Predicting Long-Term Impact of CQA Posts: A Comprehensive Viewpoint

Yuan Yao
State Key Laboratory for Novel Software Technology, China
yyao@smail.nju.edu.cn

Hanghang Tong
Arizona State University, USA
htong6@asu.edu

Feng Xu and Jian Lu
State Key Laboratory for Novel Software Technology, China
{xf,lj}@nju.edu.cn

ABSTRACT

Community Question Answering (CQA) sites have become valuable platforms to create, share, and seek a massive volume of human knowledge. How can we spot an insightful question that would inspire massive further discussions in CQA sites? How can we detect a valuable answer that benefits many users? The long-term impact (e.g., the size of the population a post benefits) of a question/answer post is the key quantity to answer these questions. In this paper, we aim to predict the long-term impact of questions/answers shortly after they are posted in the CQA sites. In particular, we propose a family of algorithms for the prediction problem by modeling three key aspects, i.e., non-linearity, question/answer coupling, and dynamics. We analyze our algorithms in terms of optimality, correctness, and complexity. We conduct extensive experimental evaluations on two real CQA data sets to demonstrate the effectiveness and efficiency of our algorithms.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database applications—*Data mining*

Keywords

Question answering; long-term impact; impact correlation

1. INTRODUCTION

Community Question Answering (CQA) sites, such as Stack Overflow[1], Yahoo! Answers[2], AnswerBag[3], and Ask.com[4], have become valuable platforms to create, share, and seek a massive volume of human knowledge. How can we spot an insightful question that would inspire massive further discussions in these CQA sites? How can we detect an valuable answer that benefits many users? The long-term impact of a question/answer post, which is the size of population it benefits in total, is the key quantity to answer these questions. *How can we predict the Long-term Impact of a Post (either a question or an answer) shortly after it is posted on the CQA site?* This, which we refer to as the *LIP* problem in this paper, is an essential task for the prosperity and sustainability of the CQA ecosystem that benefits all types of its users, including the information producers, spreaders, and consumers.

Despite its importance, it is not an easy task to predict the long-term impact of question/answer posts. Consequently, there are very few dedicated, existing tools for this problem (see Section 6 for a review). We summarize two major challenges below.

The first challenge lies in the *multi-aspect* of the long-term impact of question/answer posts. The user rating/voting mechanism within most of the CQA sites often provides a good measure of the long-term impact of CQA posts. For example, the voting score in Stack Overflow directly tells how many site users find the corresponding question/answer is beneficial to him/her. This naturally leads us to cast the LIP problem as a (supervised) data mining problem. Nonetheless, this problem has its own characteristics, making any off-the-shelf data mining algorithm sub-optimal for this problem. To be specific, while each of the following aspects might affect the long-term impact of question/answer posts, they require different treatments in the data mining algorithms. How can we build a comprehensive model to capture all these aspects to maximally boost the prediction accuracy?

- *A1. Non-linearity.* Both the contextual features (e.g., the reputation of the user who issues the question, etc.) and the content of the post (e.g., keywords, etc.) might affect its long-term impact; and the effect of each feature might be beyond the simple linear-relationship.
- *A2. Coupling between questions and answers.* Intuitively (which was also confirmed in our tech report [29]), the long-term impact of the questions might be correlated with that of its associated answers. Yet, the questions and answers may reside in different feature spaces.
- *A3. The dynamics (of training data sets).* While CQA sites usually offer a large size of training data set, it may arrive in a dynamic (stream-like) way for the mining algorithms.

The second challenge is the *computation*. Each of the above aspects (non-linearity, the question-answer coupling, and the dynamics) will add the extra complexity into the mining process. For example, while many machine learning algorithms (e.g., kernel regression, support vector regression, etc.) exist to capture the non-linearity between the features and outputs (long-term impact score in our case), they typically require at least $O(n^2)$ in time and space complexity, where n is the total number of the training examples.

[1]http://stackoverflow.com/
[2]http://answers.yahoo.com/
[3]http://www.answerbag.com/
[4]http://www.ask.com/

KDD'14, August 24–27, 2014, New York, NY, USA.

http://dx.doi.org/10.1145/2623330.2623649.

Moreover, when the new training examples arrive in a stream, ever-growing fashion, even an $O(n)$ algorithm might be too expensive. How can we make our prediction algorithms *scalable* to millions of CQA posts and *adaptive* to the newly arrived training examples over time?

In this paper, we aim to address both challenges by proposing a family of algorithms for predicting the long-term impact of question/answer posts. Our algorithms enjoy three key advantages. First, they are *comprehensive* in the sense that our model naturally captures all the above three key aspects (i.e., non-linearity, coupling, and dynamics) that matter with the long-term impact of a post. Second, they are *flexible* and *general*, being able to handle the special cases where only a fraction of these aspects are prominent. For example, in some CQA sites, we might be only interested in the prediction of answers posts (such as Yahoo! Answers), and/or the features may have a linear effect on the post impact, etc. Third, they are *scalable* and *adaptive* to the newly arrived examples. For example, one of our algorithms (LIP-KIMAA) has a sub-linear complexity in both time and space. On the Stack Overflow data set with more than 3 million posts, this LIP-KIMAA algorithm can build and update our model in seconds, while straightforward approaches would take hours.

The main contributions of this paper are summarized as follows:

- *A family of novel algorithms* for the prediction of the long-term impact of questions and answers in CQA sites.
- *Proofs and analysis*, showing the optimality, correctness, and computational efficiency, as well as the intrinsic relationship among different algorithms.
- *Extensive empirical evaluations*, demonstrating effectiveness and efficiency of our algorithms. For example, compared with alternative choices (e.g., KRR [25]), one of our proposed algorithm (LIP-KIMAA) (1) leads up to 35.8% improvement in prediction accuracy; (2) and is up to 390x faster while enjoying *sub-linear* scalability.

The rest of the paper is organized as follows. Section 2 describes the problem definition. Section 3 presents the proposed algorithms. Section 4 discusses some variants. Section 5 presents the experimental results. Section 6 reviews related work, and Section 7 concludes the paper.

2. PROBLEM STATEMENT

In this section, we first define the LIP problem, and then present its solution space to illustrate the relationship between our proposed algorithms and the existing work.

2.1 Problem Definitions

Table 1 lists the main symbols we use throughout the paper. For convenience, we use bold capital letters for existing matrices/vectors at time t, and bold lower case letters for newly arrived matrices/vectors at time $t+1$. We use superscript (i.e., q or a) to distinguish questions and answers, and use subscript (i.e., t, $t+1$, etc.) to indicate time. For example, we use $\mathbf{F}_t^q$ to denote the feature matrix for questions at time t, and $\mathbf{f}_{t+1}^q$ to denote the feature matrix of newly arrived questions at time $t+1$. Each row of $\mathbf{F}_t^q$ and $\mathbf{f}_{t+1}^q$ contains the feature vector for the corresponding question. Similarly, we use $\mathbf{Y}_t^q$ to denote the vector of impact scores at time t, and $\mathbf{y}_{t+1}^q$ to denote the vector of impact scores from new questions at time $t+1$. Following conventions, we use calligraphic letter $\mathcal{K}_t^q$ and $\mathcal{K}_t^a$ to denote the kernel matrix for questions and answers at time t. We will omit the subscript when the meaning of matrices/vectors is clear in the context. We use the row-normalized $n_q \times n_a$ matrix $\mathbf{M}_t$ to denote the association between questions and answers at time t, where non-zero element $\mathbf{M}_t(i,j)$ indicates that the j^{th} answer belongs to the i^{th} question. Thus, the matrix $\mathbf{M}_t$ is sparse since it contains only n^a non-zero elements. We also use $\mathbf{F}_t^q(i,:)$ to represent the i^{th} row of matrix $\mathbf{F}_t^q$, $\mathbf{Y}_t^q(i)$ to represent the i^{th} element of vector $\mathbf{Y}_t^q$, and $(\mathbf{F}_t^q)'$ to represent the transpose of $\mathbf{F}_t^q$.

Table 1: Symbols.

Symbol	Definition and Description
$\mathbf{F}_t^q, \mathbf{F}_t^a$	the features for existing questions/answers at time t
$\mathbf{f}_{t+1}^q, \mathbf{f}_{t+1}^a$	the features of new questions/answers at time $t+1$
$\mathcal{K}_t^q, \mathcal{K}_t^a$	the kernel matrix of $\mathbf{F}_t^q$ and $\mathbf{F}_t^a$
$\mathbf{k}_{t+1}^q, \mathbf{h}_{t+1}^q$	the kernel matrix of new questions at time $t+1$
$\mathbf{k}_{t+1}^a, \mathbf{h}_{t+1}^a$	the kernel matrix of new answers at time $t+1$
$\mathbf{U}_t^q, \Lambda_t^q$	the low-rank matrices to approximate $\mathcal{K}_t^q$
$\mathbf{U}_t^a, \Lambda_t^a$	the low-rank matrices to approximate $\mathcal{K}_t^a$
$\mathbf{M}_t$	the row-normalized association matrix between existing questions and answers at time t
$\mathbf{m}_{t+1}$	the row-normalized association matrix between new questions and answers at time $t+1$
$\mathbf{Y}_t^q, \mathbf{Y}_t^a$	the impact score for existing questions/answers at time t
$\mathbf{y}_{t+1}^q, \mathbf{y}_{t+1}^a$	the impact score for new questions/answers at time $t+1$
n^q, n^a	the number of existing questions/answers at time t
i^q, i^a	the number of new questions/answers at time $t+1$
d	the feature dimension
r	the rank of $\mathbf{U}_t^q$, Λ_t^q, $\mathbf{U}_t^a$, and Λ_t^a
th	the threshold for the filtering step

Based on the above notations, we define the LIP problem in its static form as:

PROBLEM 1. *Static LIP Problem*

Given: *the question/answer feature matrix $\mathbf{F}^q$/$\mathbf{F}^a$, the question/answer impact vector $\mathbf{Y}^q$/$\mathbf{Y}^a$, and the association matrix $\mathbf{M}$;*

Output: *the impact of new questions and their answers.*

In real CQA sites where questions and answers continuously arrive, we need to update the model to keep it up-to-date. To this end, we define the following dynamic form of the LIP problem:

PROBLEM 2. *Dynamic LIP Problem*

Given: *the question/answer feature matrix $\mathbf{F}_t^q$/$\mathbf{F}_t^a$ and the newly arrived question/answer feature matrix $\mathbf{f}_{t+1}^q$/$\mathbf{f}_{t+1}^a$, the question/answer impact vector $\mathbf{Y}_t^q$/$\mathbf{Y}_t^a$ and the newly arrived question/answer impact vector $\mathbf{y}_{t+1}^q$/$\mathbf{y}_{t+1}^a$, as well as the association matrix $\mathbf{M}_t$ and $\mathbf{m}_{t+1}$;*

Output: *the impact of new questions and their answers.*

2.2 Solution Space

Solution Space. Let us first define the solution space of the LIP problem, which is represented by a genealogy graph in Fig. 1. In this paper, we consider three key aspects that matter with the prediction performance, including (a) whether the predication models are linear or non-linear; (b) whether we treat the prediction of the questions and the answers separately (single) or jointly (coupling); and (c) whether the prediction is static or dynamic. Based on these three aspects, we could have different variants of the prediction algorithms for the LIP problem, whose intrinsic relationship is also summarized in Fig. 1. In the figure, we use the letter K, M, and I to denote non-linearity, coupling, and dynamics, respectively.

