PROVIDING PREDICTIONS ON HIDDEN MARKOV MODELS WITH PRIVACY

Şahin RENÇKEŞ
Master of Science Thesis

Computer Engineering Program
June, 2008

This thesis was supported by Grant 107E209 from TUBITAK.

JÜRİ VE ENSTİTÜ ONAYI

Şahin Rençkeş'in "**Providing Predictions On Hidden Markov Models With Privacy**" başlıklı **Bilgisayar Mühendisliği** Anabilim Dalındaki, Yüksek Lisans Tezi 10.06.2008 tarihinde, aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

	Adı-Soyadı	**İmza**
Üye (Tez Danışmanı)	**: Yard. Doç. Dr. HÜSEYİN POLAT**	**…..…….**
Üye	**: Doç. Dr. YUSUF OYSAL**	**…..…….**
Üye	**: Yard. Doç. Dr. AHMET YAZICI**	**..……**

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun …… tarih ve …… sayılı kararıyla onaylanmıştır.

Enstitü Müdürü

ABSTRACT

Master of Science Thesis

PROVIDING PREDICTIONS ON HIDDEN MARKOV MODELS WITH PRIVACY

Şahin RENÇKEŞ

Anadolu University
Graduate School of Sciences
Computer Engineering Program

Supervisor: Assist. Prof. Dr. Hüseyin POLAT
2008, 71 pages

Hidden Markov models (HMMs) are widely used in various applications to make predictions. HMM owners employ their models to compute the probability of occurrence of an observation sequence and how to choose a state sequence so that the joint probability of the observation and the state sequences given the model is maximized. In some appliactions, the model constructed for prediction purposes might be horizontally or vertically split between two parties. To be able to provide predictions, such parties might decide to integrate their split models. However, due to privacy and financial reasons, they do not want to combine their models. If privacy measures are introduced, model owners can integrate their models. HMMs can also be used for collaborative filtering (CF) purposes. The idea of Markov models can be utilized to produce recommendations to customers without jeopardizing their privacy.

In this thesis, solutions are presented to compute the probability of occurrence of an observation sequence based on split models between two parties without jeopardizing model owners' privacy. Moroever, approaches are proposed to choose a state sequence so that the joint probability of the observation and the state sequences given the split models is maximized with privacy. And finally, schemes are proposed to provide CF services with privacy using the idea of Markov model. The proposed schemes are analyzed in terms of accuracy, privacy, and efficiency. Experiments are performed on real data sets and their outcomes are displayed.

Keywords: Privacy, hidden Markov models, finance, model-based forecasting.

ÖZET

Yüksek Lisans Tezi

GİZLİLİĞİ KORUYARAK SAKLI MARKOV MODELLERİNE DAYALI TAHMİNLER ÜRETME

Şahin RENÇKEŞ

Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Yard. Doç. Dr. Hüseyin POLAT
2008, 71 sayfa

Saklı Markov modelleri (SMM) tahmin üretmek için bir çok alanda yaygın olarak kullanılır. SMM sahipleri modellerini bir gözlem serisinin görülme olasılığını hesaplama ve gözlem ve durum serilerinin birleşik olasılığının maksimum olacağı durum serisinin seçilmesi için kullanır. Bazı uygulamalarda tahmin için oluşturulan model yatay veya dikey olarak iki kişi arasında bölünmüş olabilir. Tahmin üretebilmek için bu kişiler modellerini birleştirmeye karar verebilir. Fakat gizlilik ve maddi nedenlerden dolayı bu kişiler modellerini birleştirmek istemezler. Eğer gizlilik ölçütleri kullanılırsa, model sahipleri modellerini birleştirebilirler. SMMler işbirlikçi filtreleme (İF) için de kullanılabilir. Markov model düşüncesi müşterilerin gizliliklerini tehlikeye atmadan müşterilere tahmin üretmek için kullanılabilir.

Bu tezde, bir gözlem serisinin görülme olasılığını iki firma arasında bölünmüş olan modele dayalı olarak model sahiplerinin gizliliğini tehlikeye atmadan hesaplayacak çözümler sunulmuştur. Ayrıca parçalanmış modele dayalı olarak gözlem ve durum serilerinin birleşik olasılığının maksimum olacağı durum serisinin gizlilikle seçilmesi için çözümler önerilmiştir. En son olarak Markov model düşüncesi kullanılarak İF işlerinin gizlilikle yapılması için yöntemler sunulmuştur. Önerilen yöntemler doğruluk, gizlilik ve performans açısından incelenmiştir. Gerçek verilere dayalı deneyler yapılmış ve sonuçları gösterilmiştir.

Anahtar Kelimeler: Gizlilik, saklı Markov modelleri, finans, modele dayalı tahmin.

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Assist. Prof. Dr. Hüseyin POLAT for his guidance, encouragement, and especially for his patience. His support and endurance contributes greatly to the quality of this thesis. It is great pleasure to work with him. Also, I would like to thank my fellow workers İbrahim YAKUT and Cihan KALELİ for their scientific and technical support. Finally, I would like to thank my family for their moral contributions during my studies.

Şahin RENÇKEŞ

June,2008

CONTENTS

LIST OF TABLES

ABBREVATIONS

a : Active User

k : Number of Clusters

CA : Classification Accuracy

CF **:** Collaborative Filtering

F1 : *F*-Measure

HPD : Horizontally Partitioned Data

M : Number of Groups

n : Number of Train Users

NBC : Naïve Bayesian Classifier

PPCF : Privacy-Preserving Collaborative Filtering

PPDM : Privacy-Preserving Data Mining

TN : Top-*N* Recommendation

q : Target Item

RRT : Randomized Response Techniques

PL : Privacy Level

VPD : Vertically Partitioned Data

1. INTRODUCTION

A hidden Markov model (HMM) is a statistical model, which is used for modeling the systems that are assumed to be a Markov process, which has unknown parameters. The challenge is here to determine the hidden parameters from the observable parameters. In a normal Markov model, the states are directly visible to the observers, where the state transition probabilities are the parameters. In an HMM, the state is not directly visible, but variables influenced by the states sare visible. Each state has a probability distribution, which gives us information about the hidden parameters by observing sequences. HMMs are widely used in science, engineering, and many other areas like cryptanalysis, speech recognition, sign language recognition, gesture and body motion recognition, optical character recognition, machine translation, which investigates the use of computer software to translate text or speech from one natural language to another, robot navigation, bioinformatics, finance, economics, and data mining.

To denote an HMM constructed for forecasting, $\lambda = (A, B, \pi)$ is used as a compact notation [15]. The following notation is used to define the model:

N: number of states in the model,

M: number of distinct observation symbols,

T: length of observation sequence,

i_t denotes the state at time t,

$A = \{a_{ij}\}$, where $a_{ij} = P\ (i_{t+1} = j \mid i_t = i)$, the probability of being in state j at time $t + 1$ given that previously in state i at time t,

$B = \{b_j\ (k)\}$, $b_j\ (k) = P\ (v_k \text{ at } t \mid i_t = j)$, the probability of observing the symbol v_k given that recently in state j,

$\pi = \{\pi_i\}$, $\pi_i = P\ (i_1 = 1)$, the probability of being in state i at $t = 1$,

O_t denotes the observation symbol observed at instant t,

1, 2, . . ., N denotes the N states, respectively.

HMMs are effective for discrete-valued time series. HMMs are widely used in many parts of communication and speech recognition. They are also important in bioinformatics. In data mining, they are used for database mining, sequence classification, and pattern discovery. In web, HMMs provide a theoretical framework for analyzing the behavior of users but are potentially

useful for predicting future Web resource consumption. Such information may help develop strategies to increase the sales of products offered by the Web site or improve the navigational convenience of users. Moreover, in the research of economics time series, especially the macroeconomic and financial series, the conventional framework with a fixed density function or a single set of parameters may not be suitable and it is necessary to include the possible structural change in the analysis. Therefore, in finance and economics, HMMs are also used and known as regime switching models, which have a large literature.

Many economic time series occasionally exhibit dramatic breaks in their behavior, associated with events such as financial crises or abrupt changes in government policy. Of particular interest to economists is the apparent tendency of many economic variables to behave quite differently during economic downturns, when underutilization of factors of production rather than their long-run tendency to grow governs economic dynamics. Abrupt changes are also a prevalent feature of financial data, and the approach described below is quite amenable to theoretical calculations for how such abrupt changes in fundamentals should show up in asset prices [22].

In economy, Hassan and Nath [23] present HMMs approach for forecasting stock price for interrelated markets. HMM is also used to model regime change in economic time series, especially the macroeconomic and financial series [68]. Jagannathan et al. [27] present a simple deterministic algorithm, Recluster, for I/O-efficient *k*-clustering.

In cryptanalysis, Green et al. [21] extend the model of Karlof and Wagner for modelling side channel attacks via input driven hidden Markov models (IDHMMs) to the case, where not every state corresponds to a single observable symbol. Karlof and Wagner [33] present HMM attacks, a new type of cryptanalysis on modeling randomized side channel countermeasures as HMMs.

In machine translation field, an approach to modeling long-term consistencies in a speech signal within the framework of a hybrid hidden Markov model (HHMM) / multilayer perception (MLP) speaker-independent continuous-speech recognition system is presented by Abrash et al. [1]. Abrash et al. [2] develop a hybrid speech recognition system, which uses an MLP to estimate the

observation likelihoods associated with the states of an HMM. Franco et al. [17] compare two methods for modeling context in the framework of HHMM / MLP speaker-independent continuous speech recognition system. Jiang et al. [28] propose a novel method to estimate continuous-density hidden Markov model (CDHMM) for speech recognition according to the principle of maximizing the minimum multiclass separation margin. Vlasenko and Wendemuth [64] introduce a speech emotion recognition method based on HMMs. Yau et al. [66] present a novel visual speech recognition approach on motion segmentation and HMMs.

