangleq \begin{cases} 1 & \text{if } \pi(S_t) \text{ is“presenting paywall”} \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

where x_t is called a *decision variable* indicating the decision taken at time step t. In our framework, policy π is to be designed using a data-driven method.

Transition Function S^M. This function depicts the way that the proposed model evolves from one state to another one as a result of decision and exogenous information (i.e., a requested article at the next time step). The transition function S^M determining the transition from state S_t to S_{t+1} given decision x_t is defined as follows:

$$S_{t+1} = S^M(S_t, x_t, \hat{a}_{t+1}), \tag{3}$$

where S^M maps the components of S_t to S_{t+1} as follows:

$$\text{if } x_t = 0 \quad a_{t+1} = \hat{a}_{t+1} \tag{4}$$

$$\overline{u}_{t+1} = \overline{u}_t + \phi(\hat{a}_{t+1}) \tag{5}$$

$$\overline{c}_{t+1} = \overline{c}_t + \psi(\hat{a}_{t+1})$$

$$\text{else} \quad a_{t+1} = \text{“}paywall\text{”}, \tag{6}$$

where $\hat{a}_{t+1}$ is the article to be requested at time $t+1$. Note that at time t, $\hat{a}_{t+1}$ is uncertain, and thus it is information that arrives exogenously, representing a source of randomness. As a result, its utility $\phi(\hat{a}_{t+1})$ and cost $\psi(\hat{a}_{t+1})$ are random. Also note that if $x_t = 1$ (i.e., the decision at time t is to present the paywall), we assign *paywall* to a_{t+1} to indicate the end state of the decision process. We denote the paywall state as S_p (as we go to S_p, we stay there).

Contribution Function. The immediate contribution/reward function of decision x_t in state S_t measures how much decision x_t at state S_t contributes towards the final objective of the decision process, and is defined as follows:

$$C(S_t, x_t) \triangleq \begin{cases} (\overline{u}_t - \phi(a_t))/(\overline{c}_t - \psi(a_t) + 0.05) & \text{if } x_t = 1 \\ 0 & \text{if } x_t = 0 \text{ or } S_t = S_p \end{cases}. \tag{7}$$

Note that when $x_t = 1$ (i.e., when the decision is to present the paywall), a_t is not presented to the user, thus a_t's utility and cost are not included in the accumulated utility and cost when the ratio is computed in the contribution function. 0.05 in the denominator is to avoid zero division. Also, we set the contribution to zero when $x_t = 0$, so that the contribution is only collected at the paywall time because the contribution at the paywall time considers the utilities and costs of all the articles that user reads in the whole session.

Paywall Decision Problem. We define the paywall decision problem as finding a policy π that maximizes the following objective function:

$$\mathbf{E}\left\{\sum_{t=0}^{\infty} \gamma^t \, C(S_t, X^\pi(S_t))\right\}, \tag{8}$$

where $\gamma \leq 1$ is a discount factor (emphasizing that contributions in the future is not important as the current time contribution). Note that the objective function (8) is the stochastic version of the objective function (1), where we don't know the next article(s) that a user may ask. The need for expectation in Equation (8) is due to randomness of the next requested article, which is necessary for calculating S_t.

4.2 Policy Design Foundation

Solving the optimization problem defined in (8) directly is computationally intractable [23]. In this section, we convert Equation (8) into state value functions and analyze different possibilities for the policy design, and in the next section discuss the proposed methods.

Let $V^\pi(S_t)$ (called the value of state S_t with respect to policy π) be the expected total contribution of a session starting from state S_t and following policy π. That is,

Authorized licensed use limited to: Ontario Tech University. Downloaded on November 10,2026 at 19:57:42 UTC from IEEE Xplore. Restrictions apply.

$$V^{\pi}(S_t) = \mathbf{E}\left\{\sum_{t'=t}^{\infty} \gamma^{t'-t}\ C(S_{t'}, X^{\pi}(S_{t'}))\right\} = C(S_t, X^{\pi}(S_t)) + \gamma E[V^{\pi}(S_{t+1})|S_t]. \quad (9)$$

Note that since the starting state S_t is deterministic, we can take out the first term from the expectation. Clearly, Equation (9) is the same as the objective function (8) when using S_t to denote the initial state. Thus, maximizing Equation (8) is equivalent to maximizing (9).

Theorem 1. *The optimal value of Equation (9) is given by*

$$V^{\pi^*}(S_t) = \max_{x_t \in \{0,1\}} \{x_t C(S_t, x_t) + (1 - x_t)\ \gamma \mathbf{E}[V^{\pi^*}(S_{t+1})|S_t]\}, \quad (10)$$

where π^* is the optimal policy and x_t is the optimal decision at state S_t based on π^* (i.e., $x_t = X^{\pi^*}(S_t)$).

See Appendix for proof, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2020.2969419. Equation (10) provides the insight on how to make an optimal decision for the paywall decision problem. At each state we can make the decision by comparing the optimum value of the state with the expected optimum value of the next states. Note that the value of a next state is computed recursively. One common approach to solve the Equation (10) is *value* (or similarly *policy*) *iteration* [8], which initializes the optimum value of each state randomly, and then updates the values iteratively using the immediate contribution and the expected value of the next future states until convergence. However, this type of methods only applies to problems whose states are discrete and enumerable. The state space in our model is not discrete, so the iteration over the whole space is not possible.

It is possible to design a decision function directly by solving an approximation of the problem over a horizon [2]. This class of techniques is called *lookahead policy* in approximate dynamic programming, and is distinguished from the value function approximation in that it uses random samples to simulate the future and make decisions directly based on the simulated future online without explicitly learning a value function. We present this approach in section 5.1. Alternatively, to avoid the iteration over the whole state space, we design a parametric model based on the reinforcement learning paradigm. In particular, we estimate the *action-value function* (see Section 5.2.1) explicitly by using a neural network and representing states with embeddings (i.e., low dimensional representations) of articles derived from the navigational graph. The detailed explanation of this approach is presented in Section 5.2.

5 Proposed Adaptive Paywall Policies

In this section, we propose two policy-finding methods, namely, *lookahead* and *QPaywall*, to solve the paywall decision problem defined in Equation (8) or equivalently (10).

5.1 Lookahead Paywall Policy

We propose a solution that uses a *lookahead* technique to estimate the decision function directly based on a navigational graph (defined in Definition 3) that records the article-browsing history of users. In our solution, we replace the expectation in Equation (10) with an estimation. In particular, we make the following approximations when formulating the model: 1) we uses a short horizon H to limit the number of future time steps to look into, and 2) instead of using the full set of possible outcomes, we use Monte Carlo sampling to select a subset of outcomes starting at time t based on the navigation graph. Moreover, assuming that we are in a known state S_t at time t, we have different sample paths (i.e., realizations) of the future states from time $t+1$ to $t+H$. Let ω be a sample path of possible article requests from time $t+1$ to $t+H$ (which are stochastic), and $\widetilde{S}_{t'}(\omega)$ and $\widetilde{x}_{t'}(\omega)$ be the state and decision variables at time t' ($t+1 \le t' \le t+H$) for the sample path ω accordingly (when we are in time t). Decisions are based on all stochastic variables over the horizon t to $t+H$ as follows:

$$X^{\pi^*}(S_t) = \underset{\substack{x_t, \widetilde{x}_{t'}(\omega) \in \{0,1\},\\ t+1 \le t' \le t+H, \forall \omega \in \Omega}}{\arg\max} \{x_t C(S_t, x_t) + (1 - x_t) \sum_{\omega \in \Omega} p(\omega) \sum_{t'=t+1}^{t+H} \gamma^{t'-t}\ C(\widetilde{S}_{t'}(\omega), \widetilde{x}_{t'}(\omega))\}, \quad (11)$$

where, Ω is the set of sample paths. In fact, at time t, we solve the problem optimally over horizon t to $t+H$ (using sampling) and find x_t, $\widetilde{x}_{t+1}(\omega)$ to $\widetilde{x}_{t+H}(\omega)$. However, we are not interested in values of $\widetilde{x}_{t+1}(\omega)$ to $\widetilde{x}_{t+H}(\omega)$. We are only interested in x_t, which is a decision at time t. After decision x_t is made, we advance through time and the process is repeated. Note that in Equation (11), x_t is common among all realizations. This procedure results in a simple and efficient method which can be applied to each article request in a session online.