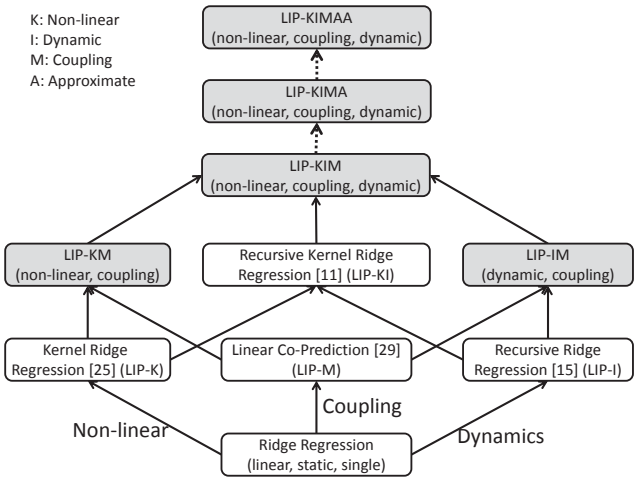


Figure 1: The solution space of LIP problem. Shaded boxes are proposed algorithms; and white boxes are existing work.

For example, LIP-KIM means that our model is non-linear and dynamic, and it jointly predict the long-term impact of questions and answers; LIP-K means that our prediction model is non-linear and static, and it treats questions and answers separately, etc.

In Fig. 1, each upward solid arrow makes the model more comprehensive by modeling more aspects in the prediction algorithms, and each dashed arrow makes the algorithms more scalable. For example, starting with the ridge regression algorithm in the bottom layer, we have the kernel ridge regression (KRR) [25] by incorporating non-linearity, and we have the recursive ridge regression [15] by incorporating dynamics. If we incorporate both non-linearity and dynamics, we have the recursive kernel ridge regression algorithm (RKRR) [11] in the third layer.

Preliminaries. In Fig. 1, we use shaded boxes to indicate the algorithms proposed in this paper, and while boxes to indicate existing work. Before presenting the proposed algorithms in the next section, we first briefly review some existing work (e.g., white boxes), which serves as the building blocks of our proposed algorithms.

(A) Linear Co-Prediction. In our tech report [29], we proposed a regularized optimization formulation to jointly predict the voting score of questions and answers

$$\min_{\boldsymbol{\alpha}^q,\boldsymbol{\alpha}^a} \quad \sum_{i=1}^{n^q}(\mathbf{F}^q(i,:)\boldsymbol{\alpha}^q - \mathbf{Y}^q(i))^2 + \sum_{i=1}^{n^a}(\mathbf{F}^a(i,:)\boldsymbol{\alpha}^a - \mathbf{Y}^a(i))^2 + \theta\sum_{i=1}^{n^q}(\mathbf{F}^q(i,:)\boldsymbol{\alpha}^q - \mathbf{M}(i,:)\mathbf{F}^a\boldsymbol{\alpha}^a)^2 + \lambda(\|\boldsymbol{\alpha}^q\|_2^2 + \|\boldsymbol{\alpha}^a\|_2^2) \quad (1)$$

where parameters λ and θ are used to control regularization and the importance of the coupling between questions and answers, respectively.

(B) Kernel Ridge Regression. In order to capture the non-linearity between the features and outputs, a natural choice is to user kernelized methods. Take question impact prediction as an example, the so-called kernel ridge regression [25] aims to estimate a coefficient $\boldsymbol{\beta}^q$ as follows

$$\min_{\boldsymbol{\beta}^q} \quad \sum_{i=1}^{n^q}(\mathcal{K}^q(i,:)\boldsymbol{\beta}^q - \mathbf{Y}^q(i))^2 + \lambda(\boldsymbol{\beta}^q)'\mathcal{K}^q\boldsymbol{\beta}^q \quad (2)$$

where $\mathcal{K}^q$ is the kernel matrix of $\mathbf{F}^q$.

3. THE PROPOSED ALGORITHMS

In this section, we propose our solutions for the LIP problem. We start with presenting two algorithms for Problem 1 (subsection 3.1) and Problem 2 (subsection 3.2), respectively; and then address the computational challenges (subsection 3.3-3.4).

3.1 LIP-KM Algorithm for Problem 1

Here, we address the static LIP problem (Problem 1). We propose an algorithm (LIP-KM) to capture both the non-linearity and the coupling aspects.

For the non-linear aspect, a natural choice is to kernelize a linear prediction model (e.g., linear ridge regression). Recall that kernel method aims to produce non-linear versions of linear learning algorithms by mapping the data points into a high-dimensional Hilbert space $\mathcal{H}$ with a non-linear function ϕ [4]. The key idea behind kernel methods is to use the kernel functions to replace the inner-product operations in the high-dimensional Hilbert space $\mathcal{H}$, and such replacement can be ensured by Mercer's Condition [6]. In other words, for two data points $\mathbf{F}(i,:)$ and $\mathbf{F}(j,:)$, the inner product of $\phi(\mathbf{F}(i,:))$ and $\phi(\mathbf{F}(j,:))$ in the Hilbert space $\mathcal{H}$ can be directly computed by a Mercer kernel $\kappa(\mathbf{F}(i,:),\mathbf{F}(j,:))$

$$\kappa(\mathbf{F}(i,:),\mathbf{F}(j,:)) =< \phi(\mathbf{F}(i,:)),\phi(\mathbf{F}(j,:)) >= \phi(\mathbf{F}(i,:))\phi(\mathbf{F}(j,:))' \quad (3)$$

where $< \cdot,\cdot >$ indicates the inner product in $\mathcal{H}$. As we can see from Eq. (3), we can derive the non-linear models without any explicit knowledge of either ϕ or $\mathcal{H}$. Common kernel functions include Gaussian kernel, polynomial kernel, cosine kernel, etc.

For the coupling aspect, LIP-KM imposes a so-called *impact consistency* on the prediction space by requiring the predicted impact of a question to be close to that of its answer (see our tech report [29] for the detailed explanations about its rationality).

Putting the non-linearity and coupling aspects together, we have the following optimization formulation for Problem 1

$$\min_{\boldsymbol{\beta}^q,\boldsymbol{\beta}^a} \quad \sum_{i=1}^{n^q}(\mathcal{K}^q(i,:)\boldsymbol{\beta}^q - \mathbf{Y}^q(i))^2 + \sum_{i=1}^{n^a}(\mathcal{K}^a(i,:)\boldsymbol{\beta}^a - \mathbf{Y}^a(i))^2 + \theta\sum_{i=1}^{n^q}(\mathcal{K}^q(i,:)\boldsymbol{\beta}^q - \mathbf{M}(i,:)\mathcal{K}^a\boldsymbol{\beta}^a)^2 + \lambda((\boldsymbol{\beta}^q)'\mathcal{K}^q\boldsymbol{\beta}^q + (\boldsymbol{\beta}^a)'\mathcal{K}^a\boldsymbol{\beta}^a) \quad (4)$$

where θ is a weight parameter to control the importance of coupling, λ is a regularization parameter, and $\mathcal{K}^q$ and $\mathcal{K}^a$ are the kernel matrices of $\mathbf{F}^q$ and $\mathbf{F}^a$, respectively. $\mathcal{K}^q$ and $\mathcal{K}^a$ can computed as $\mathcal{K}^q(i,j) = \kappa(\mathbf{F}^q(i,:),\mathbf{F}^q(j,:))$, and $\mathcal{K}^a(i,j) = \kappa(\mathbf{F}^a(i,:),\mathbf{F}^a(j,:))$.

Eq. (4) can be solved by the closed-form solution

$$\begin{aligned}\boldsymbol{\beta} &= \arg\min_{\boldsymbol{\beta}^q,\boldsymbol{\beta}^a} \|\mathcal{K}^q\boldsymbol{\beta}^q - \mathbf{Y}^q\|_2^2 + \|\mathcal{K}^a\boldsymbol{\beta}^a - \mathbf{Y}^a\|_2^2 + \theta\|\mathcal{K}^q\boldsymbol{\beta}^q - \mathbf{M}\mathcal{K}^a\boldsymbol{\beta}^a\|_2^2 \\ &\quad +\lambda((\boldsymbol{\beta}^q)'\mathcal{K}^q\boldsymbol{\beta}^q + (\boldsymbol{\beta}^a)'\mathcal{K}^a\boldsymbol{\beta}^a) \\ &= \begin{bmatrix}(\theta+1)\mathcal{K}^q + \lambda\mathbf{I} & -\theta\mathbf{M}\mathcal{K}^a \\ -\theta\mathbf{M}'\mathcal{K}^q & \mathcal{K}^a + \theta\mathbf{M}'\mathbf{M}\mathcal{K}^a + \lambda\mathbf{I}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{Y}^q \\ \mathbf{Y}^a\end{bmatrix} \quad (5)\end{aligned}$$

where $\boldsymbol{\beta} = [\boldsymbol{\beta}^q;\boldsymbol{\beta}^a]$. Once the coefficient vectors $\boldsymbol{\beta}^q$ and $\boldsymbol{\beta}^a$ are inferred from the above equation, the impact of questions/answers can then be predicted as

$$\hat{\mathbf{Y}}^q_{test} = \mathcal{K}^q_{test}\boldsymbol{\beta}^q, \quad \hat{\mathbf{Y}}^a_{test} = \mathcal{K}^a_{test}\boldsymbol{\beta}^a \quad (6)$$

where the kernel matrices on the test set $\mathbf{F}^q_{test}$ and $\mathbf{F}^a_{test}$ can be computed as $\mathcal{K}^q_{test}(i,j) = \kappa(\mathbf{F}^q_{test}(i,:),\mathbf{F}^q(j,:))$, $\mathcal{K}^a_{test}(i,j) = \kappa(\mathbf{F}^a_{test}(i,:),\mathbf{F}^a(j,:))$.