Shatkay [56] presents a formal framework for incorporating readily available odometric information and geometrical constraints into both the models and the algorithm. Savage et al. [54] describe a localization system for a mobile robot equipped with sonars. Fox et al. [16] describe a machine learning approach for acquiring a model of a robot behaviour from raw sensor data.

Birney [9] review machine learning techniques on the use of HMMs for investigating biomolecular sequences. An enhanced bioinformatics tool incorporating the participation of molecular structure as well as sequence in protein DNA recognition is proposed and tested via HMM by Thayer and Beveridge [58]. Husmeier and McGuire [26] present a statistical method for detecting recombination in DNA sequence alignments. Beausang et al. [6] present a new technique for measuring rate constants of DNA loop formation and breakdown mediated by repressor protein that binds to the DNA using a modified HMM analysis that directly incorporates the diffusive motion of the bead.

In data mining, Lin et al. [36] describe new temporal data mining techniques for extracting information from temporal health records consisting of time series of diabetic patients' treatments. Skounaki et al. [57] propose and evaluate an approach that is based on using hierarchical HMMs to represent the grammatical structure of the sentences being processed. Laxman et al. [34] establish a formal connection between two common, but previously unconnected methods for analyzing data streams: discovering frequent episodes in a computer science framework and learning generative models in a statistics framework. The potentials of HMMs in mining free-structured information are investigated in the study by Tso and Chang [59].

Today's business world is too competitive. Many business companies employ different techniques to obtain competitive edge over their competitors. To make strategic plans in order to be one step ahead of the others, companies need to know future trends of various products, whether new products will be liked or not, or how much they will be liked by customers, how much benefit they gain by investigating certain amounts of money into new business field, and so on. Corporations can learn such information using different models generated from historical data. HMMs are among such models and have many important applications in practice to make predictions as presented previously. They are widely used models in finance for forecasting.

In some cases, the model constructed for forecasting purposes might be split between various companies. This partition can be horizontal or vertical. The model owners want to integrate their split models; however, due to privacy and financial reasons, they do not want to reveal their private models to each other. To be able to provide predictions, they should integrate their models. If privacy measures are introduced, they might decide to combine their models. Furthermore, HMMs can be used for collaborative filtering (CF) to generate recommendations to customers. The idea of Markov models can be applied to CF for producing better referrals more efficiently. It is a challenge to provide CF services on Markov models idea without violating customers' privacy.

In this thesis, the following issues are investigated. First, solutions are sought to find predictions or the probability of occurrence of an observation sequence based on distributed HMMs between two parties while preserving their privacy. Second, approaches are proposed how to choose a state sequence so that the joint probability of the observation sequence and the state sequence given the distributed models between two parties is maximized without jeopardizing the model owners' privacy. Third, how to offer CF services using the idea of Markov models is investigated while preserving users' privacy. Proposed solutions are analyzed in terms of privacy, accuracy, and additional costs introduced due to underlying privacy protecting measures. Experiments are performed using real data sets and their outcomes are explained. Finally, the conclusions and future works are presented.

2. PRIVACY-PRESERVING PREDICTION ON DISTRIBUTED HMMs

2.1 Introduction

With increasing popularity of forecasting, model-based predictions are receiving increasing attention. Regression analysis, Bayesian networks, neural networks, HMMs, and so on are widely used to provide predictions.

Each forecasting method or model has its own advantages and disadvantages. Compared to other models, HMMs are very powerful methods. They can be combined into larger models. Moreover, they are transparent and employ prior knowledge. Although they have disadvantages like assumption that states are independent and low speed, their advantages surpass the drawbacks.

HMMs are widely used in many applications. As explained previously, in finance, speech recognition, bioinformatics, genomics, and so on, they are employed for forecasting purposes. They are very popular especially in finance. Financial trends, increases, and decreases can be predicted and recognized via HMMs. In addition, with HMMs, financial time series are predicted [67]. Moreover, they are used to model and forecast electricity prices [20]. To predict stock market trends, HMMs are employed, where they forecast stock price for interrelated markets [23].

An HMM is a statistical model, which is used to perform prediction. It is constructed based on historical data. It is then employed by various companies or users for forecasting purposes. With increasing available new data, HMMs are updated periodically.

After constructing an HMM, it is used to compute the probability of occurrence of an observation sequence. The model owners can offer predictions in return of some fee or benefits. In addition, they might decide to sell their models to others. When the model is owned by a single company, anyone might send an observation sequence to the model owner who can calculate the probability of occurrence of such sequence using the model. Although it is trivially easy to compute such probability when the model is held by a party, it becomes a challenge if the model is distributed between various parties, even competing companies. The model might be horizontally or vertically split between two or

more parties. To generate predictions based on the distributed model, the parties should combine the models they own. Although they want to share their models, they might not want to disclose them to each other due to privacy, financial, and legal reasons.

In this chapter, how to provide predictions or calculate the probability of occurrence of an observation sequence on horizontally or vertically distributed HMMs between two parties without violating their privacy is investigated. Privacy-preserving schemes to achieve such goals are proposed. The schemes are analyzed in terms of privacy, accuracy, and performance. The methods should allow the model owners to integrate their split models to offer accurate predictions efficiently while preserving their privacy.

2.2 Related Work

HMMs are extensively used in prediction, speech recognition, finance, and so on. Begletier et al. [7] propose to use variable order Markov models for prediction of discrete sequences. Rabiner [51] shows how HMMs can be applied to selected problems in speech recognition. The theories of HMMs from various concepts are presented. Henderson et al. [24] describe a new HMM to study how to segment human DNA into three regions. Hassan and Nath [23] study how to employ HMMs to predict stock market trends. Using HMMs, they forecast stock price for interrelated markets.

Privacy and distributed data-based computations are receiving increasing attention lately. With the evolution of the Internet, privacy becomes important. Companies and users do not want to divulge their private information to others. To perform richer data mining, provide better predictions, and offer more dependable outcomes, distributed data-based computations become popular.

Cranor et al. [13] conduct a survey about what users think about divulging private data. Great majority of people do not want to reveal their private data. Cranor [14] studies what kind of privacy risks that e-commerce sites pose while they collect data to offer predictions to their customers. Verykios et al. [63] present an overview of privacy-preserving issues in data mining. They propose a classification hierarchy that sets the basis for analyzing the works performed so

far in privacy-preserving data mining (PPDM). Clifton et al. [12] explore various privacy-preserving tools for distributed data mining. They present different constraints of privacy-preserving distributed data mining applications.

Partitioned data-based data mining has been receiving increasing attention, as well. Privacy-preserving naïve Bayes classifier for horizontally partitioned data (HPD) is discussed by Kantarcioglu and Vaidya [12], where they assume that data is horizontally partitioned. They show that using secure summation and logarithm, they can learn distributed naïve Bayes classifier securely. Although their protocols are very efficient, they compromise a little on security. Privacy-preserving association rules on HPD are discussed by Kantarcioglu and Clifton [30], where they assume that data is horizontally distributed among three or more parties. They address secure mining of association rules over HPD, while incorporating cryptographic techniques to minimize the shared data. Kaleli and Polat [29] study how to provide predictions for single items on partitioned data between two parties using naïve Bayesian classifier-based collaborative filtering schemes.

Merugu and Ghosh [38] present a framework for clustering horizontally distributed data in unsupervised and semi-supervised scenarios, taking into account privacy requirements and communication costs. Vaidya and Clifton [61] explore naïve Bayes classifier based on vertically partitioned data (VPD), where cryptographic techniques are employed to accomplish privacy.

Achieving predictions using numerical ratings on VPD without greatly exposing data owners' privacy is discussed in [46]. A solution to the privacy-preserving collaborative filtering (PPCF) on VPD problem is provided. The solution makes it possible for two parties to conduct filtering services using their joint data without revealing their data to each other. The results show that the proposed scheme produces accurate predictions compared with the true ratings. In [15], providing top-*N* recommendations on HPD is discussed. The authors discuss how to provide predictions to some items when data is horizontally distributed between two parties without violating their privacy.

2.3 Distributed HMMs-based Prediction with Privacy

HMMs can be employed to provide predictions. In other words, they are used to solve the following problem: Given the model $\lambda = (A, B, \pi)$, how $P(O|\lambda)$ can be computed, the probability of occurrence of the observation sequence $O = O_1, O_2, \ldots, O_T$. Given the model $\lambda = (A, B, \pi)$, using matrix notation, it can be defined A, B, and π, as follows:

$$A = [a_{ij}]_{NxN},$$

where $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, N$.

$$B = [b_j(k)]_{NxM},$$

where $j = 1, 2, \ldots, N$ and $k = 1, 2, \ldots, M$.

$$\pi = [\pi_i]_{Nx1},$$

for $i = 1, 2, \ldots, N$.

Dugad and Desai [15] propose a forward-backward procedure to compute $P(O|\lambda)$ efficiently, given the model $\lambda = (A, B, \pi)$, as follows: Consider the forward variable $\alpha_t(i)$, which is defined as $\alpha_t(i) = P(O_1, O_2, \ldots, O_t, i_t = i \mid \lambda)$. $\alpha_t(i)$ can be computed inductively, as follows:

A. Compute $\alpha_1(i)$ values first for all $1 \le i \le N$.

$$\alpha_1(i) = \pi_i b_i(O_1). \tag{2.1}$$

B. For $t = 1, 2, \ldots, T-1$, and $1 \le j \le N$, compute $\alpha_{t+1}(j)$.

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}). \tag{2.2}$$

C. Then;

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i). \tag{2.3}$$

Encryption methods are widely used to achieve privacy. In proposed schemes, encryption schemes with homomorphic property are also employed. Suppose that ξ is an encryption function, e is a public key, and q_1 and q_2 are private data values that are wanted to be hide. Homomorphic encryption property allows an addition, a subtraction, or a multiplication operation to be conducted based on the encrypted data without decrypting them, as follows: $\xi_e(q_1) \times \xi_e(q_2) = \xi_e(q_1 + q_2)$, $\xi_e(q_1) \times \xi_e(q_2) = \xi_e(q_1 - q_2)$, and $\xi_e(q_1) \times \xi_e(q_2) = \xi_e(q_1 \times q_2)$. The

homomorphic cryptosystems are useful to perform addition, subtraction, and multiplication operations based on private data. Several such systems are available and examples include the systems proposed by Paillier [43], Benaloh [8], and Naccache and Stern [40].