Algorithm 1 shows the designed approach for the paywall problem based on Equation (11). This algorithm receives the current state S_t and navigational graph $\mathcal{G}$ and returns either 1 (i.e., show the paywall) or 0 (i.e., continue without showing the paywall). Line 6-13 in the algorithm create a sample path (i.e., ω) based on the potential sessions encoded in $\mathcal{G}$. The only modification is that we do not proceed with sampling if the entropy of the current vertex is greater than 0.5. The entropy is calculated based on the probability of adjacent/next articles (determined based on weights of edges) in $\mathcal{G}$ as follows:

$$entropy(a_i) = -\sum_{a_j \in \mathcal{N}_e(a_i, \mathcal{G})} \frac{w_{ij}}{\sum_k w_{ik}} \log_b \frac{w_{ij}}{\sum_k w_{ik}}, \quad (12)$$

where w_{ij} is the weight of edge between node a_i and a_j in the navigational graph, and $\mathcal{N}_e(a_i, \mathcal{G})$ is the set of adjacent neighbor vertices of a_i in $\mathcal{G}$. Note that we utilize the relative entropy (i.e., the base of $\log$ is b, where b is the number of adjacent vertices), so it is always between 0 and 1. The reason for using entropy is that if the article entropy is high, jumping to the next article is likely to introduce some noise (i.e., irrelevant articles). In particular, line 8 samples a next potential article $\hat{a}_{t'}$ from the set of neighbor vertices in $\mathcal{G}$ (i.e., $\mathcal{N}_e(a_t, \mathcal{G})$) based on the distribution $Pr(\mathcal{G}, \mathcal{N}_e(a_t, \mathcal{G}))$. In this distribution, for each vertex the probability of each adjacent vertex a_j is calculated by dividing the weight of the outgoing edge to a_j by the sum of all weights of outgoing edges of the

Authorized licensed use limited to: Ontario Tech University. Downloaded on November 10,2026 at 19:57:42 UTC from IEEE Xplore. Restrictions apply.

vertex in navigational graph $\mathcal{G}$. Line 16-20 determines the best paywall time for different sample paths of articles by going over the states (line 17) in the path and finding the best contribution (line 18 and 19). Note that we need to calculate the second part of Equation (11) if we assume that x_t is 0, and in case that x_t equals 1, the first term in the Equation (11) results in the whole contribution. Given that our sampling is unbiased, we have $p(\omega) = 1/|\Omega|$. Finally, we can determine the best value for x_t by comparing the contributions of two cases ($x_t = 0$ or 1) (line 22).

Algorithm 1. Lookahead Paywall Policy (LAP)

Input: $S_t, \mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$
Output: x_t
1: $P \leftarrow 0$
2: **for** $i = 1$ *to* $|\Omega|$ **do**
3: $a_t \leftarrow$ Requested article of S_t
4: $\omega \leftarrow []$
5: $t' \leftarrow t+1, Stop \leftarrow False$
6: **while** $t' \leq t + H$ ***and*** $Stop = False$ **do**
7: **if** $entropy(a_t) \leq 0.5$ **then**
8: $\hat{a}_{t'} \sim Pr(N_e(a_t, \mathcal{G}))$ ▷ N_e is the set of adjacent vertices for a_t
9: $\omega \leftarrow [\omega, \hat{a}_{t'}]$
10: $a_t \leftarrow \hat{a}_{t'}$
11: **else**
12: $Stop \leftarrow True$
13: $t' \leftarrow t' + 1$
14: $P_i \leftarrow 0$
15: $\widetilde{S}_t \leftarrow S_t t' \leftarrow t+1$
16: **while** $t' \leq t + |\omega|$ **do**
17: $\widetilde{S}_{t'} \leftarrow S^M(\widetilde{S}_{t'-1}, 0, \omega_{t'})$ ▷ $\omega_{t'}$ is the article at time t' in ω
18: **if** $P_i < \gamma^{t'-t}\, C(\widetilde{S}_{t'}, 1)$ **then**
19: $P_i \leftarrow \gamma^{t'-t}\, C(\widetilde{S}_{t'}, 1)$
20: $t' \leftarrow t' + 1$
21: $P \leftarrow P + P_i$
22: **if** $C(S_t, 1) > P/|\Omega|$ **then**
23: **return** 1
24: **else**
25: **return** 0

Algorithm 2 illustrates the overall procedure for the adaptive paywall approach. It receives a user request and makes a decision accordingly. Line 3 initializes the session by setting $\overline{u}_0$, $\overline{c}_0$, and a_0 of the first state (i.e., S_0) to $\phi(a_0)$, $\psi(a_0)$ and requested article $\hat{a}$ accordingly. The algorithm always shows the first article (i.e., $x_0 = 0$).[2] If the requested article $\hat{a}$ is not the first one, we first change the state to the next state (line 6), and then based on the result of the policy (Algorithm 1) we either show the article, or go to the paywall and terminate the session.

The above algorithms makes decisions online by simulating future states based on the navigation graph. New articles and changes in user navigation patterns can be incorporated easily by updating the graph. As long as the graph is updated, the algorithms can capture the new changes in the environment.

5.2 QPaywall Policy Learning Approach

The lookahead policy makes paywall decisions based on the potential decisions on some sample paths without explicitly learning the optimal value function. Alternatively, we can learn the value function or action-value function (to be defined in Section 5.2.1) by exploration and exploitation (i.e., taking an action and observe what happens next) rather than estimating what would happen from different sample paths with size H (as described in Section 5.1). In Section 5.2.1, we formally derive such a rule for the paywall problem. However, as we will see this rule is not applicable when the state space is large. Therefore, we introduce a state abstraction approach and a parametric model for action-value estimation in Section 5.2.2. We then propose the learning algorithm for the model in Section 5.2.3.

Algorithm 2. Adaptive Paywall

Input: User Requests, Navigation Graph $\mathcal{G}$
Output: Paywall decision
1: **forall** *Requested article* $\hat{a}$ **do**
2: **if** *t = 0* **then**
3: Initialize the state S_0
4: Show $\hat{a}$
5: **else**
6: $S_t = S^M(S_{t-1}, x_{t-1}, \hat{a})$
7: $x_t \leftarrow$ LookaheadPaywallPolicy(S_t, $\mathcal{G}$)
8: **if** $x_t = 1$ **then**
9: Show paywall and terminate the session
10: **else**
11: Show $\hat{a}$
12: $t \leftarrow t+1$

5.2.1 Action-Value Function

The optimal policy in Equation (9) can be characterized based on the *action-value* (also known as Q) function

$$Q^{\pi^*}(S_t, x_t^*) = C(S_t, x_t^*) + \gamma\, \mathbf{E}\Big[\max_{x_{t+1}} \{Q^{\pi^*}(S_{t+1}, x_{t+1})\} | S_t, x_t^*\Big], \tag{13}$$

where π^* is the optimal policy and x_t^* is the optimal decision at state S_t. The action-value function $Q^{\pi}(S_t, x_t)$ gives the expected contribution of a session when starting in S_t, making decision x_t and following policy π. In order to find the best policy (i.e., π^*) for Equation (13), we adapt the Q-learning approach [11] in which the agent (i.e., digital news company) learns to make decisions optimally by experiencing the consequence of its decisions. In particular, the one-step Q-learning update can be written as follows:

$$Q(S_t, x_t) = C(S_t, x_t) + \gamma \max_{x_{t+1}} \{Q(S_{t+1}, x_{t+1})\}. \tag{14}$$

The learned action-value function Q approximates Q^{π^*} (i.e., optimal action-value function), and converges to it given that all state-action pairs are visited frequently [11]. In

2. If the newspaper does not want to show the first article all the time, it can use some methods to avoid it, such as presenting the paywall right away if the cost of the first article is too high or if its utility-to-cost ratio is too low.