Algorithm Analysis. Let us analyze the effectiveness and efficiency of the LIP-KM algorithm (i.e., Eq. (5)). We first summarize the optimality of LIP-KM in the following lemma, which states that LIP-KM finds (at least) a local minimum for the static LIP problem.

LEMMA 1. **Optimality of LIP-KM**. *Eq.* (5) *finds a local minimum for Eq.* (4)*.*

PROOF. Omitted for brevity. □

Algorithm 1 The LIP-KIM Algorithm.

Input: $\mathbf{S}_t^{-1}, \boldsymbol{\beta}_t, \mathbf{F}_t^q, \mathbf{f}_{t+1}^q, \mathbf{F}_t^a, \mathbf{f}_{t+1}^a, \mathbf{y}_{t+1}^q, \mathbf{y}_{t+1}^a, \mathbf{M}_t$ and $\mathbf{m}_{t+1}$
Output: $\boldsymbol{\beta}_{t+1}, \mathbf{S}_{t+1}^{-1}$

1: compute the new kernels $\mathbf{k}_{t+1}^q$, $\mathbf{h}_{t+1}^q$, $\mathbf{k}_{t+1}^a$ and $\mathbf{h}_{t+1}^a$ in Eq. (7);
2: compute $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{S}_3$, $\mathbf{D}$, and $\mathbf{E}_1$ as Eq. (9) - (13);
3: update $\mathbf{S}_{t+1}^{-1}$ as $\mathbf{E}_1'\begin{bmatrix} \mathbf{S}_t^{-1} + \mathbf{S}_t^{-1}\mathbf{S}_1\mathbf{D}\mathbf{S}_2\mathbf{S}_t^{-1} & -\mathbf{S}_t^{-1}\mathbf{S}_1\mathbf{D} \\ -\mathbf{S}_3^{-1}\mathbf{S}_2\mathbf{S}_t^{-1}(\mathbf{I} + \mathbf{S}_1\mathbf{D}\mathbf{S}_2\mathbf{S}_t^{-1}) & \mathbf{D} \end{bmatrix}\mathbf{E}_1$;
4: update $\boldsymbol{\beta}_{t+1}$ as $\mathbf{E}_1\begin{bmatrix} \boldsymbol{\beta}_t + \mathbf{S}_t^{-1}\mathbf{S}_1\mathbf{D}(\mathbf{S}_2\boldsymbol{\beta}_t - [\mathbf{y}_{t+1}^q; \mathbf{y}_{t+1}^a]) \\ -\mathbf{S}_3^{-1}\mathbf{S}_2(\boldsymbol{\beta}_t + \mathbf{S}_t^{-1}\mathbf{S}_1\mathbf{D}\mathbf{S}_2\boldsymbol{\beta}_t) + \mathbf{D}[\mathbf{y}_{t+1}^q; \mathbf{y}_{t+1}^a] \end{bmatrix}$;
5: **return** $\boldsymbol{\beta}_{t+1}, \mathbf{S}_{t+1}^{-1}$;

Next, we summarize the time complexity and space complexity of LIP-KM in the following lemma, which basically states that LIP-KM requires $O((n^q + n^a)^3)$ time and $O((n^q + n^a)^2)$ space.

LEMMA 2. **Complexity of LIP-KM**. *The time complexity of Eq.* (5) *is* $O((n^q + n^a)^3 + (n^q)^2 d + (n^a)^2 d)$*; the space complexity of Eq.* (5) *is* $O((n^q + n^a)^2 + (n^q + n^a)d)$.

PROOF. Omitted for brevity. □

3.2 LIP-KIM Algorithm for Problem 2

Here, we address the dynamic LIP problem (Problem 2). We present the LIP-KIM algorithm to incrementally update the model in Eq. (5). The basic idea of LIP-KIM is to incorporate the dynamic aspect into LIP-KM, and therefore it is adaptive to newly arrived training examples.

When new questions and answers arrive at time $t+1$, we first need to compute the new kernel matrices $\mathcal{K}_{t+1}^q$ and $\mathcal{K}_{t+1}^a$ which are as follows

$$\mathcal{K}_{t+1}^q = \begin{bmatrix} \mathcal{K}_t^q & (\mathbf{k}_{t+1}^q)' \\ \mathbf{k}_{t+1}^q & \mathbf{h}_{t+1}^q \end{bmatrix}, \quad \mathcal{K}_{t+1}^a = \begin{bmatrix} \mathcal{K}_t^a & (\mathbf{k}_{t+1}^a)' \\ \mathbf{k}_{t+1}^a & \mathbf{h}_{t+1}^a \end{bmatrix} \tag{7}$$

where the kernel matrices involving the newly arrived examples can be computed as: $\mathbf{k}_{t+1}^q(i,j) = \kappa(\mathbf{f}_{t+1}^q(i,:), \mathbf{F}_t^q(j,:))$, $\mathbf{h}_{t+1}^q(i,j) = \kappa(\mathbf{f}_{t+1}^q(i,:), \mathbf{f}_{t+1}^q(j,:))$, $\mathbf{k}_{t+1}^a(i,j) = \kappa(\mathbf{f}_{t+1}^a(i,:), \mathbf{F}_t^a(j,:))$, and $\mathbf{h}_{t+1}^a(i,j) = \kappa(\mathbf{f}_{t+1}^a(i,:), \mathbf{f}_{t+1}^a(j,:))$.

To simplify the algorithm description, let us introduce the following matrices

$$\mathbf{S}_t = \begin{bmatrix} (\theta+1)\mathcal{K}_t^q + \lambda\mathbf{I} & -\theta\mathbf{M}_t\mathcal{K}_t^a \\ -\theta\mathbf{M}_t'\mathcal{K}_t^q & \mathcal{K}_t^a + \theta\mathbf{M}_t'\mathbf{M}_t\mathcal{K}_t^a + \lambda\mathbf{I} \end{bmatrix} \tag{8}$$

$$\mathbf{S}_1 = \begin{bmatrix} (\theta+1)(\mathbf{k}_{t+1}^q)' & -\theta\mathbf{M}_t(\mathbf{k}_{t+1}^a)' \\ -\theta\mathbf{M}_t'(\mathbf{k}_{t+1}^q)' & (\mathbf{k}_{t+1}^a)' + \theta\mathbf{M}_t'\mathbf{M}_t(\mathbf{k}_{t+1}^a)' \end{bmatrix} \tag{9}$$

$$\mathbf{S}_2 = \begin{bmatrix} (\theta+1)\mathbf{k}_{t+1}^q & -\theta\mathbf{m}_{t+1}\mathbf{k}_{t+1}^a \\ -\theta\mathbf{m}_{t+1}'\mathbf{k}_{t+1}^q & \mathbf{k}_{t+1}^a + \theta\mathbf{m}_{t+1}'\mathbf{m}_{t+1}\mathbf{k}_{t+1}^a \end{bmatrix} \tag{10}$$

$$\mathbf{S}_3 = \begin{bmatrix} (\theta+1)\mathbf{h}_{t+1}^q + \lambda\mathbf{I} & -\theta\mathbf{m}_{t+1}\mathbf{h}_{t+1}^a \\ -\theta\mathbf{m}_{t+1}'\mathbf{h}_{t+1}^q & \mathbf{h}_{t+1}^a + \theta\mathbf{m}_{t+1}'\mathbf{m}_{t+1}\mathbf{h}_{t+1}^a + \lambda\mathbf{I} \end{bmatrix} \tag{11}$$

$$\mathbf{D} = (\mathbf{S}_3 - \mathbf{S}_2\mathbf{S}_t^{-1}\mathbf{S}_1)^{-1} \tag{12}$$

With these extra notations, we present our LIP-KIM algorithm for solving Problem 2, which is summarized in Alg. 1. As we can see from the algorithm, we re-use $\mathbf{S}_t^{-1}$ and $\boldsymbol{\beta}_t$ from previous computations. Therefore, we also need to update $\mathbf{S}_{t+1}^{-1}$ and $\boldsymbol{\beta}_{t+1}$ for future iterations. After we compute the new kernels as well as the matrices (i.e., $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{S}_3$ and $\mathbf{D}$) that are based on the new kernels, we can update the model in Steps 3-4. In these two steps, $\mathbf{E}_1$ stands for a permutation matrix to exchange the corresponding rows/columns

$$\mathbf{E}_1 = \begin{bmatrix} \mathbf{I}_{n_q \times n_q} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{i_q \times i_q} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{n_a \times n_a} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{i_a \times i_a} \end{bmatrix} \tag{13}$$

where i^q and i^a denote the number of questions and answers arrived at time $t+1$, respectively.

Algorithm Analysis. The correctness of LIP-KIM is summarized in the following theorem, which states that LIP-KIM can find the same coefficients (i.e., $\boldsymbol{\beta}_{t+1}$) as if we apply the LIP-KM algorithm whenever we have new training examples.

THEOREM 1. **Correctness of LIP-KIM**. *Let* $\boldsymbol{\beta}_{t+1}^*$ *be the output of Eq.* (5) *at time* $t+1$*, and* $\boldsymbol{\beta}_{t+1}$ *be the output of Alg. 1 updated from time* t *to* $t+1$*, we have that* $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_{t+1}^*$.