An HMM constructed for forecasting purposes might be distributed between various parties, even competing companies. This partition might be horizontally or vertically. In this section, how to compute $P(O|\lambda)$ when the model is distributed between two parties is investigated. Two-party schemes are presented. Such schemes can be easily extended to multi-party schemes. The model owners might want to integrate the split models in order to offer more truthful and dependable predictions for forecasting. It is more likely that the predictions generated based on the integrated model are more precise and reliable than the ones offered based on the split models alone. It sometimes might not be possible to compute $P(O|\lambda)$ from the split models alone. In order to achieve richer forecasting services and provide more trustworthy and accurate predictions, the model owners might decide to combine their models. However, due to various reasons especially privacy concerns, they do not want to disclose their models to each other. If privacy measures are introduced, the model owners might decide to combine their models is hypothesized.

Although the goal here is to achieve distributed HMMs-based forecasting without violating the model owners' privacy, it is not an easy job to define privacy succinctly. However, it can be defined, as follows: In this context, privacy means preventing the model owners from learning a_{ij}, $b_j(k)$, and π_i probability values held by each other. In other words, the parties should not be able to learn the split models owned by each other. Privacy-preserving schemes to compute $P(O|\lambda)$ from horizontally or vertically distributed models between parties without jeopardizing their privacy or while preventing them from learning the model parameters held by each other are presented.

Due to privacy measures, however, the accuracy of the predictions might become worse. In addition to achieving privacy, accurate predictions are wanted to be offered, as well. In other words, predictions computed based on the

distributed models with privacy concerns should be as accurate as the ones calculated from the integrated model without privacy concerns.

And finally, performance is another major concern that the model owners have. Providing predictions efficiently is vital. However, compared to other systems like recommender systems, which are expected to offer many predictions to many users in an online interaction, time requirements for HMMs are flexible. Since off-line computation times are not critical, if it is possible, some calculations can be performed off-line. Due to privacy concerns, it is expected that additional costs will emerge. To make it practical, extra communication and computation costs introduced by privacy measures should be small and negligible. They still allow the model owners to offer predictions efficiently.

Achieving privacy, accuracy, and performance at the same time is a challenge because they disagree with each other. Objective is to propose privacy-preserving schemes in which the model owners might be able to find equilibrium among privacy, accuracy, and performance. Proposed schemes are analyzed in terms of privacy, accuracy, and performance. The additional costs introduced due to privacy concerns are scrutinized.

As explained previously, the HMM constructed for forecasting purposes might be horizontally or vertically partitioned between various parties. In the following subsections, how to provide privacy-preserving predictions based on horizontally or vertically distributed HMMs between two parties are investigated.

2.3.1. Horizontally Distributed HMMs-based Prediction with Privacy

In horizontal partitioning, it is assumed that N is an even number and $N = 2h$, where h is the number of states held by each party. Therefore, in horizontal partitioning, it is assumed that the company C holds the part of the model for the first h states and the company D holds the remaining part of the model (the last h states). The horizontal partitioning of the model can be shown, as follows:

$A = \begin{bmatrix} A_C \\ A_D \end{bmatrix}$, $B = \begin{bmatrix} B_C \\ B_D \end{bmatrix}$, and $\pi = \begin{bmatrix} \pi_C \\ \pi_D \end{bmatrix}$, where it can be shown the parts of the model held by each party, as follows:

For $i = 1, 2, \ldots, h$ and $j = 1, 2, \ldots, N$,

$$A_C = [a_{ij}]_{hxN}.$$

And for $i = h + 1, h + 2, \ldots, N$ and $j = 1, 2, \ldots, N$,

$$A_D = [a_{ij}]_{hxN}.$$

Similarly, for $j = 1, 2, \ldots, h$ and $k = 1, 2, \ldots, M$,

$$B_C = [b_j(k)]_{hxM}.$$

And for $j = h + 1, h + 2, \ldots, N$ and $k = 1, 2, \ldots, M$,

$$B_D = [b_j(k)]_{hxM}.$$

Finally, for $i = 1, 2, \ldots, h$,

$$\pi_C = [\pi_i]_{hx1}.$$

And for $i = h + 1, h + 2, \ldots, N$,

$$\pi_D = [\pi_i]_{hx1}.$$

To compute $P(O|\lambda)$, there are three main steps, as explained previously. How to calculate $P(O|\lambda)$ based on a horizontally distributed HMM without violating model owners' privacy in three major steps is investigated, as follows:

A. *C* can compute $\alpha_1(i)=\pi_i b_i(O_1)$ values for all $1\leq i\leq h$, because *C* knows π_i and $b_i(O_1)$ probability values for $1\leq i\leq h$. Similarly, *D* can compute $\alpha_1(i)=\pi_i b_i(O_1)$ values for all $h+1\leq i\leq N$, because *D* knows π_i and $b_i(O_1)$ probability values for $h+1\leq i\leq N$. Therefore, *C* and *D* can compute $\alpha_1(i)$ values for all $1\leq i\leq h$ and $h+1\leq i\leq N$, respectively; and they store the values they calculated.

B. The parties can compute $\alpha_{t+1}(j)$ values without violating their privacy, as follows:

   a. Since the model is horizontally partitioned, $\alpha_2(j)$ values can be calculated, as follows for $t=1$:

      i. For $1\leq j\leq h$,

$$\alpha_2(j) = \left[\sum_{i=1}^{N} \alpha_1(i) a_{ij}\right] b_j(O_2) = \left[\sum_{i=1}^{h} \alpha_1(i) a_{ij} + \sum_{i=h+1}^{N} \alpha_1(i) a_{ij}\right] b_j(O_2)$$

$$\alpha_2(j) = [\Sigma_C + \Sigma_D] b_C = \Sigma_C b_C + \Sigma_D b_C,$$

where Σ_C and Σ_D represent the sum values that can be calculated by *C* and *D*, respectively, without the need of the other party's data. b_C represents probabilities of observing

symbol O_2 and they are known by C. To calculate $\alpha_2(j)$, the parties perform the following:

1. D first computes the each term($\alpha_1(i)a_{ij}$) of Σ_D value. It then encrypts them with its public key using a homomorphic encryption scheme. It finally sends them to C.
2. C divides the each term($\alpha_1(i)a_{ij}$) of Σ_C value into z_C random values, where $\alpha_1(\mathrm{i})\mathrm{a}_{\mathrm{ij}} = \sum_{C1=1}^{z_C} Z_{C1}$; z_C is a uniform random integer from a range [1, β_C] and Z_{C1} represents random numbers for $C1 = 1, \ldots, z_C$. Since D does not know β_C, it will not be able to learn z_C values, either. Note that C selects different β_C values for each term so that each term is divided into different numbers of random values.
3. After that C encrypts each random value with D's public key using a homomorphic encryption scheme.
4. It encrypts the corresponding b_C values with D's public key using a homomorphic encryption scheme, as well. It then multiplies each encrypted values both received from D and the values it has by corresponding encrypted b_C values using homomorphic encryption scheme.
5. It finally permutes the encrypted results of such multiplications using a permutation function f_{pC} and sends them to D.
6. D decrypts them and calculates the sum values($\alpha_2(j) = \Sigma_C b_C + \Sigma_D b_C$). It finally stores them for $1 \le j \le h$.
7. Due to β_C values and f_{pC}, D will not be able to learn the $\alpha_1(i)a_{ij}$ and b_C values held by C.

ii. For $h+1 \le j \le N$,

$$\alpha_2(j) = \left[\sum_{i=1}^{N} \alpha_1(i) a_{ij}\right] b_j(O_2) = \left[\sum_{i=1}^{h} \alpha_1(i) a_{ij} + \sum_{i=h+1}^{N} \alpha_1(i) a_{ij}\right] b_j(O_2)$$

$$\alpha_2(j) = [\Sigma_C + \Sigma_D] b_D = \Sigma_C b_D + \Sigma_D b_D,$$

where b_D values represent probabilities of observing symbol O_2 and they are known by *D*. The parties can similarly calculate $\alpha_2(j)$ values without jeopardizing their privacy. Note that in this case, *C* and *D* switch their roles.

b. For t=2, 3, . . ., T-1, $\alpha_{t+1}(j)$ values can be computed, as follows:

   i. For $1 \le j \le h$,

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}) = \left[\sum_{i=1}^{h} \alpha_{Dt}(i) a_{Cij} + \sum_{i=h+1}^{N} \alpha_{Ct}(i) a_{Dij}\right] b_{Cj}(O_{t+1}),$$

   where the values subscripted with *C* and *D* are held by *C* and *D*, respectively.

   1. *C* multiplies each a_C and α_C values by b_C values and computes $ab_C = a_{Cij} b_{Cj}(O_{t+1})$ and $\alpha b_C = \alpha_{Ct}(i) b_{Cj}(O_{t+1})$ values.
   2. It divides ab_C and αb_C values into z_{C2} and z_{C3} random values, respectively, where $ab_C = \sum_{C2=1}^{z_{C2}} Z_{C2}$ and $\alpha b_C = \sum_{C3=1}^{z_{C3}} Z_{C3}$; z_{C2} and z_{C3} are uniform random integers from a range [1, γ_C] and Z_{C2} and Z_{C3} represent random numbers for $C2$=1, . . . , z_{C2} and $C3$=1, . . . , z_{C3}, respectively. Since *D* does not know γ_C, it will not be able to learn z_{C2} and z_{C3} values, either. Note that *C* selects different γ_C values for each term so that each term is divided into different numbers of random values.
   3. *C* encrypts each value with *D*'s public key using a homomorphic encryption scheme.
   4. *D* encrypts each α_D and a_D values with its public key using a homomorphic encryption scheme and sends the encrypted values to *C*.