Authorized licensed use limited to: Ontario Tech University. Downloaded on November 10,2026 at 19:57:42 UTC from IEEE Xplore. Restrictions apply.

particular, for each state in a session, we need to take an action x_t, observe the contribution $C(S_t, x_t)$ and the next state S_{t+1} (in our case, the next state is obtained from the training sequence), and update the Q function according to Equation (14). Note that, at the begining, the Q-function can be initialized randomly.

5.2.2 State Representation Using Graph Embedding

While Equation (14) provides the elegant basis for learning the optimal policy in an iterative manner, in our problem the state space is huge, which makes direct usage of Equation (14) inapplicable. In fact, each state S_t in our framework is comprised of a (requested) article, aggregation of utility, and aggregation of cost. Thereby, storing a Q value for each state (e.g., in a *look-up table*) is not possible. Moreover, representing the article component of state as a *one hot* vector whose size equals to the number of articles results in unreliable parameter estimation due to sparsity of the representation and the limited number of observations.

Below we propose to learn a neural network based model $Q(S_t, x_t; \theta)$ to estimate the action-value function Q^{π^*}. We design a feed-forward neural network receiving the state and action as an input and estimating the Q function accordingly. Moreover, we propose to use the low-dimensional representations of each article (also known as the node embedding) in the navigational graph as part of the state representation. This abstract representation of states (i.e., grouping together similar states to form the compressed representations of the environment) enables effective generalization and addresses the sparsity problem.

To this end, we utilize node2vec [25], a neural network based method to learn a continuous feature representation of each node (article) in the navigational graph. The input of the network is the one-hot representation of a node and the output is the probabilities of other nodes being its neighbor. The embeddings of nodes (articles) are the weight matrix between the input and hidden layers, which are updated during the training. More precisely, the method learns the representation $\mathcal{F}(a) \in R^d$ for article a by solving the following optimization problem:

$$\max \sum_{a \in \mathcal{V}} \sum_{n_i \in \mathcal{N}_S(a,\mathcal{G})} \log P(n_i | \mathcal{F}(a)), \tag{15}$$

where $\mathcal{V}$ is the set of nodes (articles) in the navigational graph $\mathcal{G}$ and $\mathcal{N}_S(a, \mathcal{G})$ is the set of neighboring nodes of article a in $\mathcal{G}$, generated by a biased random walk procedure proposed in [25]. Note that $\mathcal{N}_S(a, \mathcal{G})$ may include nodes beyond the adjacent neighbor vertices of a_i in $\mathcal{G}$. Using the softmax function to estimate $P(n_i|\mathcal{F}(a))$, the problem (15) becomes

$$\max \sum_{a \in \mathcal{V}} \sum_{n_i \in \mathcal{N}_S(a,\mathcal{G})} \log \frac{\exp(z_a \cdot z_{n_i})}{\sum_{n_i' \in \mathcal{V}} \exp(z_a \cdot z_{n_i'})}, \tag{16}$$

where z_a is the embedding vector for article a. Maximizing such a function allows articles having similar neighbors in the graph to have similar embeddings. The node2vec algorithm uses Stochastic Gradient Descent (SGD) to train a shallow neural network (i.e., Skip-gram [26]) to obtain embeddings with negative sampling. The time complexity of the training process is $O(tdkn|\mathcal{V}|)$, where $|\mathcal{V}|$ is the number of nodes in the graph, n is the average number of neighbors of a node in the generated random walks, k is the size of negative samples, d is the dimension of the embedding, and t is the number of epochs in training. Usually, k, d, t and $n \ll |\mathcal{V}|$.

5.2.3 Action-Value Function Approximation

Q-Learning is a model-free reinforcement learning approach since it does not make any assumption about the transition function (i.e., S^M) (note that in lookahead approach we implicitly assumed that we have the navigational graph, and then performed sampling). Furthermore, it takes an action based on the greedy policy that approximates $x_t = \arg\max_{x_t'} Q(S_t, x_t'; \theta)$ while ensuring adequate exploration of the state space. We refer to $Q(S_t, x_t'; \theta)$ as a *Q-function* This can be achieved by following ϵ-greedy policy as follows:

$$x_t = \begin{cases} \text{random action} & \text{with prob. } \epsilon \\ \arg\max_{x_t'} Q(S_t, x_t'; \theta) & \text{with prob. } 1-\epsilon \end{cases}. \tag{17}$$

Without exploration, Q-learning is bound to act sub-optimally as a consequence of acting greedily and missing out on potentially better policies. We utilize a feed-forward neural network with weights θ to approximate the optimal Q-function. In particular, the optimal target value $r_t + \gamma \max_{x_{t+1}} \{Q^{\pi^*}(S_{t+1}, x_{t+1})\}$ is substituted with the approximate target values $\hat{y}$ as

$$\hat{y}_t = \begin{cases} r_t \quad \text{if } S_{t+1} = S_p \text{ where } S_p \text{ is the paywall state} \\ r_t + \gamma \max_{x_{t+1}} \hat{Q}(S_{t+1}, x_{t+1}; \hat{\theta}) \quad \text{otherwise} \end{cases}, \tag{18}$$

where r_t is the contribution/reward at time t, i.e., $C(S_t, x_t)$, and $\hat{Q}$ is the target Q-function stored based on the parameters $\hat{\theta}$ from some previous iterations.

Given Equation (18), we can set the target for the neural network (based on $\hat{\theta}$) and learn θ using stochastic gradient decent approach by minimizing the squared loss

$$(\hat{y}_t - Q(S_t, x_t; \theta))^2. \tag{19}$$

However, Q-learning is susceptible to oscillations and even divergence when a non-linear function approximator such as a neural network is utilized to model the Q-function [27]. The oscillations are due to correlation among sequences of observations and also between state-action values (Q) and target values $r + \gamma \max_{x'} \hat{Q}(s', x')$. Inspired by [18], to make the learning stable: (1) we store the observations at time t, $o_t = < S_t, x_t, r_t, S_{t+1} >$ in a buffer $\mathcal{D}$, then during the learning we apply Q-learning updates on a mini-batch $\mathcal{M}$ of observations drawn uniformly at random from $\mathcal{D}$. We thereby remove the correlation among the sequence of observations and reduce the variance of updates. (2) we periodically update the target value function ($\hat{Q}$) with the action-value function (Q). This change imposes a delay between the Q update time and the time that update effects the target values, and as a result makes the correlation occurrence between Q values and target values less likely.

Algorithm 3 outlines the procedure for learning the parameter θ of the Q-function. The algorithm receives the training sessions $\mathcal{X}$ and updates the parameters θ iteratively. The

Authorized licensed use limited to: Ontario Tech University. Downloaded on November 10,2026 at 19:57:42 UTC from IEEE Xplore. Restrictions apply.

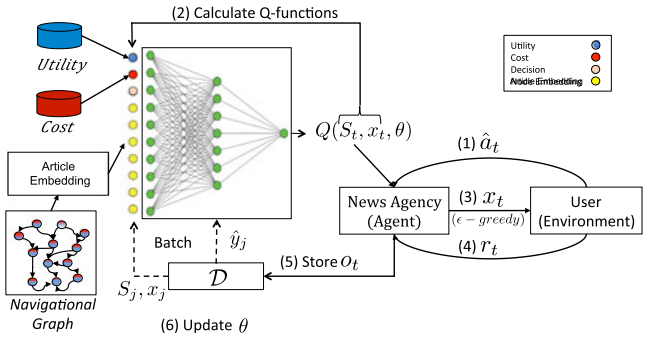


Fig. 4. The proposed QPaywall framework.