PROOF. Notice that Eq. (5) can be re-written as

$$\boldsymbol{\beta}_t = \mathbf{S}_t^{-1}\begin{bmatrix} \mathbf{Y}_t^q \\ \mathbf{Y}_t^a \end{bmatrix}$$

Therefore, we need to first prove the update procedure for $\mathbf{S}_t^{-1}$ (i.e., Step 3 in Alg. 1). $\mathbf{S}_{t+1}$ can be written as

$$\mathbf{S}_{t+1} = \begin{bmatrix} (\theta+1)\mathcal{K}_{t+1}^q + \lambda\mathbf{I} & -\theta\mathbf{M}_{t+1}\mathcal{K}_{t+1}^a \\ -\theta\mathbf{M}_{t+1}'\mathcal{K}_{t+1}^q & \mathcal{K}_{t+1}^a + \theta\mathbf{M}_{t+1}'\mathbf{M}_{t+1}\mathcal{K}_{t+1}^a + \lambda\mathbf{I} \end{bmatrix}$$

where $\mathbf{M}_{t+1} = [\mathbf{M}_t, \mathbf{0}; \mathbf{0}, \mathbf{m}_{t+1}]$, and $\mathcal{K}_{t+1}^q$ and $\mathcal{K}_{t+1}^a$ are shown in Eq. (7).

By exchanging the rows and columns of $\mathbf{S}_{t+1}$ (i.e., moving all the features at time t into the upper-left corner), we have

$$\mathbf{S}_{t+1} = \mathbf{E}_1\begin{bmatrix} \mathbf{S}_t & \mathbf{S}_1 \\ \mathbf{S}_2 & \mathbf{S}_3 \end{bmatrix}\mathbf{E}_1' \tag{14}$$

where $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{S}_3$ and $\mathbf{E}_1$ are specified in Eq. (9), Eq. (10), Eq. (11) and Eq. (13), respectively.

Applying Matrix Inversion Lemma [24, 14] to Eq. (14), we have that

$$\begin{aligned} \mathbf{S}_{t+1}^{-1} &= \left(\mathbf{E}_1\begin{bmatrix} \mathbf{S}_t & \mathbf{S}_1 \\ \mathbf{S}_2 & \mathbf{S}_3 \end{bmatrix}\mathbf{E}_1'\right)^{-1} \\ &= \mathbf{E}_1\begin{bmatrix} \mathbf{S}_t^{-1} + \mathbf{S}_t^{-1}\mathbf{S}_1\mathbf{D}\mathbf{S}_2\mathbf{S}_t^{-1} & -\mathbf{S}_t^{-1}\mathbf{S}_1\mathbf{D} \\ -\mathbf{S}_3^{-1}\mathbf{S}_2\mathbf{S}_t^{-1}(\mathbf{I} + \mathbf{S}_1\mathbf{D}\mathbf{S}_2\mathbf{S}_t^{-1}) & \mathbf{D} \end{bmatrix}\mathbf{E}_1' \end{aligned}$$

where $\mathbf{D}$ is specified in Eq. (12), and $\mathbf{E}_1^{-1} = \mathbf{E}_1'$.

Based on the updated $\mathbf{S}_{t+1}^{-1}$, we have that

$$\begin{aligned} \boldsymbol{\beta}_{t+1}^* &= \mathbf{S}_{t+1}^{-1}\begin{bmatrix} \mathbf{Y}_t^q \\ \mathbf{y}_{t+1}^q \\ \mathbf{Y}_t^a \\ \mathbf{y}_{t+1}^a \end{bmatrix} \\ &= \mathbf{E}_1\begin{bmatrix} \mathbf{S}_t^{-1} + \mathbf{S}_t^{-1}\mathbf{S}_1\mathbf{D}\mathbf{S}_2\mathbf{S}_t^{-1} & -\mathbf{S}_t^{-1}\mathbf{S}_1\mathbf{D} \\ -\mathbf{S}_3^{-1}\mathbf{S}_2\mathbf{S}_t^{-1}(\mathbf{I} + \mathbf{S}_1\mathbf{D}\mathbf{S}_2\mathbf{S}_t^{-1}) & \mathbf{D} \end{bmatrix}\begin{bmatrix} \mathbf{Y}_t^q \\ \mathbf{Y}_t^a \\ \mathbf{y}_{t+1}^q \\ \mathbf{y}_{t+1}^a \end{bmatrix} \\ &= \mathbf{E}_1\begin{bmatrix} \boldsymbol{\beta}_t + \mathbf{S}_t^{-1}\mathbf{S}_1\mathbf{D}(\mathbf{S}_2\boldsymbol{\beta}_t - [\mathbf{y}_{t+1}^q; \mathbf{y}_{t+1}^a]) \\ -\mathbf{S}_3^{-1}\mathbf{S}_2(\boldsymbol{\beta}_t + \mathbf{S}_t^{-1}\mathbf{S}_1\mathbf{D}\mathbf{S}_2\boldsymbol{\beta}_t) + \mathbf{D}[\mathbf{y}_{t+1}^q; \mathbf{y}_{t+1}^a] \end{bmatrix} \\ &= \boldsymbol{\beta}_{t+1} \end{aligned}$$

which completes the proof. □

The time complexity and space complexity of our LIP-KIM is summarized in the following lemma, which basically states that LIP-KIM requires $O((n^q + n^a)^2)$ time and $O((n^q + n^a)^2)$ space.

LEMMA 3. **Complexity of LIP-KIM**. *The time complexity of Alg. 1 is* $O((n^q+n^a)^2(i^q+i^a)+(n^q+n^a)(i^q+i^a)^2+(i^q+i^a)^3+(n^q i^q+n^a i^a+(i^q)^2+(i^a)^2)d)$*; the space complexity of Alg. 1 is* $O((n^q+n^a+i^q+i^a)^2+(n^q+n^a+i^q+i^a)d)$.

PROOF. Omitted for brevity. □

Since i^q and i^a are usually much smaller than n^q and n^a, and the feature dimension d is a fixed constant, the time and space complexity of LIP-KIM can be re-written as $O((n^q+n^a)^2)$. Compared with LIP-KM which is cubic in time, LIP-KIM is much more efficient. However, it is still quadratic wrt the total number of the training examples. In the next two subsections, we propose two approximate algorithms to further speed-up the computation.

3.3 LIP-KIMA Algorithm

The reason that LIP-KIM is quadratic is that we need to maintain two kernel matrices of the size $n_q \times n_q$ and $n_a \times n_a$, respectively. In order to avoid quadratic cost for both time and space, we need an efficient way to approximate/compress the full kernel matrices and update them over time.

Take the kernel matrix for questions as an example. Notice that $\mathcal{K}_t^q$ is symmetric and semi-positive definite; therefore, we can approximate it by eigen-decomposition: $\mathcal{K}_t^q \approx \mathbf{U}_t^q \Lambda_t^q (\mathbf{U}_t^q)'$, where $\mathbf{U}_t^q$ is an $n^q \times r$ orthogonal matrix, and Λ_t^q is an $r \times r$ diagonal matrix whose entries are the largest r eigenvalues of $\mathcal{K}_t^q$. By doing so, we reduce the space cost from $O(n_q^2)$ to $O(n_q r)$.

When new questions arrive at time $t+1$, we have the new kernel matrix $\mathcal{K}_{t+1}^q$ as shown in Eq. (7). We approximate $\mathcal{K}_{t+1}^q$ by Nyström method [10] as

$$\begin{aligned}
\mathcal{K}_{t+1}^q &= \begin{bmatrix} \mathcal{K}_t^q & (\mathbf{k}_{t+1}^q)' \\ \mathbf{k}_{t+1}^q & \mathbf{h}_{t+1}^q \end{bmatrix} \\
&\approx \begin{bmatrix} \mathcal{K}_t^q \\ \mathbf{k}_{t+1}^q \end{bmatrix} (\mathcal{K}_t^q)^{-1} \begin{bmatrix} \mathcal{K}_t^q \\ \mathbf{k}_{t+1}^q \end{bmatrix}' \\
&\approx \begin{bmatrix} \mathcal{K}_t^q \\ \mathbf{k}_{t+1}^q \end{bmatrix} \mathbf{U}_t^q (\Lambda_t^q)^{-1} \Lambda_t^q (\Lambda_t^q)^{-1} (\mathbf{U}_t^q)' \begin{bmatrix} \mathcal{K}_t^q \\ \mathbf{k}_{t+1}^q \end{bmatrix}' \\
&= \begin{bmatrix} \mathbf{U}_t^q \\ \mathbf{k}_{t+1}^q \mathbf{U}_t^q (\Lambda_t^q)^{-1} \end{bmatrix} \Lambda_t^q \begin{bmatrix} \mathbf{U}_t^q \\ \mathbf{k}_{t+1}^q \mathbf{U}_t^q (\Lambda_t^q)^{-1} \end{bmatrix}' \\
&= \mathbf{X}_1 \Lambda_t^q \mathbf{X}_1' \qquad (15)
\end{aligned}$$

where we define $\mathbf{X}_1$ as the $(n^q+i^q) \times r$ matrix $[\mathbf{U}_t^q; \mathbf{k}_{t+1}^q \mathbf{U}_t^q (\Lambda_t^q)^{-1}]$.

To make the decomposition of $\mathcal{K}_{t+1}^q$ reusable for future updates, we need to find the eigen-decomposition form of $\mathcal{K}_{t+1}^q$. To this end, we first perform the Singular Value Decomposition (SVD) on $\mathbf{X}_1$ as $\mathbf{X}_1 = \mathbf{P}\Lambda_1^q \mathbf{Q}'$, where both $\mathbf{P}$ and $\mathbf{Q}$ are orthogonal. Next, we perform eigen-decomposition on an $r \times r$ matrix $\mathbf{X}_2 = \Lambda_1^q \mathbf{Q}' \Lambda_t^q \mathbf{Q} \Lambda_1^q$, that is, $\mathbf{X}_2 = \mathbf{V}\Lambda^q \mathbf{V}'$.