5. *C* then finds the multiplications of the encrypted values received from *D* and the corresponding values encrypted by it using the homomorphic encryption property.
6. *C* finally permutes the results using a permutation function f_{pC1} and sends them to *D*.
7. *D* decrypts the received encrypted values and calculates the sum values($\alpha_{t+1}(j)$). It finally stores them for $1 \le j \le h$.
8. Due to γ_C values and f_{pC1}, *D* will not be able to learn α, *a*, and *b* values held by *C*.

ii. For $h+1 \le j \le N$,

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}) = \left[\sum_{i=1}^{h} \alpha_{Ct}(i) a_{Dij} + \sum_{i=h+1}^{N} \alpha_{Dt}(i) a_{Cij}\right] b_{Dj}(O_{t+1}).$$

The parties can similarly calculate $\alpha_{t+1}(j)$ values with privacy while switching their roles.

C. At the end of the computations for $t=T-1$, the parties own $\alpha_T(i)$ values for $1 \le i \le h$ and $h+1 \le i \le N$, respectively. They compute $P(O|\lambda)$, as follows:

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) = \sum_{i=1}^{N_h} \alpha_{CT}(i) + \sum_{i=N_h+1}^{N} \alpha_{DT}(i).$$

Each party finds sum of $\alpha_T(i)$ values they hold and they exchange them. They finally find $P(O|\lambda)$ by summing two aggregated values.

2.3.2. Vertically Distributed HMMs-based Forecasting with Privacy

In vertical partitioning, the model is distributed between two parties (*C* and *D*), as follows: $A = [A_C \quad A_D]$ and $B = [B_C \quad B_D]$. Note that, unlike horizontal partitioning, in vertical partitioning, the initial state probabilities are known by both parties. Therefore, the matrix, π, is known by both companies. It is assumed that the transition probabilities from one state to the first *y* states are held by *C* and the remaining ones are held by *D*; and $N = 2y$. Moreover, it is assumed assume that $M = 2v$ and the observation probabilities for the first *v* symbols are held by *C*

and the remaining ones are held by *D*. The distributed model can be shown, as follows:

$A_C = [a_{ij}]_{Nxy}$ for $1 \le i \le N$ and $1 \le j \le y$.

$A_D = [a_{ij}]_{Nxy}$ for $1 \le i \le N$ and $y+1 \le j \le N$.

$B_C = [b_j(k)]_{Nxv}$ for $1 \le j \le N$ and $1 \le k \le v$.

$B_D = [b_j(k)]_{Nxv}$ for $1 \le j \le N$ and $v+1 \le k \le M$.

Since the model is vertically distributed, $\pi = \pi_C = \pi_D$.

Unlike horizontal partitioning, in vertical partitioning, the computations depend on what party holds the observation symbol. We investigate how to calculate $P(O|\lambda)$ on a vertically distributed HMM with privacy in three major steps, as follows:

A. To compute $\alpha_1(i)=\pi_i b_i(O_1)$ values for all $1 \le i \le N$, $b_i(O_1)$ probabilities are needed. Such values are known by the party that owns O_1. Such party might be *C* or *D*. Therefore, the party that holds O_1 computes $\alpha_1(i)$ values and stores them.

B. The parties can compute $\alpha_{t+1}(j)$ values with privacy, as follows:

    a. Since the model is vertically partitioned and observation symbols might be held either *C* or *D*, $\alpha_2(j)$ values can be calculated, as follows for $t=1$:

        i. For $1 \le j \le y$, there are four possible cases, as seen in Table 2.1. We can discuss how the parties can compute $\alpha_2(j)$ values considering four cases with privacy.

            1. **Case1**: Since O_2 is held by *C*, it knows $b(O_2)$. Moreover, it knows α_1 and a_{ij} values. Therefore, it computes $\alpha_2(j)$ values, encrypts them using its public key, sends them to *D*, which keeps them in encrypted form.

            2. **Case2**: *C* computes $\alpha_1 a_{ij}$ values, encrypts them with its public key using homomorphic encryption scheme, and sends them to *D*. *D* encrypts $b(O_2)$ values with *C*'s public key using homomorphic

encryption scheme. It then finds $\alpha_2(j)$ values by multiplying encrypted $\alpha_1 a_{ij}$ values with encrypted $b(O_2)$ values using homomorphic property. It finally keeps them in encrypted from.

3. **Case3**: *C* computes $a_{ij}b(O_2)$ values, encrypts them with its public key using homomorphic encryption scheme, and sends them to *D*. *D* divides each $\alpha_1(i)$ value into z_D random values, where $\alpha_1(\mathrm{i}) = \sum_{D1=1}^{z_{D1}} Z_{D1}$; z_{D1} is a uniform random integer from a range [1, δ_D] and Z_{D1} represents random numbers for $D1=1, \ldots,$ z_{D1}. Since *C* does not know δ_D, it will not be able to learn z_{D1} values, either. Note that *D* selects different δ_D values for each term so that each term is divided into different numbers of random values. It then encrypts each value generated after random division with *C*'s public key using homomorphic encryption scheme and multiplies them with encrypted $a_{ij}b(O_2)$ values using homomorphic property. It finally permutes the results using a permutation function f_{pD} and sends them to *C*, which decrypts them and calculates $\alpha_2(j)$ values. *C* then encrypts them with its public key and finally sends them to *D*, which keeps them in encrypted form. Due to δ_D values and f_{pD}, *C* will not be able to learn α values held by *D*.
4. **Case4**: *D* computes $\alpha_1(i)b(O_2)$ values, encrypts them with its public using homomorphic encryption scheme, and sends them to *C*. *C* divides each a_{ij} value into z_{C4} random values, where $a_{\mathrm{ij}} = \sum_{C4=1}^{z_{C4}} Z_{C4}$; z_{C4} is a uniform random integer from a range [1, δ_C] and Z_{C4} represents random numbers for $C4=1, \ldots,$

z_C. Since D does not know δ_C, it will not be able to learn z_{C4} values, either. Note that C selects different δ_C values for each term so that each term is divided into different numbers of random values. It then encrypts each value generated after random division with D's public key using homomorphic encryption scheme and multiplies them with encrypted $\alpha_1(i)b(O_2)$ values using homomorphic property. It finally permutes the results using a permutation function f_{pC1} and sends them to D, which decrypts them and calculates $\alpha_2(j)$ values. D then encrypts them with its public key and keeps them in encrypted form. Due to δ_C values and f_{pC1}, D will not be able to learn the a_{ij} values held by C.

Table 2. 1 Various Cases for t =1 & 1≤j≤y

Cases	α_1 held by	O_2 held by	C can compute	D can compute
Case1	C	C	αab	-
Case2	C	D	αa	b
Case3	D	C	ab	α
Case4	D	D	a	αb

ii. For $y+1 \leq j \leq N$, there are four possible cases, as seen in Table 2.2. The parties can compute $\alpha_2(j)$ values as they do for $1 \leq j \leq y$ while switching their roles.

Table 2. 2 Various Cases for t =1 & y + 1≤j≤N

Cases	α_1 held	O_2 held	C can	D can
Case1	D	D	-	αab
Case2	D	C	b	αa
Case3	C	D	α	ab
Case4	C	C	αb	a

b. For $t=2, 3, \ldots, T-1$, $\alpha_{t+1}(j)$ values can be computed, as follows:

i. For $1 \leq j \leq y$, there are two possible cases, as seen in Table 2.3. The parties can compute $\alpha_{t+1}(j)$ values considering two possible cases with privacy.

1. **Case1**: *C* encrypts *a* values with its public key using homomorphic encryption scheme and sends them to *D*. *D* first encrypts $b_j(O_{t+1})$ values with *C*'s public key using homomorphic encryption scheme and computes $\alpha_t(j)b_j(O_{t+1})$ values using homomorphic property. It then again uses homomorphic property to calculate $\alpha_t(j)a_{ij}b_j(O_{t+1})$. Note that such results are encrypted with *C*'s public key. To prevent the *C* from deriving data, *D* generates bogus data and encrypts them with *C*'s public key. It inserts them into encrypted $\alpha_{t+1}(j)$ values and permutes them using a permutation function f_{pD1}. It finally sends the permuted results to *C*, which first decrypts them. Due to bogus data and f_{pD1}, *C* is not able to derive data. *C* finds the sum by aggregating the received values. After encrypting it with its public key, sends the sum to *D*. Since *D* knows the sum of the bogus data, it then can get rid of it by subtracting it from the received aggregate data using homomorphic property to obtain the encrypted $\alpha_{t+1}(j)$ values. It finally stores such encrypted results.
2. **Case2**: The computations are similar to the ones in Case1. *C* computes $a_{ij}b_j(O_{t+1})$ values, encrypts them as done in Case1, and sends them to *D*. *D* performs the similar steps as in Case1 and sends the permuted results including the bogus data to *C*. *C* performs the same steps as done in Case1 and sends the result to *D*, which finds the encrypted $\alpha_{t+1}(j)$ values and keeps them as done in Case1.

Table 2. 3 Various Cases for t = 2, 3, . . ., T-1 & 1≤j≤y

Cases	α_t held by	O_{t+1} held by	*C* can	*D* can compute
Case1	*D*	*D*	*a*	*αb*
Case2	*D*	*C*	*ab*	*α*

ii. For $y+1 \leq j \leq N$, there are two possible cases, as displayed in Table 2.4. As seen from the cases, the parties can compute $\alpha_{t+1}(j)$ values as they do for $1 \leq j \leq y$ by switching their roles.