algorithm starts by initializing the parameters θ randomly and setting $\hat{\theta} = \theta$. In each iteration, we sample a session from the training data, pick the first article of the session and initialize the first state using the utility function ϕ, and the cost function ψ, and the article embedding mapping $\mathcal{F}$ (line 6). While the session is not terminated (i.e., when the paywall is not presented, and the end of session is not reached), first, we choose the action using Equation (17) (based on the the current Q-function). Then, in line 9-12, we observe the contribution r_t, move to the next state S_{t+1} and store this transition in $\mathcal{D}$. As we discussed, the stochastic gradient descent update of θ is based on a mini-batch $\mathcal{M}$ (rather than a single observation) drawn randomly from the whole past transitions stored in $\mathcal{D}$. For each transition $o_j = \langle S_j, x_j, r_j, S_{j+1} \rangle$ in $\mathcal{M}$, $\hat{y}_j$ is computed using Equation (18) as the target value with the target network $\hat{Q}$, and the predicted value (i.e., $Q(S_j, x_j, \theta)$) is calculated using current network Q. The current network is updated by minimizing $(\hat{y}_j - Q(S_j, x_j; \theta))^2$ with all the transitions in $\mathcal{M}$. Finally, we update $\hat{\theta}$ with θ once every k updates of θ, and meanwhile we calculate the target value $\hat{y}_i$ based on the parameters $\hat{\theta}$. Fig. 4 illustrates how QPaywall simulates the interactions between users and the news agent in its training. Briefly, (1) when a user requests an article ($\hat{a}_t$), the state is computed using its embedding along with the aggregated utility and cost in the session. (2) The Q-values of the state-action pairs are calculated using the current neural network. (3) The agent then takes an action (x_t) with an ϵ-greedy policy according to the Q-values. (4) A reward (r_t) is calculated and we go to the next state. If the action is to show the paywall, then the session ends; if not, we move on to the next article that user requests. (5) The transition $o_t = \langle S_t, x_t, r_t, S_{t+1} \rangle$ is stored in $\mathcal{D}$. Meanwhile, (6) training continues by sampling a mini-batch from $\mathcal{D}$ and updating the parameters θ.

To use the QPaywall policy with the overall adaptive paywall procedure, line 7 of Algorithm 2 can be changed to

$$x_t \leftarrow \arg\max_{x'_t} Q(S_t, x'_t; \theta), \tag{20}$$

in order to calculate decision x_t based on the learned action-value function Q.

Algorithm 3. QPaywall (Learning Parameters)

Input: Training sessions $\mathcal{X}$, ϵ, k
Output: θ
1: Initialize θ randomly
2: $\hat{\theta} \leftarrow \theta$
3: **for** $i = 1 : N$ **do**
4: Sample session $s \in \mathcal{X}$
5: $\hat{a} \leftarrow$ first article of s
6: $S_1 \leftarrow (\phi(\hat{a}), \psi(\hat{a}), \mathcal{F}(\hat{a}))$, $t \leftarrow 1$
7: **while** s *NOT terminated* **do**
8: $x_t \leftarrow \epsilon$-greedy policy (Equation (17))
9: $r_t \leftarrow C(S_t, x_t)$
10: $\hat{a} \leftarrow$ next article from s
11: $S_{t+1} \leftarrow S^M(S_t, x_t, \hat{a})$
12: Store transition $o_t =< S_t, x_t, r_t, S_{t+1} >$ in $\mathcal{D}$
13: Sample mini-batch $\mathcal{M} = \{o_j\}$ from $\mathcal{D}$
14: $\hat{y}_j \leftarrow$ calculate for all $o_j \in \mathcal{M}$ (Equation (18))
15: Update θ using gradient descent on $(\hat{y}_j - Q(S_j, x_j; \theta))^2$ w.r.t θ
16: After every k gradient descent updates, $\hat{\theta} \leftarrow \theta$
17: $t \leftarrow t + 1$
18: **end**
19: **end**

6 Empirical Study

We evaluated the proposed method on a real dataset from The Globe and Mail,[3] a major newspaper in Canada.

6.1 Dataset

The Globe and Mail dataset was collected based on the *omniture*[4] data collection platform. Whenever a user reads an article, watches a video or generally interacts with the news portal, a record called *hit* is generated and stored. Typically, a hit contains information such as user environmental variables (i.e., *browser type*, *IP address*), *user id* (for a subscribed user), *session id*,[5] *visited article id*, *click timestamp* as well as *special events* (e.g., subscription, sign in, etc). The original dataset contains about 2 billion hits. We aggregate page view hits to the session level. Moreover, we calculate dwell time (i.e., the time that a user spends on an article) using the difference between two consecutive hits' timestamps. The last article in the session is ignored since we cannot calculate its dwell time. In addition, the time beyond 500 seconds are considered as page idle time. The threshold 500 is chosen heuristically based on dwell time distribution observed in the preprocessing step[6] (the distribution is not disturbed when a few problematic instances with possibly long idle time are removed using the threshold). Note that we do not have any client-side signal beyond the click timestamps in our dataset. In case that this information available, it is possible to have more accurate idle time for each individual user (e.g., see [28]). We use the data collected within 2014-01 to 2014-07 for the evaluation purpose. After preprocessing (i.e., aggregation), the dataset contains 4,913,423 sessions from subscribed users. The number of subscribed users is 137,170. An additional 14,050 users were converted to subscribed ones during the period. The average number of sessions per subscribed users and converted ones are 31.13 and

3. www.theglobeandmail.com

4. https://my.omniture.com

5. In Omniture data collection platform, no activity in 30 minutes is considered as the end of session.

6. Interestingly, we did not observe any strong correlation between articles lengths and their dwell times, so we use such a threshold for all articles.

Authorized licensed use limited to: Ontario Tech University. Downloaded on November 10,2026 at 19:57:42 UTC from IEEE Xplore. Restrictions apply.

45.75 respectively. We use sessions with minimum 10 articles as the test and the rest as the training set (e.g., to build the navigational graph). The test data set provide real sequences of article requests for use to evaluate the proposed methods. A minimum of length 10 gives us more possible time points for making paywall decisions in a session. The numbers of sessions in the training and test sets are 4,806,204 and 107,219, respectively.

6.2 Utility and Cost Models

While the utility and cost of an article can be defined differently in our model, we use two intuitive utility and cost models in our experiments, namely, *Engagement to Cost (E/C)* and *Acquisition to Cost (A/C)*. For both models the cost (i.e., ψ) is defined as the article length (in terms of the number of 1 KB pages) as the lengthy articles often need more efforts and resources to produce. The utilities for these models are defined as follows:

- The utility of article a in (E/C) model is defined as

$$\phi(a) = \frac{Total\ dwell\ time\ by\ all\ users\ on\ a}{Total\ no.\ of\ visits\ of\ a}, \tag{21}$$

  where the unit of time is second.
- The utility of article a in (A/C) model is defined as

$$\phi(a) = \frac{Total\ no.\ of\ visits\ of\ a\ before\ subscription}{Total\ no.\ of\ visits\ of\ a}, \tag{22}$$

  where the numerator is the number of subscribers who visited the article before the subscription in a subscription session.

Note that the second measure has very small values as the number of subscribers is much smaller than non-subscribed users. It is computed based on sessions from both subscribed and non-subscribed users.

6.3 Baselines and Performance Measures

We compare the proposed Lookahead Paywall Policy (denoted as *LAP*) and QPaywall (denoted as *QP*) with the following baselines:

- *Fixed Policy (FP):* This is a commonly-used paywall mechanism which allows a user to visit a maximum number, T, of articles during a session.
- *Average Threshold (AT):* This is a type of myopic policy [23] which defines the analytical decision function only based on the *current state* using a threshold. The decision function in this method is defined as follows:

$$x_t \triangleq X^{\pi}(S_t|\tau) \triangleq \begin{cases} 1 & \text{if } (\overline{u}_t/\overline{c}_t \leq \tau) \\ 0 & \text{Otherwise} \end{cases}, \tag{23}$$

  where τ is set as the average utility-to-cost ratio, calculated based on the sessions in the training set.
- *Policy Function Approximation (PFA):* This policy is based on Equation (23), but the parameter τ is optimized using sessions in the training set. Given (23), the policy search in the base optimization problem (8) is changed to a parameter search. Therefore, we can rewrite the optimization function in (8) as

$$\max_{\tau}\ J(\tau) = \max_{\tau} \mathbf{E}\left\{\sum_{t=0}^{\infty}\ \gamma^t\ C(S_t, X^{\pi}(S_t|\tau))\right\}. \tag{24}$$

  Finding the best value of τ is a stochastic search problem. However, we cannot compute $J(\tau)$ in a compact form and $X^{\pi}(S_t|\tau)$ is not differentiable. Therefore, we use finite difference gradient [15], [29] which is a common approach for solving Equation (24).