Based on the above two steps, we have the approximate eigen-decomposition of the new kernel matrix $\mathcal{K}_{t+1}^q$ as follows

$$\begin{aligned}
\mathcal{K}_{t+1}^q &\approx \mathbf{X}_1 \Lambda_t^q \mathbf{X}_1' \\
&= \mathbf{P}(\Lambda_1^q \mathbf{Q}' \Lambda_t^q \mathbf{Q} \Lambda_1^q)\mathbf{P}' \\
&= \mathbf{P}(\mathbf{V}\Lambda^q \mathbf{V}')\mathbf{P}' \\
&= \mathbf{U}_{t+1}^q \Lambda_{t+1}^q (\mathbf{U}_{t+1}^q)' \qquad (16)
\end{aligned}$$

where we define $\mathbf{U}_{t+1}^q = \mathbf{P}\mathbf{V}$ and $\Lambda_{t+1}^q = \Lambda^q$. Notice that $\mathbf{U}_{t+1}^q$ is orthogonal because both $\mathbf{P}$ and $\mathbf{V}$ are orthogonal.

We use the same approach to approximate and update the kernel matrix for answers: $\mathcal{K}_{t+1}^a \approx \mathbf{U}_{t+1}^a \Lambda_{t+1}^a (\mathbf{U}_{t+1}^a)'$. We further define the following notations to simplify the algorithm description

$$\begin{aligned}
\mathbf{U} &= [\mathbf{U}_{t+1}^q, \mathbf{0}; \mathbf{0}, \mathbf{U}_{t+1}^a] \\
\Lambda &= [\Lambda_{t+1}^q, \mathbf{0}; \mathbf{0}, \Lambda_{t+1}^a] \\
\mathbf{G} &= [\mathbf{I}, -\mathbf{M}_{t+1}; -\mathbf{M}_{t+1}', \mathbf{M}_{t+1}'\mathbf{M}_{t+1}] \qquad (17)
\end{aligned}$$

Then, we have the following approximation for the $\mathbf{S}_{t+1}$ matrix defined in Eq. (8)

$$\begin{aligned}
\mathbf{S}_{t+1} &= \begin{bmatrix} (\theta+1)\mathcal{K}_{t+1}^q + \lambda\mathbf{I} & -\theta\mathbf{M}_{t+1}\mathcal{K}_{t+1}^a \\ -\theta\mathbf{M}_{t+1}'\mathcal{K}_{t+1}^q & \mathcal{K}_{t+1}^a + \theta\mathbf{M}_{t+1}'\mathbf{M}_{t+1}\mathcal{K}_{t+1}^a + \lambda\mathbf{I} \end{bmatrix} \\
&= \lambda\mathbf{I} + (\mathbf{I}+\theta\mathbf{G}) \begin{bmatrix} \mathcal{K}_{t+1}^q & \mathbf{0} \\ \mathbf{0} & \mathcal{K}_{t+1}^a \end{bmatrix} \\
&\approx \lambda\mathbf{I} + \mathbf{U}\Lambda\mathbf{U}' + \theta\mathbf{G}\mathbf{U}\Lambda\mathbf{U}' \\
&= \lambda\mathbf{I} + \mathbf{A}\mathbf{B} \qquad (18)
\end{aligned}$$

where we define $\mathbf{A} = [\mathbf{U}, \theta\mathbf{G}\mathbf{U}]$ and $\mathbf{B} = [\Lambda\mathbf{U}'; \Lambda\mathbf{U}']$.

Finally, applying Matrix Inversion Lemma to Eq. (18), we have the coefficients $\boldsymbol{\beta}_{t+1}$

$$\begin{aligned}
\boldsymbol{\beta}_{t+1} &= \mathbf{S}_{t+1}^{-1} \begin{bmatrix} \mathbf{Y}_{t+1}^q \\ \mathbf{Y}_{t+1}^a \end{bmatrix} \\
&\approx (\lambda\mathbf{I} + \mathbf{A}\mathbf{B})^{-1} \begin{bmatrix} \mathbf{Y}_{t+1}^q \\ \mathbf{Y}_{t+1}^a \end{bmatrix} \\
&= \frac{1}{\lambda}(\mathbf{I} - \mathbf{A}(\lambda\mathbf{I} + \mathbf{B}\mathbf{A})^{-1}\mathbf{B}) \begin{bmatrix} \mathbf{Y}_{t+1}^q \\ \mathbf{Y}_{t+1}^a \end{bmatrix} \qquad (19)
\end{aligned}$$

The complete algorithm of LIP-KIMA is summarized in Alg. 2. As we can see, in addition to $\boldsymbol{\beta}_{t+1}$, the only variables we need to update are $\mathbf{U}_t^q$, Λ_t^q, $\mathbf{U}_t^a$ and Λ_t^a. Compared to Alg. 1, we do not need to store $\boldsymbol{\beta}_t$; instead we need to store $\mathbf{Y}_t^q$ and $\mathbf{Y}_t^a$. More importantly, we do not need to store $\mathbf{S}_t$; instead, we only need to store the much smaller matrices of $\mathbf{U}_t^q$, Λ_t^q, $\mathbf{U}_t^a$ and Λ_t^a.

Algorithm 2 The LIP-KIMA Algorithm.

Input: $\mathbf{U}_t^q$, Λ_t^q, $\mathbf{U}_t^a$, Λ_t^a, $\mathbf{F}_t^q$, $\mathbf{f}_{t+1}^q$, $\mathbf{F}_t^a$, $\mathbf{f}_{t+1}^a$, $\mathbf{Y}_t^q$, $\mathbf{y}_{t+1}^q$, $\mathbf{Y}_t^a$, $\mathbf{y}_{t+1}^a$, $\mathbf{M}_t$ and $\mathbf{m}_{t+1}$

Output: $\boldsymbol{\beta}_{t+1}$, $\mathbf{U}_{t+1}^q$, Λ_{t+1}^q, $\mathbf{U}_{t+1}^a$ and Λ_{t+1}^a

1: compute the new kernels $\mathbf{k}_{t+1}^q$ and $\mathbf{k}_{t+1}^a$ in Eq. (7);
2: update $\mathbf{U}_{t+1}^q$, Λ_{t+1}^q, $\mathbf{U}_{t+1}^a$ and Λ_{t+1}^a as Eq. (16);
3: define $\mathbf{U}$, Λ and $\mathbf{G}$ as Eq. (17);
4: define $\mathbf{A}$ and $\mathbf{B}$ as $[\mathbf{U}, \theta\mathbf{G}\mathbf{U}]$ and $[\Lambda\mathbf{U}'; \Lambda\mathbf{U}']$;
5: update $\boldsymbol{\beta}_{t+1}$ as $\frac{1}{\lambda}(\mathbf{I} - \mathbf{A}(\lambda\mathbf{I} + \mathbf{B}\mathbf{A})^{-1}\mathbf{B}) \begin{bmatrix} \mathbf{Y}_{t+1}^q \\ \mathbf{Y}_{t+1}^a \end{bmatrix}$;
6: **return** $\boldsymbol{\beta}_{t+1}$, $\mathbf{U}_{t+1}^q$, Λ_{t+1}^q, $\mathbf{U}_{t+1}^a$ and Λ_{t+1}^a;

Algorithm Analysis.The effectiveness of LIP-KIMA is summarized in Lemma 4. According to Lemma 4, there are two possible places where we could introduce the approximation error in the LIP-KIMA algorithm, including (a) eigen-decomposition for the $\mathcal{K}_t$ and (b) the Nyström method for $\mathcal{K}_{t+1}$. Notice that if we only do eigen-decomposition at $t = 1$, such approximation error might be accumulated and amplified over time. In practice, we could 're-start' the algorithm every few time ticks, that is, to re-compute (as opposed to approximate) the eigen-decomposition for the current kernel matrix.

LEMMA 4. **Effectiveness of LIP-KIMA**. *Let* $\boldsymbol{\beta}_{t+1}^*$ *be the output of Eq.* (5) *at time* $t+1$*, and* $\boldsymbol{\beta}_{t+1}$ *be the output of Alg. 2 updated from time* t *to* $t+1$*, we have* $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_{t+1}^*$ *if* $\mathcal{K}_t = \mathbf{U}_t\Lambda_t(\mathbf{U}_t)'$ *and* $\mathbf{h}_{t+1} = \mathbf{k}_{t+1}(\mathcal{K}_t)^{-1}(\mathbf{k}_{t+1})'$ *hold for both questions and answers.*

PROOF. Omitted for brevity. □

The time complexity and space complexity of Alg. 2 is summarized in the following lemma. It basically says that the LIP-KIMA algorithm requires *linear* time and space wrt the total number of questions and answers.

LEMMA 5. **Complexity of LIP-KIMA**. *The time complexity of Alg. 2 is* $O((n^q + n^a + i^q + i^a)r^2 + r^3 + (n^q i^q + n^a i^a)d)$*; the space complexity of Alg. 2 is* $O((n^q + n^a + i^q + i^a)d + (n^q + n^a + i^q + i^a)r + r^2)$.

PROOF. Omitted for brevity. □

Since i^q and i^a are usually much smaller than n^q and n^a, and the low rank r and feature dimension d are fixed constants, the time complexity and space complexity of Alg. 2 can be re-written as $O(n^q + n^a)$ in terms of the total number of questions and answers.

3.4 LIP-KIMAA Algorithm

Compared with LIP-KIM, LIP-KIMA is much more scalable, being *linear* in terms of both time and space complexity. However, if the new training examples arrive in a stream-like, ever-growing fashion, a linear algorithm might be still too expensive. To address this issue, we further present the LIP-KIMAA algorithm to reduce the complexity to be *sub-linear*.

Our LIP-KIMAA is built upon LIP-KIMA. The main difference between LIP-KIMAA and LIP-KIMA is that we add an additional filtering step between Step 1 and Step 2 in Alg. 2. That is, when new questions and answers arrive at time $t+1$, we first treat them as test set and apply the existing model at time t on this test set. Based on the prediction results, we only add the questions and answers whose prediction error is larger than a given threshold *th*. Notice that the complexity of LIP-KIMA is linear wrt the number of questions and answers; as a result, our LIP-KIMAA scales linearly wrt the number of *remaining* questions and answers after the filtering steps. Therefore, LIP-KIMAA scales sub-linearly wrt to the total number of questions and answers in both time and space. We omit the detailed algorithm for brevity.