Table 2. 4 Various Cases for t = 2, 3, . . ., T-1 & y + 1≤j≤N

Cases	α_t held	O_{t+1} held	*C* can	*D* can
Case1	*C*	*C*	αb	*a*
Case2	*C*	*D*	α	*ab*

C. As in horizontal partitioning, at the end of the computations for $t=T-1$, the parties own $\alpha_T(i)$ values for $1 \leq i \leq y$ and $y+1 \leq i \leq N$, respectively. As explained in previously, they can compute $P(O|\lambda)$ without jeopardizing their privacy.

2.5. Privacy, Accuracy, and Performance Analysis

To preserve the model owners' privacy, it is mainly utilized homomorphic encryption, permutation, random division, and inserting bogus data. In homomorphic encryption schemes, private data items are encrypted with the sender's public key. In order to decrypt the encrypted data, the sender's corresponding private key is needed. However, that key is known by the sender only. Since the receiver does not know the sender's private key, it will not be able to decrypt the encrypted data. Therefore, it is not able to learn the private data held by the sender. Note that the sender might be *C* (or *D*) and the receiver might be *D* (or *C*) in proposed schemes.

In order to prevent each other from learning the order of the data items, the parties make use of permutation functions. Although *C* or *D* does not know the exact order of the received values due to permutation functions, they might guess it with a probability. For example, in horizontally distributed HMM-based scheme, *C* permutes, on average, $h\beta_C/2$ values using a permutation function f_{pC}. Therefore, for *D*, the probability of guessing the correct order of the received values is 1 out of $(h\beta_C/2)!$. Remember that *h* shows the number of states held by *C*. Also note that with increasing β_C values, such probability decreases. For other proposed schemes and/or cases, privacy analysis of using permutation functions can be similarly done.

In addition to using homomorphic encryption and permutation, the parties also apply random division to accomplish privacy. To simplify the discussion, random division is analyzed in terms of privacy for horizontally distributed HMM-based scheme only. After dividing αa values into different numbers of random values and permuting them, *C* sends them to *D*. Since *D* does not know β_C and uniformly randomly selected z_C values, it does not learn which values are part of which αa value. Therefore, it will not be able to derive αa values held by *C*. However, as explained previously, it might be able to guess them with a probability. Although *D* does not know β_C, it knows how many random values *h* number of αa values are divided into. If the number of received values from *C* is C_Σ, then, on average, *D* can estimate the value of β_C as $\beta_C^{'} = hC_\Sigma / 2$. The probability for *D* then to guess the number of random values each αa value are divided into is 1 out of $(\beta_C^{'})^h$. As expected, with increasing β_C, privacy level improves. Similar analyses can be done for other schemes and/or cases.

Due to bogus data items, it becomes a challenge to figure out the true αab values. Although the receiver does not know which received values are true αab values, it knows the number of such values. It can guess the true ones with a probability. If it is assumed that the number of bogus data values is *y'*, then for *D*, the probability of guessing the correct αab values is 1 out of $G_{y'}^{y+y'}$, where $G_{y'}^{y+y'}$ represents the number of ways of picking *y'* unordered outcomes from *y* + *y'* possibilities.

In conclusion, it can be said that the proposed schemes are secure and they allow the model owners to integrate their split models to provide predictions without jeopardizing their privacy. In case of distributed HMMs, model holders feel comfortable to combine their models for forecasting purposes without revealing them to each other.

Due to privacy measures, it is expected that accuracy becomes worse. When privacy is achieved through the use of randomization, it is more likely that accuracy deteriorates. However, this is not the case for proposed schemes. Although it is utilized various privacy-preserving techniques, accuracy does not

worsen. Described schemes achieve the same accuracy as the models without privacy concerns.

Performance is one of the major concerns for forecasting schemes. It is vital to offer many predictions online. Due to privacy concerns, described schemes introduce additional computation and communication costs. Due to homomorphic encryption, assuming $N > T$, the numbers of encryptions and decryptions are of the order of $Nh\beta_C$ for horizontally distributed schemes. The numbers of encryptions and decryptions can be similarly computed for vertically distributed methods. Benchmarks for the CRYPTO++ toolkit from www.eskimo.com/~weidai/benchmarks.html can be used to determine the running times of cryptographic algorithms [11].

The number of multiplications to calculate the $P(O|\lambda)$ without privacy concerns is of the order of N^2T [59]. Due to random divisions, the number of multiplications increases by a factor of order of $N\beta_C$ for horizontally distributed schemes. The number of multiplications similarly increases for vertically distributed schemes. Compared to encryption/decryption and multiplications, additional computations due to permutation and inserting bogus data are negligible.

The number of communications increases due to privacy concerns because the model is distributed between two parties. For horizontally distributed schemes, the number of communications is $4T$ or the order of T. It can be similarly estimated for vertically distributed methods. As expected, with privacy concerns, performance degrades. There is a trade-off between privacy, accuracy, and performance. Since there is no accuracy loss due to privacy concerns, performance degrades are inevitable.

2.6. Conclusions

It is shown that it is possible to provide predictions based on horizontally or vertically distributed HMMs between two parties without violating their privacy. With the advent of the Internet, privacy happens to be vital. To protect the model owners' privacy while still allowing them to provide predictions by combining their split models, approaches are proposed. It is shown that such

methods are secure. The schemes prevent the model owners from deriving data about each other's models.

In addition to preserving privacy, providing predictions with decent accuracy efficiently is also imperative for the success of HMMs. Therefore, the proposed schemes are analyzed in terms of accuracy and performance. Fortunately, distributed HMMs-based schemes with privacy accomplish the same accuracy as the ones without privacy concerns. On the flip side, performance degrades because privacy, accuracy, and efficiency conflict with each other. By sacrificing on performance, described schemes make it feasible to integrate split HMMs between two parties, even competing companies, without revealing the models to each other.

To accomplish privacy, various techniques are employed such as homomorphic encryption, permutation, random division, and introducing bogus data. The model owners can adjust the parameters of privacy-preserving techniques that is used in order to reach required levels of performance and privacy. For example, they might determine the values of β, γ, and δ based on privacy and performance levels they want.

In this chapter, horizontally or vertically distributed HMMs are investigated. Such model partition might be hybrid too. It is more likely that described proposed schemes can be modified in such a way to provide predictions on hybrid distributed HMMs with privacy. There still remains work to be done to study hybrid distributed HMMs-based forecasting with privacy. Although two-party schemes only are scrutinized, the model might be partitioned among more than two parties. The schemes can be expanded to multi-party schemes. In that case, communication bottlenecks will become a major issue. It will be deeply explored how two-party schemes can be expanded to multi-party methods.

Supplementary communication costs introduced due to privacy concerns are significant. Compared to other online prediction systems, online time for HMMs might not be that critical. Even if this is the case, it will be searched for solutions to reduce the additional costs both communication and computation. This might be achieved by sacrificing on accuracy and/or privacy. Randomization techniques might be utilized to overcome additional costs. Some aggregate data,

whose disclosure does not significantly violate privacy, can be revealed to improve performance.

To sum up, described proposed schemes make it feasible to suggest predictions based on distributed HMMs between various parties while preserving their privacy, even though they introduce some extra costs. Those companies that do not want to integrate their models with others due to privacy, legal, and financial reasons can use proposed schemes. Moreover, they can adjust various parameters to reach privacy and performance levels they want.

3. FINDING THE STATE SEQUENCE MAXIMIZING P(O,I|λ) ON DISTRIBUTED HMMS WITH PRIVACY

3.1. Introduction

With increasing popularity of model-based forecasting, hidden Markov models (HMMs) has become one of the widely used models in science, engineering and many other areas like cryptanalysis, speech recognition, sign language recognition, gesture and body motion recognition, optical character recognition, machine translation which investigates the use of computer software to translate text or speech from one natural language to another, robot navigation, bioinformatics, finance, economics, and data mining. In bio-informatics, HMMs are employed for prediction of protein-coding regions in genome sequences, modeling families of related DNA or protein sequences, and prediction of secondary structure elements from protein primary sequences.

To provide predictions, regression analysis, Bayesian networks, neural networks, HMMs, and so on are widely used. For various applications, an appropriate model is chosen for forecasting purposes. Such models have their own advantages and disadvantages. HMMs are very powerful models due to the following reasons: They are transparent, employ prior knowledge, and they can be combined into larger models. Although they have disadvantages like assumption that states are independent and low speed, their advantages surpass such drawbacks. They are constructed from historical data.

HMMs are extensively used in prediction, speech recognition, finance, and so on. Begletier et al. [7] propose to use variable order Markov models for prediction of discrete sequences. Rabiner [51] shows how HMMs can be applied to selected problems in speech recognition. The theories of HMMs from various concepts are presented. Henderson et al. [24] describe a new HMM to study how to segment human DNA into three regions. Hassan and Nath [23] study how to employ HMMs to predict stock market trends. They forecast stock prices for markets.

Besides calculating $P(O \mid \lambda)$, the probability of occurrence of an observation sequence $O = O_1, O_2, ..., O_T$, given the model; many applications of HMMs are utilized to solve the following problem, as well: Given the model,

$\lambda = (A, B, \pi)$, how to choose a state sequence $I = I_1, I_2, ..., I_T$ so that $P(O, I \mid \lambda)$, the joint probability of the observation sequence $O = O_1, O_2, ..., O_T$ and the state sequence given the model is maximized.