We use the following *performance measures* when comparing the methods. (1) *Policy performance*, which is defined as the average ratio of aggregated utility of all articles in a session to the aggregated cost of all articles in the session over all the tested sessions. The utility of an article is defined using either the E/C or A/C model (according to Equations (21) or (22)). Thus, we have two policy performance measures: E/C based or A/C based. (2) *Policy performance at different percentages of articles delivered to users*, which measures the average utility-to-cost ratio of the sessions when a percentage of articles is presented to users. Obviously, the higher the ratio, the better the performance. (3) *The percentage of active sessions at each time point*, which is the percentage of sessions that have not received the paywall at each time point. A method A with more active sessions at time t than method B is better at t if it has at least the same or better policy performance than B at t. This is because as long as the trade-off between cost and utility is fine, keeping the user active without presenting the paywall can further engage the user and also deliver advertisements.

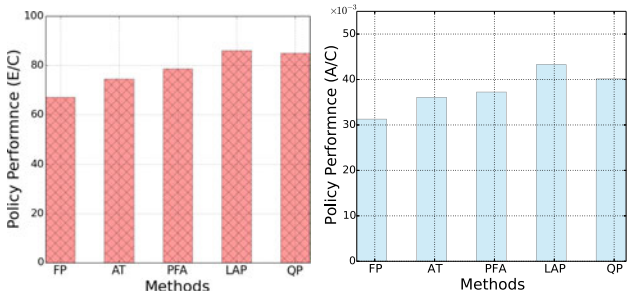
6.4 Experimental Setting

In our experiments, we vary the value for T in the FP method. We set γ to 1 (which means that the future articles visits have the same value as the current one) and the horizon size H in Algorithm 1 to 4 unless indicated otherwise. Moreover, we checked different sample sizes and observed that after 10, the performance is not improved significantly. Therefore the sample size is set to 10 for all experiments.

For the QPaywall-Learning (Algorithm 3) samples are drawn uniformly at random from the training sessions. Moreover, 10 percent of the training sessions drawn uniformly as the validation set to tune the hyperparameters. The exploration parameter ϵ is set to 0 for testing and for training is set to a piecewise schedule (starting from 1) according to [30] (i.e., ϵ decreases over iterations based on a linear piecewise function). Moreover, the neural network used in the QPaywall experiments is a feed-forward fully-connected neural network with a single hidden layer of 256 nodes. The input layer has 130 nodes consisting of 128 nodes for the node2vec embedding vector and 2 nodes for the aggregated cost and utility, respectively. The output layer consists of one node representing the value of the action (paywall or Not paywall) in a particular state. The ReLU activation function is used throughout the network. We use ADAM optimizer to train the network with default parameters from Tensorflow.

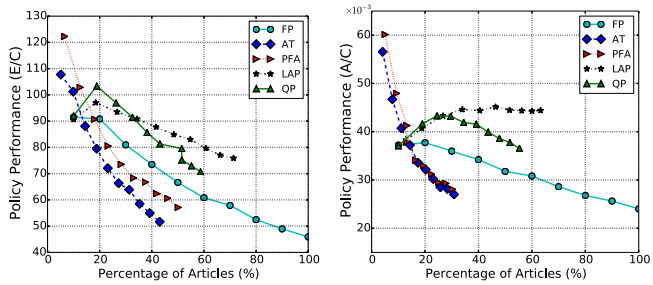
The hyperparameters (including exploration parameter ϵ, learning rate, discount factor γ, batch size and target network update period k) are set according to a random hyperparameter search [31] on the validation set. The hyperparameters are the same for both A/C and E/C models unless indicated otherwise. That is, the target network update period k is set

Authorized licensed use limited to: Ontario Tech University. Downloaded on November 10,2026 at 19:57:42 UTC from IEEE Xplore. Restrictions apply.



(a) E/C model (b) A/C model

Fig. 5. Average policy performance of different methods.



(a) E/C model (b) A/C model

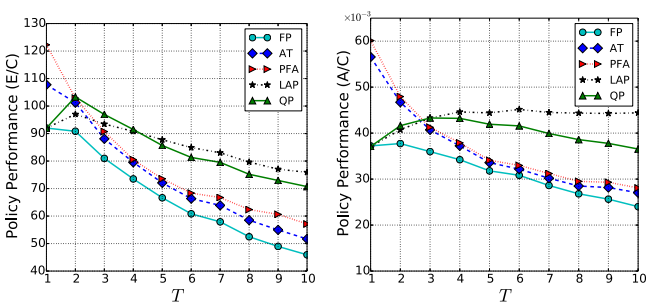
Fig. 7. Policy performance versus article delivery percentage.

to $20K$ steps, and replay buffer size (i.e., $|\mathcal{D}|$) and batch size are set to $10K$ and 64, respectively. Finally, node2vec parameters are set to default [25].

6.5 Policy Performance Analysis

Fig. 5 shows the overall average policy performance of the proposed models compared to the baselines for the engagement (i.e., E/C) and acquisition (i.e., A/C) utility models. For the FP method, the result is the average over T values ranging from 1 to 10. The LAP method shows the best performance with 28.4 and 38.3 percent performance improvements over the traditional FP method for the E/C and A/C model respectively. The second best method is QP with 26.8 and 28.3 percent improvements over the FP method for the E/C and A/M model respectively.

In Fig. 6, we compare the policies at different T values, where all policies are allowed to show maximum T articles. Fig. 6a illustrates the performance of the polices for the E/C model, which shows that the commonly-used FP policy has the lowest performance for all T values. It also shows that the performances of PFA and AT are better than the other methods at the beginning when T is small (i.e., less than 3). This is because PFA and AT greedily terminate sessions if the articles requested so far do not look promising based on the threshold τ without looking at the future, and if forced to stop early, the LAP and QP methods do not have the opportunity to make a better decision later on even if they find one by looking into the future. Moreover, QP shows better performance in comparison to LAP for small T (i.e., less than 4 and 3 for E/C and A/C models respectively). However, LAP performs better than QP for the bigger T values. One reason is that QP learns the Q-value function by only observing one-step contributions while the LAP method takes a look at a longer horizon to calculate the expected contribution



(a) E/C model (b) A/C model

Fig. 6. Policy performance for different models.

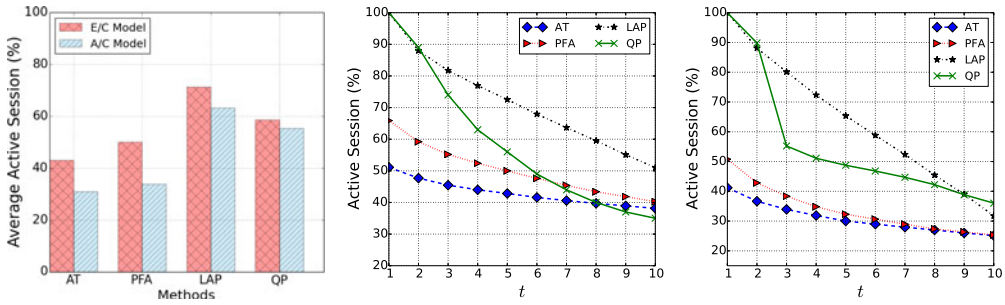
estimation. As T increases, LAP performs better as it can present the paywall at a later time if it thinks it is better than current ones. Another observation is that while the performances of all policies for the E/C model decline as we increase T (due to fact that users may visit more engaging articles at the start of sessions), the LAP policy outperforms the other policies by keeping alive the sessions with the promising future. Finally, between PFA and AT, PFA is marginally better than the AT method. Similar observations are found in Fig. 6b, which illustrates the performances of the policies for the A/C model at different T values. The only difference is that as T increases, LAP's performance does not decline, which means that the LAP policy successfully navigates users to the articles that are useful for user acquisition (i.e., those which have been visited by the converted users and have good utility-to-cost ratios).