4. VARIANTS

The proposed LIP-KIM and its two approximate algorithms (LIP-KIMA and LIP-LIMAA) are *comprehensive*. In terms of the modeling power, they capture all the three aspects (non-linearity, coupling, and dynamics). In this section, we show that our algorithms are also *flexible*. That is, if only a subset of these three aspects matter with the prediction performance for some applications, our algorithms can be naturally adapted to these special cases. We briefly discuss some of these variants, and then summarize the algorithms in Fig. 1.

4.1 Variant 1: Linear Model LIP-IM

If we only consider the coupling and dynamic aspects (i.e., ignoring the non-linear aspect), our LIP-KIM can be simplified as the LIP-IM algorithm. Essentially, LIP-IM aims to efficiently update the solution of Eq. (1)

$$\boldsymbol{\alpha} = \begin{bmatrix} (\theta+1)(\mathbf{F}^q)'\mathbf{F}^q + \lambda\mathbf{I} & -\theta(\mathbf{F}^q)'\mathbf{M}\mathbf{F}^a \\ -\theta(\mathbf{F}^a)'\mathbf{M}'\mathbf{F}^q & (\mathbf{F}^a)'\mathbf{F}^a + \theta(\mathbf{F}^a)'\mathbf{M}'\mathbf{M}\mathbf{F}^a + \lambda\mathbf{I} \end{bmatrix}^{-1} \begin{bmatrix} (\mathbf{F}^q)'\mathbf{Y}^q \\ (\mathbf{F}^a)'\mathbf{Y}^a \end{bmatrix}$$

where $\boldsymbol{\alpha} = [\boldsymbol{\alpha}^q; \boldsymbol{\alpha}^a]$.

In this case, the $\mathbf{S}_t$ matrix becomes

$$\mathbf{S}_t = \begin{bmatrix} (\theta+1)(\mathbf{F}_t^q)'\mathbf{F}_t^q + \lambda\mathbf{I} & -\theta(\mathbf{F}_t^q)'\mathbf{M}_t\mathbf{F}_t^a \\ -\theta(\mathbf{F}_t^a)'\mathbf{M}_t'\mathbf{F}_t^q & (\mathbf{F}_t^a)'\mathbf{F}_t^a + \theta(\mathbf{F}_t^a)'\mathbf{M}_t'\mathbf{M}_t\mathbf{F}_t^a + \lambda\mathbf{I} \end{bmatrix}$$

When new questions and answers arrive at time $t+1$, we define the $\mathbf{L}$ matrix and the $\mathbf{R}$ matrix as

$$\mathbf{L} = \begin{bmatrix} (\theta+1)(\mathbf{f}_{t+1}^q)' & -\theta(\mathbf{f}_{t+1}^q)' & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\theta(\mathbf{f}_{t+1}^a)'\mathbf{m}_{t+1}' & \left[(\mathbf{f}_{t+1}^a)', \theta(\mathbf{f}_{t+1}^a)'\mathbf{m}_{t+1}'\right] \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} \mathbf{f}_{t+1}^q & \mathbf{0} \\ \mathbf{0} & \mathbf{m}_{t+1}\mathbf{f}_{t+1}^a \\ \mathbf{f}_{t+1}^q & \mathbf{0} \\ \mathbf{0} & \left[\mathbf{f}_{t+1}^a; \mathbf{m}_{t+1}\mathbf{f}_{t+1}^a\right] \end{bmatrix}$$

As we can see, both $\mathbf{L}$ and $\mathbf{R}$ are only based on new observations $\mathbf{f}_{t+1}^q$, $\mathbf{f}_{t+1}^a$ and $\mathbf{m}_{t+1}$.

Next, we further define matrix $\mathbf{C} = \mathbf{S}_t^{-1}\mathbf{L}(\mathbf{I} + \mathbf{R}\mathbf{S}_t^{-1}\mathbf{L})^{-1}\mathbf{R}$, and we can show that the model can be updated as follows

$$\begin{aligned} \mathbf{S}_{t+1}^{-1} &= (\mathbf{I} - \mathbf{C})\mathbf{S}_t^{-1} \\ \boldsymbol{\alpha}_{t+1} &= (\mathbf{I} - \mathbf{C})\left(\boldsymbol{\alpha}_t + \mathbf{S}_t^{-1}\begin{bmatrix} (\mathbf{f}_{t+1}^q)'\mathbf{y}_{t+1}^q \\ (\mathbf{f}_{t+1}^a)'\mathbf{y}_{t+1}^a \end{bmatrix}\right) \end{aligned}$$

Remarks. Notice that compared with LIP-KIM, LIP-IM is much more efficient since it only needs to compute the inverse of a much smaller $(3i^q + i^a) \times (3i^q + i^a)$ matrix (see the computation of matrix $\mathbf{C}$). By ignoring the smaller terms (i.e., i^q and i^a), the overall time complexity of LIP-IM is $O(d^2)$ where d indicates feature dimension; on the other hand, directly updating the model would require $O((n^q + n^a)d + d^3)$ time. In the meanwhile, by a similar procedure as the proof for Theorem 1, we can show that LIP-IM finds the *exact* solution of Eq. (1).

4.2 Variant 2: Approximate LIP-KI

If we only consider the non-linearity and dynamics aspects (i.e., ignoring the coupling aspect), our LIP-KIMA can be further simplified. Take the prediction for questions as an example. With the approximate eigen-decomposition by Eq. (16): $\mathcal{K}_{t+1}^q \approx \mathbf{U}_{t+1}^q \Lambda_{t+1}^q (\mathbf{U}_{t+1}^q)'$, we can update the coefficients $\boldsymbol{\beta}_{t+1}^q$ as follows

$$\begin{aligned} \boldsymbol{\beta}_{t+1}^q &\approx (\mathbf{U}_{t+1}^q \Lambda_{t+1}^q (\mathbf{U}_{t+1}^q)' + \lambda\mathbf{I})^{-1}\mathbf{Y}_{t+1} \\ &= \mathbf{U}_{t+1}^q \Lambda_3^q (\mathbf{U}_{t+1}^q)'\mathbf{Y}_{t+1} \end{aligned}$$

where Λ_3^q is still a diagonal matrix with $\Lambda_3^q(i,i) = 1/(\Lambda_{t+1}^q(i,i) + \lambda)$.

Remarks. Although sharing the same linear time complexity as LIP-KIMA, this variant is even more efficient in practice since it does not need any matrix inversion at all.

4.3 Variant 3: LIP-KIM with Only Questions

Next, we discuss a special case of LIP-KIM when only new questions arrive at time $t+1$. Compared to Alg. 1, we can simplify the following notations

$$\begin{aligned} \mathbf{S}_1 &= \begin{bmatrix} (\theta+1)(\mathbf{k}_{t+1}^q)' \\ -\theta\mathbf{M}_t'(\mathbf{k}_{t+1}^q)' \end{bmatrix} \\ \mathbf{S}_2 &= \begin{bmatrix} (\theta+1)\mathbf{k}_{t+1}^q & \mathbf{0}_{i^q \times (n^q + n^a)} \end{bmatrix} \\ \mathbf{S}_3 &= (\theta+1)\mathbf{h}_{t+1}^q + \lambda\mathbf{I} \\ \mathbf{E}_1 &= \begin{bmatrix} \mathbf{I}_{n_q \times n_q} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{i_q \times i_q} \\ \mathbf{0} & \mathbf{I}_{n_a \times n_a} & \mathbf{0} \end{bmatrix} \end{aligned}$$

Then the model can be updated as

$$\begin{aligned} \mathbf{S}_{t+1}^{-1} &= \mathbf{E}_1\begin{bmatrix} \mathbf{S}_t^{-1} + \mathbf{S}_t^{-1}\mathbf{S}_1\mathbf{D}\mathbf{S}_2\mathbf{S}_t^{-1} & -\mathbf{S}_t^{-1}\mathbf{S}_1\mathbf{D} \\ -\mathbf{S}_3^{-1}\mathbf{S}_2\mathbf{S}_t^{-1}(\mathbf{I} + \mathbf{S}_1\mathbf{D}\mathbf{S}_2\mathbf{S}_t^{-1}) & \mathbf{D} \end{bmatrix}\mathbf{E}_1' \\ \boldsymbol{\beta}_{t+1} &= \mathbf{E}_1\begin{bmatrix} \boldsymbol{\beta}_t + \mathbf{S}_t^{-1}\mathbf{S}_1\mathbf{D}(\mathbf{S}_2\boldsymbol{\beta}_t - \mathbf{y}_{t+1}^q) \\ -\mathbf{S}_3^{-1}\mathbf{S}_2(\boldsymbol{\beta}_t + \mathbf{S}_t^{-1}\mathbf{S}_1\mathbf{D}\mathbf{S}_2\boldsymbol{\beta}_t) + \mathbf{D}\mathbf{y}_{t+1}^q \end{bmatrix} \end{aligned}$$

Remarks. The time complexity of this variant is $O((n^q + n^a)^2 i^q + (n^q + n^a)(i^q)^2 + (i^q)^3 + (n^q i^q + (i^q)^2)d)$, which can be re-written as $O((n^q + n^a)^2)$ by ignoring smaller terms of i^q and d. The correctness of this variant can also be shown by following a similar procedure as the proof for Theorem 1.

Table 2: The summarization of the algorithms from the genealogy graph in Fig. 1. The complexity is the time complexity for updating the model at each time tick, where the n is the totally number of the training examples and those smaller terms are omitted for clarity. The right five columns are the proposed algorithms in this paper.

	SVR	KRR (LIP-K)	RKRR (LIP-KI)	CoPs (LIP-M)	LIP-IM	LIP-KM	LIP-KIM	LIP-KIMA	LIP-KIMAA
Non-linearity	✓	✓	✓	x	x	✓	✓	✓	✓
Coupling	x	x	x	✓	✓	✓	✓	✓	✓
Dynamics	x	x	✓	x	✓	x	✓	✓	✓
Complexity	$\geqslant O(n^2)$	$O(n^3)$	$O(n^2)$	$O(n)$	$O(1)$	$O(n^3)$	$O(n^2)$	$O(n)$	$< O(n)$

Table 3: The statistics of SO and Math data sets.