Data owners or collectors are able to construct HMMs from historical or collected data. After generating the model, they can start providing forecasting services to other customers or vendors. When the model is held by party, it is an easy task to offer such services. The party that wants to obtain predictions send the observation sequence to the model owners. Using the model, the owners then can compute predictions based on the sequence and finally, send the result to the query owner. Although it is trivial performing such task when the model is held by single party only, it becomes a challenge to conduct the similar jobs when the model is distributed between various parties. This partition can be horizontal or vertical. The model owners might want to integrate their split models; however, due to privacy, legal, and financial reasons, they do not want to reveal their private models to each other. To be able to provide predictions and to achieve more accurate and dependable services, they should integrate their models. If privacy measures are introduced, they might decide to combine their models.

In this part, how to choose a state sequence $I = I_1, I_2, ..., I_T$ so that $P(O, I \mid \lambda)$ of the observation sequence $O = O_1, O_2, ..., O_T$ and the state sequence is maximized when the model is horizontally or vertically partitioned between two companies without deeply violating the model holders' privacy are investigated. Privacy-preserving schemes to achieve goal is proposed. The proposed schemes are analyzed in terms of accuracy, privacy, and supplementary costs.

3.2. Related Work

With the evolution of the Internet and the computerized works, privacy protection has become imperative. Individual users and companies have concerns about their privacy. Performing various data mining functionalities while preserving privacy is increasingly receiving attention. Moreover, conducting different data mining tasks based on distributed data while preserving parties' privacy is also becoming imperative. To perform richer data mining, provide

better services, and offer more dependable outcomes, distributed data-based computations become popular without greatly jeopardizing data owners' privacy.

Privacy-preserving data mining (PPDM) on distributed data has been receiving increasing attention during the past few years. Clifton et al. [12] explore various privacy-preserving tools for distributed data mining. Privacy-preserving naive Bayes classifier (NBC) for horizontally partitioned data (HPD) is discussed by Kantarcioglu and Vaidya [32]. They show that using secure summation and logarithm, they can learn distributed NBC securely. Privacy-preserving association rules on HPD are discussed in [30]. They address secure mining of association rules over HPD, while incorporating cryptographic techniques to minimize the shared data. Kantarcioglu and Clifton [31] present a method for privately computing *k-nn* classification from distributed sources without revealing any information about the sources or their data, other than that revealed by the final classification result. Wright and Yang [65] present a privacy-preserving protocol for learning the Bayesian network structure for distributed heterogenous data.

Sanil et al. [52] describe an algorithm to conduct a linear regression analysis based on vertically partitioned data (VPD). The agencies who poses a few attributes of every data record do not want to disclose values of their own attributes while conducting regression analysis on joint data. Vaidya and Clifton [15-17] present privacy-preserving methods for different data mining tasks on VPD.

They discuss privacy-preserving association rule mining, NBC, and *k*-means clustering using VPD. Vaidya [62] develops and evaluates new algorithms to efficiently solve several types of distributed computations over large data sets in a secure manner. Oliveira and Zaiane [42] address the problem of protecting the underlying attribute values when sharing data for clustering. Polat and Du [46] study how to provide predictions on VPD. The authors discuss how to provide predictions when data is vertically distributed between two parties without violating their privacy. In [50], the authors present privacy-preserving schemes to offer top-*N* recommendations on distributed data.

3.3 Distributed HMMs-based Prediction with Privacy

As explained previously, HMMs are employed to find the state sequence *I* that maximizes $P(O,I \mid \lambda)$ the joint probability of the observation sequence $O=O_1,O_2,.....,O_T$ and the state sequence given the model. To solve this problem, the Viterbi algorithm, which is briefly defined, as follows, can be employed [15]: The Viterbi algorithm is an inductive algorithm in which at each instant the best possible state sequence is kept for each of the *N* states as the intermediate state for the desired observation sequence $O=O_1,O_2,.....,O_T$. In this way, finally, the best path is found for each of the *N* states as the last state. Out of these, the one is selected, which has the highest probability. Suppose being currently in state *i* and considering visiting state *j* next. It can be said that the weight on the path from state *i* to state *j* is $-\ln(a_{ij}b_i(O_1))$, where O_t is the observation symbol selected after visiting state *j*. This is the same symbol that appears in the given observation sequence $O=O_1,O_2,.....,O_T$. The corresponding weight is $-\ln(\pi_i b_i(O_1))$, when the initial state is selected as state *i*; and this will be called the initial weight. The weight of a sequence of states is defined as the sum of the weights on the adjacent states. This actually corresponds to multiplying the corresponding probabilities. Finding the optimum sequence is finding the path of minimum weight through which the given observation sequence occurs. $\delta_t(i)$ denotes the weight accumulated when in state *i* at time *t* as the algorithm proceeds. $\psi_1(j)$ represents the state at time *t*-1, which has the lowest cost corresponding to the state transition to state *j* at time *t*. There are four main steps, as follows:

1. Initialization For $1 \le i \le N$

$$\delta_1(i) = -\ln(\pi_i) - \ln(b_i(O_1))$$

$$\Psi_1(i) = 0$$

2. Recursive computation

   For $2 \le i \le T$ for $1 \le j \le N$

$$\delta_t(j) = \min_{1 \le i \le N} [\, \delta_{t-1}(i) - \ln(a_{ij}) \,] - \ln(b_j(O_t))$$

3. Termination

$$P^* = \min_{1 \le i \le N} [\delta_T(i)]$$

$$q_T^* = \arg\min_{1 \le i \le N} [\delta_T(i)]$$

4. Tracing back the optimal state sequence

For $t = T-1, T-2, \ldots\ldots\ldots\ldots, 1$

$$q_T^* = \psi_{t+1}(q_{T+1}^*)$$

Hence exp(-P^*) gives the required state-optimized probability, and $Q^* = \{q_1^*, q_2^*, ..., q_T^*\}$ is the optimal state sequence. The complexity of the Viterbi algorithm is order of N^2T. Without privacy concerns and when the model is owned by a single party, it is an easy task to solve the problem by utilizing the Viterbi algorithm. However, when the model is distributed between two parties; and they want to integrate their split models while preserving their privacy, it becomes a challenge. Although it is not easy to define privacy succinctly, in this context, it can be briefly defined, as follows: Remember that HMMs are represented by $a_{i,j}$, $b_j(k)$, and π_i values. In a distributed environment, such parameters or values are partitioned between two parties. Privacy means preventing the parties from learning such values held by each other. In other words, the parties should not be able to learn the split models owned by each other.

In addition to providing privacy, the proposed privacy-preserving schemes should allow the parties to generate meaningful outcomes based on the integrated model with privacy concerns. Therefore, besides privacy, providing HMM-based services with decent accuracy is another goal should be achieved. Due to privacy concerns, however, proposed solutions do not cause any loss in accuracy. With privacy concerns, it is still possible to generate the same outcomes as in when there is no privacy protection.

And finally, online performance is another major concern that the model owners have. Finding the state sequence, which gives the maximum $P(O,I \mid \lambda)$ value effciently is vital. However, compared to other systems like recommender systems, which are expected to offer many predictions to many users in an online interaction, time requirements for HMMs are flexible. Since off-line costs are not critical, if it is possible, as many calculations as possible should be performed off-line. Due to privacy concerns, it is expected that additional costs will emerge. To

make it practical, extra costs like computation, communication, and storage costs introduced due to privacy concerns should be negligible. Since privacy, accuracy, and effciency are conflicting, there is a trade-off between them.

Since the proposed privacy-preserving schemes achieve the same accuracy as the schemes without privacy concerns while preserving the model owners' privacy, it is a reasonable trade-off to sacrifice on performance.

Privacy-preserving schemes are presented to maximize $P(O,I|\lambda)$ from horizontally or vertically distributed models between two parties without violating their privacy. Since there are four major steps in the Viterbi algorithm, how the proposed schemes compute such four steps with privacy is shown. The details of the schemes explained in the following subsections.

3.3.1 Horizontally Distributed HMMs-based Schemes with Privacy

In horizontal partitioning, it is assumed that N is an even number and $N = 2h$, where h is number of states held by each party. Therefore, it is assumed that company C holds the part of the model for the first h states and company D holds the remaining part (the last h states). Horizontally distributed HMMs can be shown, as in Section 2.3.1.

How to find the state sequence I maximizing $P(O,I|\lambda)$ on a horizontally distributed HMM between two parties with privacy in four major steps is investigated, as follows:

1. *Initialization*: At the beginning, $\psi_1(j)$ values are all 0. For $1 \le i \le h$ and for $h + 1 \le i \le N$, C and D can compute $\delta_1(i)$ values because they know the corresponding required values (corresponding $b_i(O_1)$, and π_i values).

2. *Recursive computation*: For $2 \le t \le T$ for $1 \le j \le h$, the parties perform the following:

(a) D finds the local minimum value, which is $min_{Dtj} = \min_{1 \le i \le h} [\delta_{t-1}(i) - \ln(a_{ij})]$. Similarly, C finds $min_{Ctj} = \min_{h+1 \le i \le N} [\delta_{t-1}(i) - \ln(a_{ij})]$. Note that the parties are able to calculate such values without the need of the other party's data.

(b) To calculate $\delta_t(j) = \min(min_C, min_D) - ln_C$, the parties need to find $min(min_C, min_D)$, where $ln_C = ln(b_j(O_t))$. Note that C holds $b_j(O_t)$ values. D uniformly randomly generates a large enough random number (R_{Dt1}).

(c) *D* then computes $D'_{tj} = min_{Dtj+} R_{Dt1}$ and sends it to *C*.

(d) *C* uniformly randomly generates a large enough random number (R_{Ct}). It then finds $C'_{tj} = min_{Ctj} - ln(b_j(O_t)) - R_{Ct}$ and $D''_{tj} = min_{Dtj+} R_{Dt1} - ln(b_j(O_t)) - R_{Ct}$; and sends them to *D*.

(e) *D* computes $D'''_{tj} = min_{Dtj} - ln(b_j(O_t)) - R_{Ct}$. It then finds min_{tj} =min(C'_{tj} , D'''_{tj}).