6.6 Performance Versus Delivery Percentage

In this section, we study the performance of each policy at different percentages of delivered articles (similar to precision at each recall level in information retrieval). To do this, for each policy and each test session s and at each time point t (from 1 to 10), we compute the percentage of delivered articles by time t as the number of articles delivered by the policy by time t divided by the total number of articles in session s, and also compute the ratio of the aggregated utility to the aggregated cost of all the delivered articles in the session by time t. In this way, for each policy and each session we obtain 10 $\langle$ delivered article percentage, policy performance $\rangle$ pairs, one for each time point. We finally take an average of each pair over all the test sessions for the policy. Thus, each policy has 10 averaged pairs, one for each time point. For the FP method, we vary T from 1 to 10 to collect the data for each time point.

Fig. 7a shows the policy performance against the percentage of delivered articles for the E/C model. As can be observed, at the small article percentage levels (less than 14 percent), PFA and AT are better than other models. However, as we increase the percentage of articles, both LAP and QP outperform PFA and AT, with QP being better than LAP at the beginning then LAP significantly outperforming all other polices consistently. In particular, LAP performs 31.0 percent better than FP when 70 percent of articles are shown to users. Similar results are found in Fig. 7b, which shows the policy performance against the percentage of delivered articles for the A/C model. For example, the LAP's performance is 48.0 percent better than that of FP when 63 percent of articles are shown to the users. Also, QP is

Authorized licensed use limited to: Ontario Tech University. Downloaded on November 10,2026 at 19:57:42 UTC from IEEE Xplore. Restrictions apply.



(a) Average active session (b) E/C model (c) A/C model

Fig. 8. Active session for different utility to cost models.

slightly better than LAP when the delivered article percentage is less than 25 percent.

6.7 The Effect of Policies on Users' Sessions

In this section, we investigate the effect of different policies on the number of active sessions (i.e., the percentage of sessions/users which have not received paywall before time point t). Intuitively, an online news agency prefers to have a high number of active readers all the time as long as the articles the reader reads have good utilities and their total cost is reasonable (i.e., the ratio of utility to cost is acceptable). Keeping a user active without presenting the paywall can further engage the user and allow more display of advertisements. Fig. 8a illustrates the average percentage of active sessions (averaged over different time points in a session) for both E/C and A/C models. As can be seen, LAP has more active sessions on average compared to other policies. For example, the average number of active sessions of LAP is 42.5 and 66.0 percent more than PFA and AT methods respectively for the E/C model. The next best model is QP with 58.7 and 55.4 percent average percentage of active sessions for E/C and A/C models, respectively. Note that we did not show the FP method in this study because all sessions are active till time T in FP (which is assumed to be 10). Figs. 8b and 8c illustrate the percentage of active sessions at different t values. PFA and AT terminate many sessions at the beginning by showing the paywall because close to half of the articles do not meet their threshold which is the average or close to average utility-to-cost ratio learned from the training data, while in LAP there are more active sessions at all the time points and the percentage of active sessions decreases more smoothly. Considering that LAP has better utility-to-cost ratios than other methods at almost all the time points (as shown in Fig. 6), LAP is the best as it has more active users without sacrificing the utility-to-cost ratio. QP is the second best in the same sense.

6.8 Runtime Analysis

Fig. 9 shows the average run time per request for the LAP and QP methods during the decision making process (i.e., the inference time). The experiments are run on a 2.2 GHz Intel Core i7 machine and a GeForce GTX 1080 Ti GPU[7] As can be seen, the response time per request for both methods is in the order of milliseconds. The time for LAP increases almost linearly with horizon H. However, the decision time for QP does not depend on a horizon because it only relies on the time to calculate the Q-values from the learned Q function and return the action with the maximum Q-value. Thus, QP is much faster than LAP in decision making. For example, the QP inference time is 9.6 times better than LAP where the horizon length (H) is 4.

The learning of the Q function in the QP method is conducted offline. Assuming the navigation graph is given, training in QP consists of computing the article embeddings using node2vec and learning the Q function using Algorithm 3. On our data set, node2vec takes about 4 and a half minutes and learning the Q function takes 20 minutes on average. Training time (upper horizontal axis) can be seen in Figs. 11 through 13 as part of the sensitivity analysis conducted in the next section. In comparison, LAP doesn't have a training phase since the method works on the fly.

6.9 Sensitivity Analysis

6.9.1 LAP Sensitivity Analysis

We analyze the performance sensitivity of the LAP policy with different horizon sizes (i.e., H). We change the horizon size H and calculate the policy performance accordingly. Fig. 10 shows the effect of horizon size H on the LAP performance for the E/C and A/C models. As can be observed, by incrementing H, the performance increases slightly initially for both E/C and A/C models, becomes stable in range of 2 to 4, and then starts to decline. This suggests that any value in range 2 to 4 would be a good choice for this parameter.

6.9.2 QP Sensitivity Analysis

We show the robustness/sensitivity of the QPaywall algorithm against the hyperparameters (i.e., minibatch size,

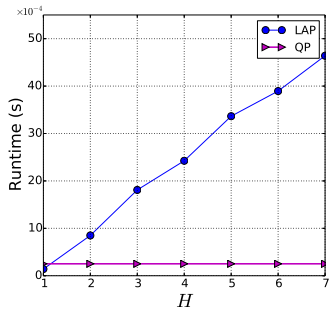


Fig. 9. Average Inference Time per request.

7. GPU is used only for training the Q-function in QPaywall. The inference time comparison shown Fig. 9 was done on CPU.

Authorized licensed use limited to: Ontario Tech University. Downloaded on November 10,2026 at 19:57:42 UTC from IEEE Xplore. Restrictions apply.

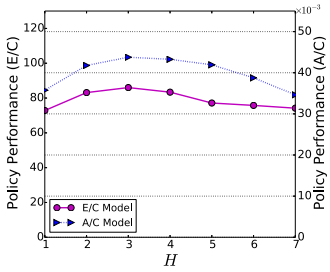
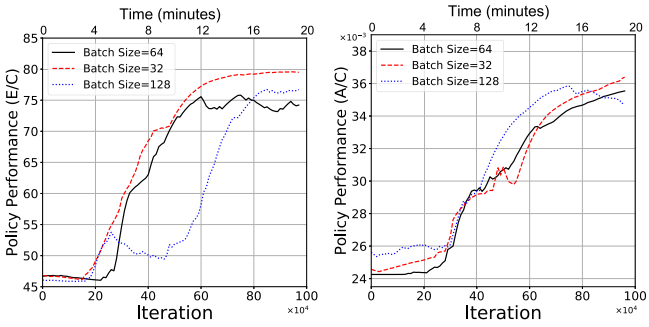
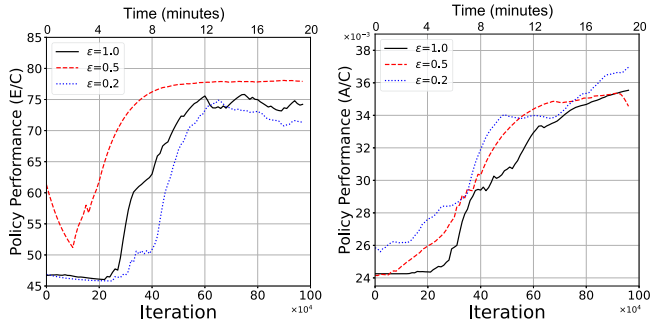


Fig. 10. Performance versus H.



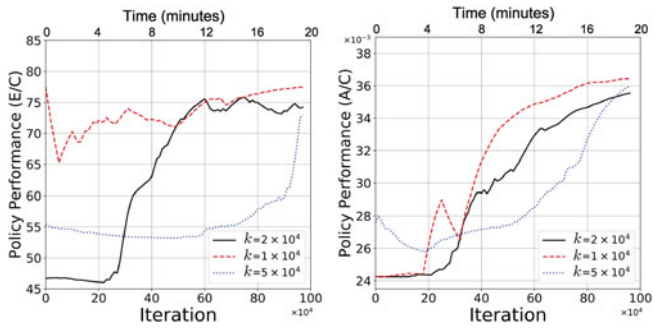
(a) E/C (b) A/C

Fig. 11. The effect of batch size on performance. Iteration is the i value in Algorithm 3.