Data	Questions	Answers	Users	Votes
SO	1,966,272	4,282,570	756,695	14,056,000
Math	16,638	32,876	12,526	202,932

4.4 Summarization

Finally, we make a comparison of different algorithms in Fig. 1 in terms of their modeling power and efficiency. The results are summarized in Table 2. In this table, we first check whether a given algorithm captures each of the three desired aspects (non-linearity, coupling, and dynamics). As we can see, only our LIP-KIM, LIP-KIMA, and LIP-KIMAA algorithms meet this requirement. We also summarize the time complexity of the algorithm for updating the model at each time tick, where the smaller terms (e.g., the feature dimensionality d, the number of new training examples i^q, i^a, etc) are omitted for clarity. For non-linear algorithms like KRR [25], support vector regression (SVR) [9, 7], RKRR [11], LIP-KM and LIP-KIM, they need at least quadratic time for the model training because they typically need to maintain the kernel matrix. Such a complexity is unaffordable in many CQA sites. On the other hand, the time complexity of the proposed LIP-KIMA and LIP-KIMAA algorithms is *linear* and *sub-linear* wrt the total number of questions and answers, respectively.

5. EXPERIMENTS

In this section, we present the experimental evaluations. The experiments are designed to answer the following questions:

- *Effectiveness*: How accurate are the proposed algorithms for predicting the long-term impact of questions/answers?
- *Efficiency*: How scalable are the proposed algorithms?

5.1 Experiment Setup

We use the data from two real CQA sites, i.e., Stack Overflow (SO) and Mathematics Stack Exchange (Math), to evaluate our algorithms. They are popular CQA sites for programming and math, respectively. The statistics of the two data sets are summarized in Table 3.

We use both content and contextual features. For content features, we adopt the "bag of words" model to extract content features after removing the infrequent words. This model is widely used in natural language processing where the frequency of each word is used as a feature for training. In addition, we adopt some commonly used contextual features in the literature, including the questioner's reputation at question creation time, the answerer's reputation at answer creation time, the length of the question/answer, the number of questioner's previous questions at question creation time, and the number of answerer's previous answers at answer creation time. For these features, we can extract them at the moment when the question/answer is posted. For other contextual features, we need to choose a short time window by the end of which the voting score is predicted. With such a fixed time window, we can include some time-related features such as the number of comments/answers received during the fixed time window. In this work, we fix this time window as 24 hours. Overall, we have 4,180 and 4,444 features for Math data and SO data, respectively (i.e., $d = 4,180$ for Math data and $d = 4,444$ for SO data). For the kernel matrix, we adopt the cosine kernel function due to the sparsity of our feature matrix.

For long-term impact, we restrict our attention to predict the impact of a question/answer after it is posted for six months. For each post in the data set, there are several choices to measure impact including the number of pageviews, the number of favorites, and the user voting score (which is the difference between the number of up-votes and the number of down-votes on the post). In this work, we choose the voting score for the following two reasons. First, we conduct a survey, asking different users about which is the best metric as the long-term impact measure of CQA posts; and most of the users (79.4%) choose the voting score. Second, to some extent, the voting score of a question/answer resembles the number of the citations that a research paper receives in the scientific publication domain. It reflects the net number of users who have a positive attitude toward it. In addition, we also measured the correlation between the three choices and found that all of them are strongly positive correlated. Thus, we expect that our algorithms could also be used to predict the other two metrics (i.e., pageviews and favorites). We normalize the voting scores into the range of $[0, 1]$.

To evaluate the dynamic aspect of our algorithms, we start with a small initial training set, and gradually add new examples into the training set in chronological order. For Math data, we start with 5% initial data, add 5% data for each update, and use the latest 10% data as the test set. For SO data whose size is much larger than the Math data, we start with 0.1% initial data, add 0.1% data for each update, and use the latest 0.1% data as the test set.

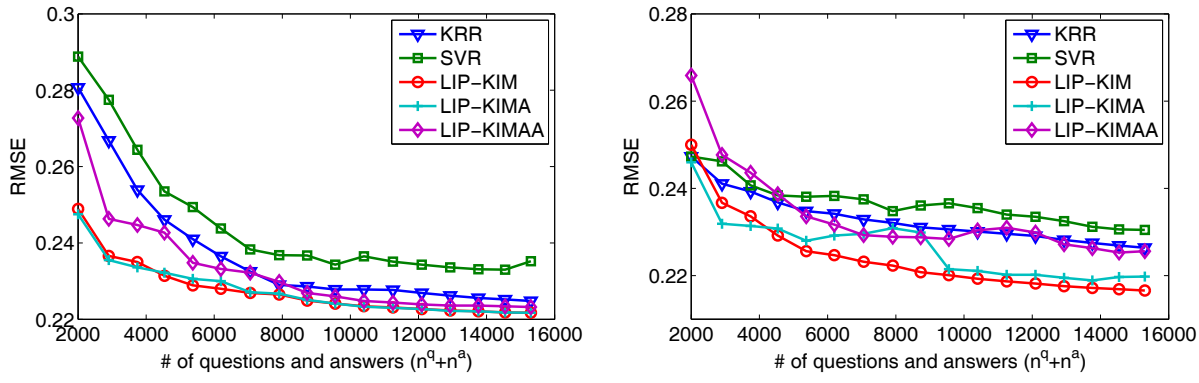
For evaluation metrics, we adopt the root mean square error (RMSE) between the real impact and the estimated impact for effectiveness, and the wall-clock time for efficiency. All the efficiency experiments were run on a machine with eight 3.4GHz Intel Cores and 24GB memory.

Repeatability of Experimental Results. Both data sets are officially published and publicly available[5]. Furthermore, we will make the code of the proposed algorithms as well as the extracted feature files publicly available. For all the results reported in this section, the specific parameter settings are as follows. We set $\lambda = \theta = 1$ for Math data and $\lambda = \theta = 0.1$ for SO data. For the low rank r in LIP-KIMA and LIP-KIMAA, we set it as 10. For LIP-KIMAA, we set $th = 0.22$ for Math data and $th = 0.18$ for SO data.

5.2 Effectiveness Results

We first compare the effectiveness of the proposed algorithms with two state-of-the-art non-linear regression methods, i.e., kernel

[5]http://blog.stackoverflow.com/category/cc-wiki-dump/



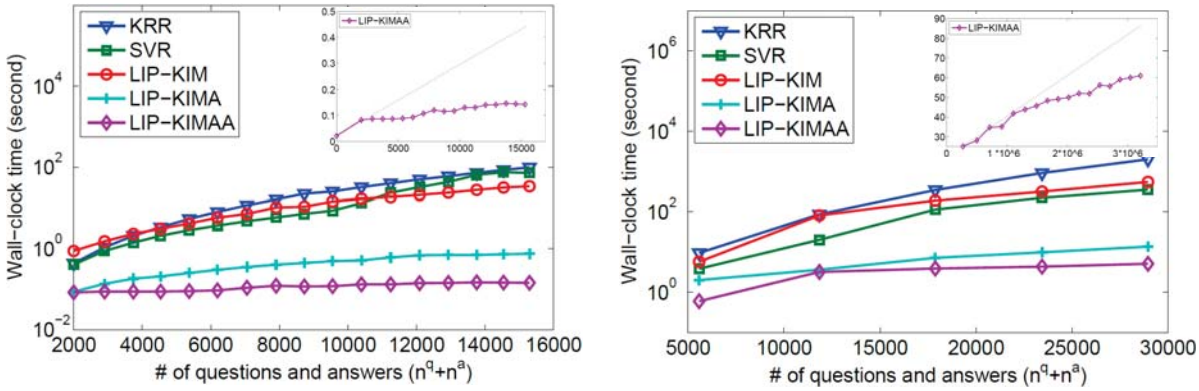
(a) Question impact prediction on Math data

(b) Answer impact prediction on Math data

(c) Question impact prediction on SO data

(d) Answer impact prediction on SO data

Figure 2: The effectiveness comparisons. Lower is better. The proposed algorithms outperform both SVR and KRR.



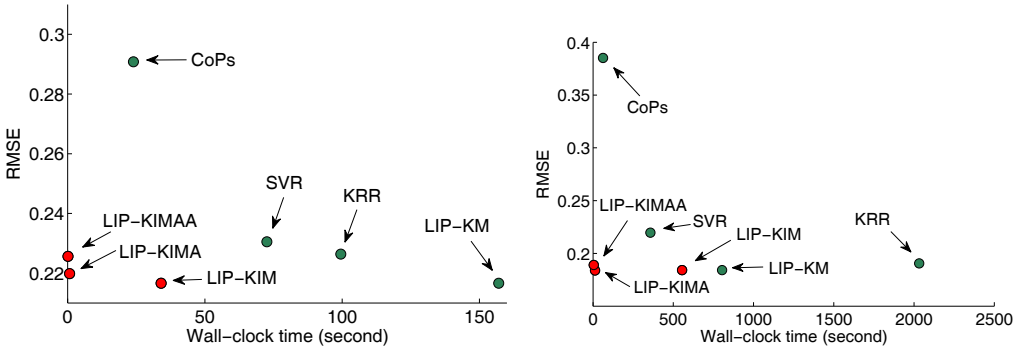
(a) Math data

(b) SO data

Figure 3: The speed comparisons. The proposed LIP-KIMA and LIP-KIMAA are much faster. Furthermore, LIP-KIMAA scales sub-linearly (in the upper-right corner).

ridge regression (KRR) [25] and support vector regression (SVR) [7]. The prediction results of questions and answers on the two data sets are shown in Fig. 2. On SO data, we only report the first few points because some of the algorithms (e.g., KRR) cannot finish training within 1 hour. We do not report the results by linear models (e.g., linear ridge regression) since their performance (RMSE) is much worse than SVR.

We make several observations from Fig. 2. First, the proposed LIP-KIM algorithm performs the best in most of the cases. For example, when the size of training set increases to 90% on the Math data, LIP-KIM improves the SVR method by 5.7% for questions and 6.0% for answers. On SO data, LIP-KIM improves the KRR method by up to 35.8% for questions and 3.6% for answers. This indicates that the coupling aspect indeed helps in impact prediction. Second, the performance of the proposed LIP-KIMA algorithm is close to LIP-KIM. This result indicates that while it reduces the time complexity from quadratic to linear, the approximation method introduces little performance loss. Finally, although not as good as LIP-KIM and LIP-KIMA, the LIP-KIMAA algo-



(a) Answer impact prediction on Math data (b) Answer impact prediction on SO data

Figure 4: The quality-speed balance-off. The proposed LIP-KIMA and LIP-KIMAA achieve a good balance between the prediction quality and the efficiency (in the left-bottom corner). Best viewed in color.