(f) For those computations in the subsequent iterations, the parties need to compute $\delta_t(j)$. Therefore, *D* uniformly randomly generates a large enough random number (R_{Dt}), adds it to min_{tj} $-ln(b_j(O_t))-R_{Ct}$, and obtains min_{tj} - $ln(b_j(O_t))-R_{Ct}+R_{Dt}$. It then sends it to *C*. By adding R_{Ct} to that value, *C* gets min_{tj} - $ln(b_j(O_t)) + R_{Dt}$. In other words, *C* is able to get $\delta'_t(j) = \delta'_t(j) + R_{Dt} = min_{tj}$ - $ln(b_j(O_t)) + R_{Dt}$.

(g) After determining $\min_{1\le i\le N}[\delta_{t-1}(i) - \ln(a_{ij})]$, the parties then can easily figure out $\psi_t(j) = arg \min_{1\le i\le N}[\delta_{t-1}(i) - \ln(a_{ij})]$ and exchange such information.

For $2\le t\le T$ for $h+1\le j\le N$, the parties switch their roles and perform the same steps as they do for $1\le j\le h$. Since the parties do not need the exact δ_{t-1} values in order to find the $\min_{1\le i\le N}[\delta_{t-1}(i) - \ln(a_{ij})]$ values and the same random numbers are used to disguise δ_{t-1} values, the parties are still be able to find the minimum values. Remember that the parties add random numbers in each iteration. If they continue to do that, they will get aggregate values due to random numbers. To avoid from such case, in each iteration the parties first get rid of the random numbers they added in the previous step and then add a new random number. In addition, they can add uniformly randomly selected random numbers from a range [-α, α], where α is a positive integer.

3. *Termination*: In this step, the parties need to compute *P** and determine q_T^* .

Note that at the end of the recursive computation phase, *D* has $\delta'_{T_D} = \delta_{T_D} + R_{C_{T-1}}$ for $1\le j\le h$ and *C* has for $\delta'_{T_C} = \delta_{T_C} + R_{D_{T-1}}$ for $h+1\le j\le N$. For this purpose, they perform the following:

(a) Each party determines local minimum values based on the values they receive at the end of the recursive computation phase. *C* determines $\min'_{T_C} = \min_{h+1 \le i \le N} [\, \delta'_{T_C}(i) \,]$ while *D* determines $\min'_{T_D} = \min_{1 \le i \le h} \left[\delta'_{T_D}(i)\right]$.

(b) *D* computes $\min''_{T_D} = \min'_{T_D} + R_{D_{T-1}}$ and sends it to C. Remember that $\min'_{T_D}$ is masked with $R_{C_{T-1}}$.

(c) Since *C* knows $R_{C_{T-1}}$, it can find $\min''_{T_D} - R_{C_{T-1}}$ and compares the result with $\min'_{T_C}$ to find the global minimum.

(d) After determining P*, *C* then determines q_T^*.

(e) It finally informs *D* about q_T^* .

In termination phase, any party can act as a master party to determine *P** and q_T^* . Since this is the case, the parties can alternately act as a master party.

4. *Tracing back the optimal state sequence*: And finally, the parties are able to determine $q_T^* = \psi_{t+1}(q_{t+1}^*)$ by tracking back. This is an easy task because the parties exchange ψ_t values throughout the process. Again, one party can act as a master party and returns the result. Similarly, they can successively act as a master party.

3.3.2 Vertically Distributed HMMs-based Schemes with Privacy

Unlike horizontal partitioning, initial state probabilities are known by both parties. Therefore, π is known by both companies. It is assumed that transition probabilities from one state to the first *y* states are held by *C* and the remaining ones are held by *D*; and $N = 2y$. Moreover, it is assumed that $M = 2v$ and observation probabilities for the first *v* symbols are held by *C* and the remaining ones are held by *D*. The distributed model can be shown, as shown in Section 2.3.2.

Finally, since the model is vertically distributed, $\pi = \pi_C = \pi_D$. How to find the state sequence *I* maximizing $P(\mathrm{O}, I \mid \lambda)$ on a vertically distributed HMM with privacy in four major steps is investigated, as follows:

1. *Initialization*: Since the model is vertically distributed, both *C* and *D* knows π_i values for $1 \le i \le N$. To compute $\delta_1(i)$ for $1 \le i \le N$, which party holds the $b_i(O_1)$

values. The first item of the object sequence (O_1) might be held by either *C* or *D*. Therefore, the party that owns *O*1 can compute π_i values for all *i*; and saves them. Also note that at the beginning, $\psi_1(i)$ values are all 0.

2. *Recursive computation*: In order to calculate $\delta_t(j)$ and determine $\psi_t(j)$ values, it is important that δ_{t-1}, O_t, and a_{ij} values are held by which party. When they are all held by one of the parties only, it is trivial to achieve the recursive computations. However, when they are split, the parties need to collaborate. It is needed to be consider all possible cases because the above mentioned values might be held by either parties.

For $2 \le i \le T$ for $1 \le j \le y$, possible cases that might occur are summarized in Table 3.1. For each possible case, the parties perform the following:

Table 3. 1 Possible Scenarios ($2 \le i \le T$ & $1 \le j \le y$)

Cases	δ_{t-1} Held by	O_t Held by	a_{ij} Held by
Case 1	C	C	C
Case 2	C	D	C
Case 3	D	C	C
Case 4	D	D	C

(a) **Case 1**. *C* can easily compute $\delta_t(j)$ and determine $\psi_t(j)$ because it has the all required values.

(b) **Case 2**. *C* can determine $\min_{1 \le i \le N}[\delta'_{T_C}(i)]$ and $\psi_t(j)$ values by itself. However, it needs to collaborate with *D* to compute $\delta_t(j)$ values, which are needed in the subsequent steps. *D* uniformly randomly generates a random number (R_{Dt}) and disguises $-ln(b_j(O_t))$ values, for $1 \le j \le y$, by adding R_{Dt} to them. It then sends them to *C*, which finds masked $\delta_t(j)$ values and saves them.

(c) **Case 3.** *C* first uniformly randomly generates a random number (R_{Ct}), Disguises $-\ln(a_{ij})$ values, and sends them to *D*. Now, *D* can determine $\min_{1 \le i \le N}[\delta_{t-1}(i) - \ln(a_{ij})]$ and $\psi_t(j)$. To compute $\delta_t(j)$, *D* needs $-ln(b_j(O_t))$ values, which are held by *C*. Therefore, *C* disguises such

values with a random number (R_{Ct1}) and sends them to *D*. *D* then can compute the masked $\delta_t(j)$ values.

(d) **Case 4**. In this case, *C* has the a_{ij} values. To obtain required data, *D* needs a_{ij} values. So, *C* first perturbs them with a random number (R_{Ct}) and sends them to *D*. *D* then determines $\psi_t(j)$ and computes masked $\delta_t(j)$.

For $2 \leq t \leq T$ for $h+1 \leq j \leq N$, similar cases that might occur are summarized in Table 3. 2. For each possible case, the parties perform the same steps as they do for $2 \leq t \leq T$ for $1 \leq j \leq y$ explained above. However, in these cases, the parties switch their roles.

3.*Termination*: At the end of recursive computation, perturbed $\delta_T(j)$ values are held by *C* and *D* for $1 \leq j \leq y$ and $y+1 \leq j \leq N$, respectively. In order to compute P^* and determine q_T^* , the parties perform the same steps as they do when the model is horizontally distributed. Moreover, as explained previously, they can alternately act as a master party.

4.*Tracing back the optimal state sequence*: The parties can successfully achieve this step as they do the model is horizontally distributed.

Table 3. 2 Possible Scenarios ($2 \leq i \leq T$ & $1 \leq j \leq N$)

Cases	δ_{t-1} Held by	O_t Held by	a_{ij} Held by
Case 1	D	D	D
Case 2	D	C	D
Case 3	C	D	D
Case 4	C	C	D

3.4 Privacy Analysis

As explained before, the proposed privacy-preserving schemes should prevent the parties from learning the actual values of the model parameters. To determine the state sequence maximizing $P(O,I \mid \lambda)$, the model owners need to know $\psi_t(j)$ for all *t* and *j*. There is no way to prevent them from learning which state sequence achieves the maximum $P(O,I \mid \lambda)$ if they want to solve the problem.

As seen from the proposed solutions for horizontally partitioned HMMs, proposed schemes prevent the model owners from learning the actual values of model parameters. However, since the state sequence maximizing $P(O,I \mid \lambda)$ is

determined based on the comparisons of the local minimum values ($\delta_{t-1}(i) - \ln(a_{ij})$) held by both parties, the parties will learn whose value is smaller than the other. Therefore, they can determine that such local minimums are smaller or bigger than each other's local minimum values. Even if they have such information, they will not be able to learn the exact values of $\delta_{t-1}(i)$ and a_{ij}.

In horizontally distributed HMMs, the privacy-preserving computations are performed on local minimum values. The parties do not need to exchange all values of the model parameters. Therefore, they cannot learn such values that are not involved in collaborative computations. In both horizontally and vertically distributed HMMs, the parties perturb actual values using random numbers. Since the parties do not know such numbers, they will not be able to learn the actual values held by each other.

In the proposed schemes, the same random number is used to disguise *y* values. When either party is able to figure out the random number, it will learn all values. To improve privacy, it is proposed to use the following solution: Instead of using a single random number to hide all data items, they can divide their data values into two, four, eight, and so on groups; and perturb data items in each group independently using various random numbers. For example, they can divide their data items that they want to compare into two groups. They then disguise data items in each group using a different random number. After finding local minimums in these two groups, they perturb such values again. They finally perform one more step to find the global minimum value. When they divide their data items into 2^k groups, they use *k* random numbers for data disguising. They perform *k*-1 more steps to obtain the global minimum. As expected, with increasing *k*, privacy improves while performance degrades. Therefore, the parties are able to decide *k* in such a way to achieve required levels of privacy and performance. It can be said that proposed schemes are flexible in terms of achieving the required levels of privacy and efficiency.