(a) E/C (b) A/C

Fig. 12. The effect of initial ϵ on performance.



(a) E/C (b) A/C

Fig. 13. The effect of target update period k on the performance.

exploration parameter ϵ and target network update period k). In each experiment, we change the value of the parameter in focus, while keeping the rest of hyperparameters fixed.

Fig. 11 shows the effect of batch size on the QP performance. As can be seen, the size of batch in Algorithm 3 makes slight difference in the performance for the A/C model, and on both E/C and A/C models, the algorithm converges to similar values for the three batch sizes.

In QPaywall (Algorithm 3), parameter ϵ makes a trade-off between the exploration and exploitation. We follow the method in [30] and set the ϵ according to a piecewise schedule, where the learning rate is set to 0.001 for half of the iterations and 0.0005 for the rest. Fig. 12 illustrates the performance using different initial values for the ϵ parameter. As can be seen, for both E/C and A/C models, the performance is increasing over time. However, for the E/C model, the moderate value (i.e., 0.5) of ϵ produces the faster convergence, and for the A/C model, the low value (i.e., 0.2) of ϵ is slightly better in terms of convergence. Another hyperparameter in QPaywall is the period of target update k. Fig. 13 illustrates that although smaller k values (e.g., $k = 1 \times 10^4$) may result in faster convergence, the final performance is not sensitive to this hyperparameter.

In summary, Figs. 11, 12, and 13 show that the proposed algorithm can converge to similar performance values across a wide range of hyperparameter values.

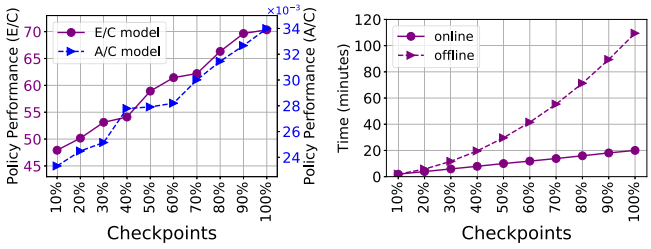
6.10 Discussion on LAP Versus QP

LAP is a paywall planning algorithm where the decision at each state is made online by simulating trajectories (i.e., sample paths) controlled by the horizon size from that state. This approach is similar to algorithms where Monte Carlo (MC) simulation is used to estimate the average return from a state and the decision is made by comparing the average returns (i.e., choosing an action which gives the highest average return). Thus, LAP needs more computation time per decision than QP. On the other hand, QP has an offline training phase, in which it uses one-step Temporal Difference learning to learn estimates of action-values by bootstrapping (i.e., updates to the estimated action-values use previously estimated values of the same kind) and function approximation. For a comprehensive comparison in terms of training time and inference time we refer the reader to the runtime analysis Section 6.8. Such a one-step learning method is beneficial in applications where sampling trajectories for more than a few steps is infeasible (which is not the case for the paywall problem as the decision time is very reasonable in LAP). However, in the paywall problem, observations are sparse and rewards are delayed. Therefore, estimation based on more than one-step might be more helpful in order to capture delayed rewards. This could be a reason for the superiority of LAP over QP on larger T values. The better performance of QP over LAP for smaller T values (as shown in the results) can be explained by the fact that LAP makes a decision based on only a relatively small sample of trajectories online, while QP learns a generalized function by going over a large number of state transitions during training. In addition, in QP learning states are vectors (abstract representations of states) and thus what is learned for one state can be used for other states with similar vector representations.

7 Practical Considerations

In practice, the proposed framework is connected to other components (i.e., navigational graph management, embedding

Authorized licensed use limited to: Ontario Tech University. Downloaded on November 10,2026 at 19:57:42 UTC from IEEE Xplore. Restrictions apply.



(a) Performance (b) Online vs. Offline

Fig. 14. QPaywall online scenario.

calculation, and utility and cost prediction). Although a complete description of these components is beyond of the scope of this paper, for completeness, we discuss some major issues in designing each component.

Updating the Navigational Graph Over Time. The navigational graph encodes the users' (i.e., subscribed users) sessions. Visiting one article after another one by a user increases the weight of the edge between two articles (i.e., nodes) in the graph. Moreover, as a new article is published, we can easily add an article (i.e., node) to the navigational graph. Both these operations can be performed efficiently (i.e., O(1)). Therefore, changes in the navigational graph can easily be reflected in the LAP approach because it directly exploits this graph for making the decision.

Updating Article Embeddings Over Time. Updating graph embedding over time can be done periodically, but due to inefficiency, researchers are exploring online versions of updates. Dynamic graph embedding is an active area of research [32]. Alternatively, the inductive graph embedding approaches [33] have the capability of generating a node embedding for a previously unseen node. These methods are used to learn node embeddings from dynamical graphs.

Online Learning of Q Function. In order for the QPaywall algorithm to stay accurate over time, it has to adapt to new changes in the navigational graph. Similar to graph embedding, QPaywall can also be updated periodically offline. To see if it is feasible to learn the Q function online, we conducted an experiment simulating an online scenario. The experiment is constructed by dividing the sessions (navigational graph) into 10 chunks in order of occurrence. We assume that the embedding is given for all articles in the graph beforehand. Fig. 14a shows the performance over time (checkpoints are when 10 percent of the sessions is added). Note that we have used the default hyperparameters from the previous experiments for each model. As training goes on and more data are added, we see an improvement in the policy performance of the model. This improvement is the result of generalization with more information in the graph. We also calculate the average training time of E/C and A/C models (i.e., the GPU time dedicated to training) needed for both online and offline scenarios (Fig. 14b). For the offline version, the model at a checkpoint was trained using all the data from the beginning to that checkpoint, while an online model was initialized with the model trained in the previous checkpoint and fine-tuned with the new data. The online version, at each checkpoint, updates the model at a rate of 0.00027 seconds per session on average.

Selecting Utility and Cost. The proposed model is independent of definitions of utility and cost. In this paper, we defined two measures for the utility. One is based on the average time that users spend on the article. It measures the time engagement of the article and thus useful when the news agency wants to improve revenue through advertisement as more dwell time increases the probability that user notice advertisements. The other one measures how likely the article leads the reader to subscription. The above measures are based on users' browsing history and thus may not be used for new articles. Thus, building a model to predict the article utility is necessary. In [34], we developed the model to predict the article engagement based on its content, which can be used to predict utility. Other ways to define utilities include the number of advertisements on a page, the uniqueness of articles among news agencies, the number of submissions to social media and/or the number of comments to name a few.

8 Conclusion and Future Work

We proposed an adaptive paywall mechanism for digital news media. The traditional paywall model allows a user to see a fixed number of articles and then directs them to the subscription page. We argued that this approach does not lead to more subscriptions and sacrifices other business objectives (e.g., increasing the user dwell time, the number of visits, etc.). We proposed a solution by formulating the paywall problem as a sequential decision problem that optimizes the ratio of the aggregated utility of the articles presented to the user to their aggregated cost. We defined our problem and its components and analyzed possible ways to solve it. In particular, we proposed two approaches: lookahead policy (LAP) and QPaywall policy (QP) as data-driven policies to solve the adaptive paywall problem. The LAP method works on the fly. It makes paywal decisions by simulating trajectories of article requests from a state using Monte Carlo sampling over a navigational graph, estimating the average reward based on the sampled paths, and comparing it with the reward for presenting the paywall. QP is based on reinforcement learning. It learns a neural network-based action-value (Q) function offline/online using one-step observations from the training sessions.

We applied the proposed methods to a real dataset from a national newspaper in Canada and showed the benefits and superiority of the methods over the fixed paywall policy and two other alternative baselines. Our results showed that both the Lookahead and QPaywall policies outperformed the three baseline methods in terms of *policy performance* measures. Between the two proposed methods, the Lookahead method generally performs better in terms of *policy performance* except when it is not allowed to look more than 3 or 4 steps into the futures. In terms of decision-making/inference time, both methods run in the order of milliseconds. Between the two methods, QPaywall is faster since its inference only involves computation of Q values based on a learned Q function, while LAP does all its computation online. However, LAP does not incur training time as long as the navigational graph is given (which is trivial to built and update), while QPaywall requires an offline training of a neural network, which needs to run periodically (or incrementally online) in a real-world setting.