Table 4: Performance gain analysis. Smaller is better. All three aspects of non-linearity, coupling, and dynamics are helpful.

Questions/Answers	SO	Math
Ridge regression	0.4920/0.4409	0.2799/0.3860
LIP-K	0.4214/0.2044	0.2461/0.2368
LIP-KM	0.2704/0.1987	0.2314/0.2292
LIP-KIM	0.2595/0.1867	0.2249/0.2208

rithm is still better than the compared methods for most of the cases.

To further show the effects of all the three aspects (i.e., non-linearity, coupling, and dynamics), we analyze the performance gain in Table 4. In the table, LIP-K incorporates non-linearity into ridge regression, LIP-KM incorporates coupling into LIP-K, and LIP-KIM incorporates dynamics into LIP-KM. As we can see, all three aspects are helpful to improve the prediction performance.

5.3 Efficiency Results

Next, we compare the efficiency of the proposed algorithms with KRR and SVR in Fig. 3. Notice that the y-axis is in log scale. In Fig. 3, we also plot the results of LIP-KIMAA with y-axis in linear scale in the upper-right corner. We only report the results by LIP-KIMAA there because it is the only algorithm that can handle the entire SO data set.

As we can see from the figure, our LIP-KIMA and LIP-KIMAA are much faster than the other algorithms. In the upper-right corner, we can observe that the LIP-KIMAA scales *sub-linearly* wrt the total number of questions and answers. For instance, it only requires about 60 seconds when there are more than 3,000,000 questions and answers. In contrast, KRR requires more than 2,000 seconds when the size of the training set is about 30,000.

Finally, we study the quality-speed balance-off of different algorithms in Fig. 4. In the figure, we show the answer prediction results only. Similar results are observed in question prediction, and we omit the results for brevity. In Fig. 4, we plot the RMSE on the y-axis and the wall-clock time on the x-axis. We also plot the results of the linear co-prediction method CoPs [29] and LIP-KM. Ideally, we want an algorithm sitting in the left-bottom corner. As we can see, both our LIP-KIMA and LIP-KIMAA are in the left-bottom corner. For example, for answer impact prediction on the SO data, compared with SVR, LIP-KIMAA is 70x faster in wall-clock time and 14.0% better in RMSE. Overall, we recommend LIP-KIMAA in practice.

6. RELATED WORK

In this section, we briefly review related work including mining CQA sites and mining stream data.

Mining CQA Sites: There is a large body of existing work on mining CQA sites. For example, Li et al. [19] aim to predict question quality, which is defined as the combination of user attention, answer attempts and the arrival speed of the best answer. Jeon et al. [18] and Suryanto et al. [28] evaluate the usefulness of answer quality and incorporate it to improve retrieval performance. To predict the quality of both questions and answers, Agichtein et al. [2] develop a graph-based model to catch the relationships among users, Li et al. [20] adopt the co-training approach to employ both question features and answer features, and Bian et al. [5] propose to propagate the labels through user-question-answer graph so as to tackle the sparsity problem where only a small number of questions/answers are labeled. Recently, Anderson et al. [3] propose to predict the long-lasting value (i.e., the pageviews) of a question and its answers. How to predict the answer that the questioner will probably choose as the accepted answer is also well studied [22, 26, 1]. Overall, our work differs from these existing work at the methodology level. While most of the existing work treats the prediction problem as a single, and/or linear, and/or static problem, we view the problem from a comprehensive perspective and propose to incorporate all these important aspects into the prediction models.

Mining Stream Data: From the dynamic aspect, our LIP problem is related to stream mining [13] and time-series mining [12]. The main focus of existing stream/time-series mining work is on pattern discovery, clustering, and classification tasks. Chen et al. [8] and Ikonomovska et al. [17] study the regression problem in data streams; however, they still focus on a single and linear prediction problem. Several researchers also consider the non-linear and dynamic aspects in regression problem [11, 23]. Different from these existing work, we consider the coupling between questions and answers, and propose approximation methods to speed-up and scale-up the computation.

Other Related Work: There are several pieces of interesting work that are remotely related to our work. Liu et al. [21] propose the problem of CQA site searcher satisfaction, i.e., whether or not the answer in a CQA site satisfies the information searcher using the search engines. Shtok et al. [27] attempt to answer certain new questions by existing answers. The question routing problem (e.g., how to route the right question to the right answerer) is also an active research area [30, 16].

7. CONCLUSIONS

In this paper, we have proposed a family of algorithms to predict the long-term impact of questions/answers in CQA sites. The proposed algorithms enjoy three key advantages. First, they are *comprehensive* in the sense that our model naturally captures three key aspects (i.e., non-linearity, coupling, and dynamics) that matter with the long-term impact of a post. Second, they are *flexible* and *general*, being able to handle the special cases where only a fraction of these aspects are prominent. Third, they are *scalable* and *adaptive* to the newly arrived questions and answers. We analyze our algorithms in terms of *optimality*, *correctness*, and *complexity*, and reveal the intrinsic relationship among different algorithms. We conduct extensive experimental evaluations on two real CQA data sets to demonstrate the effectiveness and efficiency of our approaches.

8. ACKNOWLEDGMENTS

This work is supported by the National 863 Program of China (No. 2012AA011205), and the National Natural Science Foundation of China (No. 91318301, 61321491, 61100037). This material is partially supported by by the National Science Foundation under Grant No. IIS1017415, by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053, by Defense Advanced Research Projects Agency (DARPA) under Contract Number W911NF-11-C-0200 and W911NF-12-C-0028, and by Region II University Transportation Center under the project number 49997-33 25.

The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

9. REFERENCES

[1] L. Adamic, J. Zhang, E. Bakshy, and M. Ackerman. Knowledge sharing and yahoo answers: everyone knows something. In *WWW*, pages 665–674. ACM, 2008.

[2] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *WSDM*, pages 183–194. ACM, 2008.

[3] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec. Discovering value from community activity on focused question answering sites: a case study of stack overflow. In *KDD*, pages 850–858. ACM, 2012.

[4] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

[5] J. Bian, Y. Liu, D. Zhou, E. Agichtein, and H. Zha. Learning to recognize reliable users and content in social media with coupled mutual reinforcement. In *WWW*, pages 51–60. ACM, 2009.

[6] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[7] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

[8] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang. Multi-dimensional regression analysis of time-series data streams. In *VLDB*, pages 323–334, 2002.

[9] R. Collobert and S. Bengio. Svmtorch: Support vector machines for large-scale regression problems. *The Journal of Machine Learning Research*, 1:143–160, 2001.

[10] P. Drineas and M. W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *The Journal of Machine Learning Research*, 6:2153–2175, 2005.

[11] Y. Engel, S. Mannor, and R. Meir. The kernel recursive least-squares algorithm. *IEEE Transactions on Signal Processing*, 52(8):2275–2285, 2004.

[12] T.-c. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[13] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams: a review. *ACM Sigmod Record*, 34(2):18–26, 2005.

[14] G. Golub and C. Van Loan. Matrix computations. 1996.

[15] S. S. Haykin. *Adaptive filter theory*. 2005.

[16] D. Horowitz and S. Kamvar. The anatomy of a large-scale social search engine. In *WWW*, pages 431–440. ACM, 2010.

[17] E. Ikonomovska, J. Gama, and S. Džeroski. Learning model trees from evolving data streams. *Data mining and knowledge discovery*, 23(1):128–168, 2011.

[18] J. Jeon, W. Croft, J. Lee, and S. Park. A framework to predict the quality of answers with non-textual features. In *SIGIR*, pages 228–235. ACM, 2006.

[19] B. Li, T. Jin, M. R. Lyu, I. King, and B. Mak. Analyzing and predicting question quality in community question answering services. In *WWW*, pages 775–782. ACM, 2012.

[20] B. Li, Y. Liu, and E. Agichtein. Cocqa: co-training over questions and answers with an application to predicting question subjectivity orientation. In *EMNLP*, pages 937–946, 2008.

[21] Q. Liu, E. Agichtein, G. Dror, E. Gabrilovich, Y. Maarek, D. Pelleg, and I. Szpektor. Predicting web searcher satisfaction with existing community-based answers. In *SIGIR*, pages 415–424, 2011.

[22] Y. Liu, J. Bian, and E. Agichtein. Predicting information seeker satisfaction in community question answering. In *SIGIR*, pages 483–490. ACM, 2008.

[23] B. Pan, J. J. Xia, P. Yuan, J. Gateno, H. H. Ip, Q. He, P. K. Lee, B. Chow, and X. Zhou. Incremental kernel ridge regression for the prediction of soft tissue deformations. In *Medical Image Computing and Computer-Assisted Intervention*, pages 99–106. Springer, 2012.

[24] W. W. Piegorsch and G. Casella. Inverting a sum of matrices. *SIAM Review*, 32(3):470–470, 1990.

[25] C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *ICML*, pages 515–521, 1998.

[26] C. Shah and J. Pomerantz. Evaluating and predicting answer quality in community qa. In *SIGIR*, pages 411–418, 2010.

[27] A. Shtok, G. Dror, Y. Maarek, and I. Szpektor. Learning from the past: answering new questions with past answers. In *WWW*, pages 759–768, 2012.

[28] M. Suryanto, E. Lim, A. Sun, and R. Chiang. Quality-aware collaborative question answering: methods and evaluation. In *WSDM*, pages 142–151. ACM, 2009.

[29] Y. Yao, H. Tong, T. Xie, L. Akoglu, F. Xu, and J. Lu. Want a good answer? ask a good question first! *arXiv preprint arXiv:1311.6876*, 2013.

[30] Y. Zhou, G. Cong, B. Cui, C. Jensen, and J. Yao. Routing questions to the right users in online communities. In *ICDE*, pages 700–711. IEEE, 2009.