There are interesting directions for future work. We are currently using one-step observations sampled from the

Authorized licensed use limited to: Ontario Tech University. Downloaded on November 10,2026 at 19:57:42 UTC from IEEE Xplore. Restrictions apply.

historical sessions to learn the Q function, which may be the reason why QPaywall does not perform as well as the Look-ahead method with large time horizons. A Q-learning method that can learn action-value functions using n-step observations and their returns will be investigated in the future. Another interesting direction is to use *game theory* to formalize the problem. Game theory is the study of two or more rational agents interacting with each other to either achieve different or the same goals. As an example, we believe that this problem can be formulated as a sequential (extensive-form) game where the news agency's action is whether to show the paywall or not and the user's action is whether to subscribe or not if they see the paywall, and whether or not to click on the next article (if there's no paywall). Assuming a finite number of articles, a subgame perfect Nash equilibrium can be found (since every finite extensive game will have an equilibrium). Finally, we hope what we have presented in this paper can inspire others searching for solutions to similar real-world problems.

Acknowledgments

This work was funded by Natural Sciences and Engineering Research Council of Canada (NSERC), The Globe and Mail, and the Big Data Research, Analytics and Information Network (BRAIN) Alliance established by the Ontario Research Fund - Research Excellence Program. We would also like to thank the anonymous reviewers for their comments and suggestions, which greatly improved this article.

References

[1] A. Williams, "Paying for digital news: The rapid adoption and current landscape of digital subscriptions at US newspapers," American Press Institute, 2016. [Online] Available at: https://www.americanpressinstitute.org/publications/reports/digital-subscriptions/

[2] H. Davoudi, A. An, M. Zihayat, and G. Edall, "Adaptive paywall mechanism for digital news media," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 205–214.

[3] N. Newman, R. Fletcher, A. Kalogeropoulos, D. A. Levy, and R. K. Nielsen, "Reuters institute digital news report 2017," Reuters Institute for the Study of Journalism, 2017. [Online] Available at: https://reutersinstitute.politics.ox.ac.uk/sites/default/files/Digital%20News%20Report%202017%20web_0.pdf

[4] R. Fletcher and R. K. Nielsen, "Paying for online news: A comparative analysis of six countries," *Digit. Journalism*, vol. 5, no. 9, pp. 1173–1191, 2017.

[5] M. Goyanes, "An empirical study of factors that influence the willingness to pay for online news," *Journalism Pract.*, vol. 8, no. 6, pp. 742–757, 2014.

[6] H. Chyi, "Willingness to pay for online news: An empirical study on the viability of the subscription model," *J. Media Econ.*, vol. 18, no. 2, pp. 131–142, 2005.

[7] H. Davoudi, M. Zihayat, and A. An, "Time-aware subscription prediction model for user acquisition in digital news media," in *Proc. SIAM Int. Conf. Data Mining*, 2017, pp. 135–143.

[8] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2011.

[9] G. Shani, D. Heckerman, and R. I. Brafman, "An MDP-based recommender system," *J. Mach. Learn. Res.*, vol. 6, pp. 1265–1295, 2005.

[10] R. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, no. 1, pp. 9–44, 1988.

[11] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3/4, pp. 279–292, 1992.

[12] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3/4, pp. 229–256, 1992.

[13] S. M. Kakade, "A natural policy gradient," in *Proc. 14th Int. Conf. Neural Inf. Process. Syst.*, 2002, pp. 1531–1538.

[14] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. 31st Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

[15] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. 12th Int. Conf. Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.

[16] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[17] G. Tesauro, "Temporal difference learning and TD-gammon," *Commun. ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[18] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, 2015, Art. no. 529.

[19] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.

[20] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. 4th Int. Conf. Learn. Representations*, 2016.

[21] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6351–6361.

[22] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 385–394.

[23] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Hoboken, NJ, USA: Wiley, 2011.

[24] M. Broadie, P. Glasserman, and Z. Ha, "Pricing american options by simulation using a stochastic mesh with optimized weights," in *Probabilistic Constrained Optimization*. Berlin, Germany: Springer, 2000, pp. 26–44.

[25] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 855–864.

[26] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[27] J. N. Tsitsiklis and B. Van Roy, "Analysis of temporal-diffference learning with function approximation," in *Proc. 9th Int. Conf. Neural Inf. Process. Syst.*, 1997, pp. 1075–1081.

[28] X. Yi, L. Hong, E. Zhong, N. N. Liu, and S. Rajan, "Beyond clicks: Dwell time for personalization," in *Proc. 8th ACM Conf. Recommender Syst.*, 2014, pp. 113–120.

[29] P. Deisenroth *et al.*, "A survey on policy search for robotics," *Found. Trends Robot.*, vol. 2, no. 1/2, pp. 1–142, 2013.

[30] A. Hill *et al.*, "Stable baselines," 2018. [Online]. Available: https://github.com/hill-a/stable-baselines

[31] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. Feb, pp. 281–305, 2012.

[32] S. Mahdavi, S. Khoshraftar, and A. An, "dynnode2vec: Scalable dynamic network embedding," in *Proc. IEEE Int. Conf. Big Data*, 2018, pp. 3762–3765.

[33] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.

[34] H. Davoudi, A. An, and G. Edall, "Content-based dwell time prediction model for news articles," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol.*, 2019, pp. 226–233.



Heidar Davoudi received the PhD degree in computer science from the Department of Electrical Engineering and Computer Science, York University, Canada. He is an assistant professor at Ontario Tech University, Canada. Before joining Ontario Tech, he was a postdoctoral reseach fellow at the University of Waterloo, Waterloo, Canada. His research interests include data mining, machine learning, user modeling for acquisition, and engagement. He has ongoing collaborations with IBM Canada, AT&T and The Globe and Mail. His research has been published in SIGKDD, SIAM SDM, NAACL-HLT, and PAKDD.

Authorized licensed use limited to: Ontario Tech University. Downloaded on November 10,2026 at 19:57:42 UTC from IEEE Xplore. Restrictions apply.



Zana Rashidi received the BSc degree in computer engineering from the Sharif University of Technology, Tehran. He is currently working toward the MSc degree in computer science at York University, Toronto. His research focuses on both theoretical and practical aspects of machine learning, especially reinforcement learning. His research has been published in the top-tier machine learning venue, ICLR.



Aijun An received the PhD degree in computer science from the University of Regina, Regina, Canada. She is a professor of computer science at York University, Canada. Her main research area is data mining. She has worked on data stream mining, high utility pattern mining, sentiment and emotion analysis from text, recommender systems, parallel and distributed deep learning, and graph analytics. She has published extensively in various well-respected journals and conferences in data mining, databases, and information retrieval. She leads the Big Data Research, Analytics and Information Network (BRIAN) Alliance, a research network involving four universities and more than a dozen private and public sector partners.



Morteza Zihayat is an assistant professor with the School of Information Technology Management (ITM), Ryerson University, Toronto, Canada from 2016 and IBM CAS Faculty fellow from 2018. Before joining ITM, he was a postdoctoral fellow at the University of Toronto, Toronto, Canada. His research interests include big data analytics and machine learning. He has been awarded multiple research Grants, including NSERC Discovery Grant, NSERC Engage, and MITACS. He has ongoing collaborations with industry, including IBM Canada, The Globe and Mail and AT&T. His research has been published in top-tier data mining and data management venues.



Gordon Edall is the director of Globe Labs at The Globe and Mail as well as the corporate security officer for Canada's national newspaper. In his innovation role, he heads up a program with a joint mandate to accelerate innovation by enabling entrepreneurial employees to do what they think needs done while also tapping into the startup ecosystem to find up and coming companies to partner with. Prior to that work, he worked in the data science group at The Globe And Mail and managed the build out of an advanced analytic stack including the predictive modeling and recommendation systems the company relies on today.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

Authorized licensed use limited to: Ontario Tech University. Downloaded on November 10,2026 at 19:57:42 UTC from IEEE Xplore. Restrictions apply.