At one end, the parties can perturb their private data values using a single random value, while at the other end, they can disguise them by dividing them into *y*/2 groups and using *y*/2 random numbers. They might group their data into *k* groups and disguise the data in each group using a different random number,

where $1 \leq k \leq y/2$. Although they will have an idea about the order of each other's private data values, such information is not enough to generate HMMs-based services. Moreover, in most cases, such data items are aggregate values such as values $\delta_{t-1}(i) - \ln(a_{ij})$ and $\delta_{t-1}(i)$, rather than π_i, a_{ij}, or $b_i(O_1)$ values or single model parameters. It becomes difficult for parties to derive true model parameters held by each other even if they learn aggregate values.

3.5 Accuracy and Overhead Costs Analysis

The proposed schemes should provide accurate results efficiently besides achieving privacy. Due to the underlying privacy-preserving schemes that it is proposed that, there is no accuracy losses. In other words, the model owners are able to achieve the same accuracy level with privacy concerns as they do without privacy concerns. Due to privacy concerns, the parties do not need to sacrifice on accuracy.

Although the parties are able to offer the same predictions with privacy concerns, it is expected that performance degrades due to privacy-preserving schemes because privacy, accuracy, and performance conflict with each other. The proposed schemes should be efficient; otherwise, it makes no sense to use them. Since off-line costs are not critical, online costs should be focused on. Moreover, in additional costs due to privacy concerns are interested. Such costs might be additional storage, communication, and computation costs. They should be small and allow the model owners to offer predictions efficiently.

The proposed vertically partitioned HMMs-based schemes do not introduce extra storage costs due to privacy concerns. However, supplementary storage costs due to the methods proposed for horizontally distributed HMMs is of the order of *T*. When the model is held by a single party, after receiving the object sequence, the model owner finds the prediction and sends it back to the query owner. Therefore, the number of communications is only two (or of the order of 1). However, when the model is split between two parties, communication costs increase. When the model is horizontally distributed, the number of communications is of the order of *NTk*. Similarly, in vertically partitioned HMMs-based schemes, the number of communications is of the order of *NTk*, except Case

1 in recursive computation phase. In that case, since one of the parties hold all required values to find predictions, the parties do not need to exchange anything. Therefore, the number of communications in this case is of the order of k.

Additional computation costs due to privacy concerns are negligible compared to extra communication costs. Remember that the original Viterbi algorithm's complexity is of the order of N^2T. In both horizontally and vertically distributed HMMs-based schemes, additional computations due to privacy concerns are random number generations, additions, and subtractions. The number of random number generations, additions, and subtractions is of the order of NTk. Remember that k might be chosen by the parties as an integer between 1 and log_2N inclusively in order to achieve required levels of privacy and performance, as explained previously. On average, compared to N, k is small and can be considered as a constant. Therefore, it can be said that the supplementary computation costs are small; and the parties are still able to generate outcomes efficiently based on distributed models without violating their privacy.

3.6 Conclusions and Future Work

HMMs are popular models used for forecasting purposes in many applications. When the model is held by a single party, it is a trivial task to generate outcomes based on the model. However, the model might be distributed between various parties even competing companies. In this part, how to choose a state sequence so that the joint probability of an observation sequence and a state sequence given an HMM is maximized is studied when the model is distributed between two parties, without violating their privacy. Privacy-preserving schemes for both horizontally and vertically partitioned HMMs is presented. The proposed methods is investigated in terms of privacy, accuracy, and performance.

First of all, due to privacy concerns, there is no accuracy loss. The same outcomes, generated by integrated model without privacy concerns, are also provided by the proposed schemes. Secondly, the secure proposed schemes is showed. Moreover, proposed solutions allow the parties to chose the parameters of privacy-preserving schemes in such a way to achieve required levels of privacy and performance. And finally, the solutions are scrutinized in terms of

supplementary costs due to privacy concerns. Acceptance of extra storage and computation costs are demonstrated. Proposed schemes, however, introduce additional communication costs. Since accuracy, privacy, and performance are conflicting goals, it is needed to be sacrificed on one of them. The proposed schemes are secure and output accurate results. It is believed that these features surpass extra communication costs. This is an acceptable trade-off because there is no such solution to accomplish accuracy, privacy, and good performance.

Only horizontal or vertical partition are considered. The partition might be hybrid. How to provide similar HMM-based services will be studied when the model is hybrid distributed between two parties without jeopardizing the model owners' privacy. Although it is assumed that the model is partitioned between two parties, it might be distributed between more than two parties. Proposed solutions can be extended to multi-party schemes. However, detail analysis should be done and solutions should be proposed to overcome the bottlenecks that might occur in multi-party schemes.

4. A NEW HYBRID RECOMMENDATION ALGORITHM WITH PRIVACY

4.1. Introduction

Collaborative filtering (CF) is a recent technique for filtering and recommendation purposes. CF is widely used in e-commerce, direct recommendations, and search engines to suggest items to users. Users can obtain referrals about many of their daily activities with the help of CF. Goldberg et al. [19] first coined the term "collaborative filtering" in the early nineties. CF systems predict how well a user, called an active user (*a*), will like or dislike an item that she has not purchased previously using other users' preferences [25]. Ahn et al. [5] propose simultaneously optimized case-based reasoning to increase the classification accuracy, where customers are classified into either purchasing or non-purchasing groups. They suggest simultaneous optimization using genetic algorithm.

To offer referrals, data collected from many users is used. Such data contains users' ratings, which show preferences of users about products. Users' preferences can be represented with numerical or binary ratings. After data collection, a user-item matrix is created. The database contains rating values (v_{uj}), where v_{uj} represents user u's ratings on item j. When a asks recommendations, the CF system or the server first finds similar users to a using various similarity metrics. It then computes referrals using their data. The main idea is that a will prefer those items that like-minded users prefer, or that dissimilar users do not [45].

With increasing popularity of the Internet and e-commerce, CF has been increasingly receiving attention. Many people are buying or selling various products over the Internet using e-commerce sites. E-companies can increase their sales and profits through CF. Schafer et al. [55] present an explanation of how recommender systems help e-commerce sites increase sales. However, since the number of users accessing the Internet and the number of products available over the Internet are increasing rapidly, it becomes vital to generate referrals efficiently. Customers do not want to spend too much time to get predictions. CF systems should provide referrals to many users within a limited time. In addition

to providing referrals efficiently, accuracy is another concern for both online vendors and customers. Inaccurate predictions lead angry customers. Shoppers usually prefer those companies with accurate referrals.

To generate accurate referrals efficiently, various approaches have been proposed. CF algorithms proposed so far can be classified as memory- or model-based algorithms. Memory-based algorithms operate over the entire user database to make predictions, while model-based CF algorithms use the user database to learn a model, which is then used for referrals [10]. Although memory-based schemes achieve higher accuracy, performance degrades rapidly with increasing number of users and/or items. Conversely, model-based approaches improve performance while accuracy diminishes. Both groups of algorithms have their advantages and disadvantages.

Privacy is becoming a major concern for users recently. Collecting users' preferences about many products to provide referrals poses various privacy risks. According to study conducted by Cranor [14], privacy risks are severe and many like unsolicited marketing, price discrimination, profiling users, being subject to government surveillance, and so on. Moreover, users are concerned about data transfer and misused. Therefore, due to privacy risks, users do not want to reveal their true ratings to others. They sometimes supply false data or refuse to provide data at all. It then becomes difficult to generate accurate recommendations on false data or inadequate data. It is vital to offer privacy measures for generating truthful and dependable predictions.

A new hybrid method for CF is proposed in which advantage of both memory- and model-based algorithms are able to be taken. The new scheme is based on trees or graphs in which each node or leaf represents a user and each link represents similarity between two users. To improve performance, some works are done off-line because off-line costs are not critical to overall performance. To create trees or graphs, to represent users and similar users, and the similarities between them, the idea of Markov models is used.

To protect users' privacy while still doing various data mining tasks, different methods have been used. Randomization techniques are employed to hide individual users' data. Users want to disguise their true ratings or preferences

about products. Moreover, they would like to prevent the server or e-commerce sites from learning their rated and/or unrated items. It sometimes might be more damaging to reveal rated and/or unrated products. Randomization techniques allow users to mask both their true ratings and empty cells. They also allow e-commerce sites or CF systems to provide referrals with decent accuracy from masked data. It is still possible to estimate aggregate values from the data perturbed using randomization methods. Since CF is based on aggregate values, accurate predictions are able to be provided still even if data collected for CF purposes is masked.

To evaluate the overall performance of proposed schemes, various experiments are performed on real data sets. Then the outcomes are analyzed of such experiments. Factors that might affect the overall performance of schemes are presented. Finally, schemes are analyzed in terms of privacy, accuracy, and performance.

4.2. Related Work

To increase CF systems' performance and to provide many predictions with decent accuracy to lots of users in a limited time, various approaches have been suggested. Sarwar et al.[53] propose to apply singular value decomposition (SVD) to produce accurate referrals while performing many predictions per second for too many customers and items. Although their scheme achieves higher performance, the model needs to be updated frequently. Goldberg et al. [18] suggest Eigentaste algorithm, which requires constant time to compute referrals, given a database of n users. However, including new users' data is time consuming. Miyahara and Pazzani [39] propose to employ naïve Bayesian classifier (NBC) to produce referrals, which are calculated, based on binary ratings. Ungar and Foster [60] and O'Connor and Herlocker [41] propose to use data partitioning and clustering algorithms to improve performance of CF systems. To overcome information overload problem, Lee [35] proposes a multi-agent system that is capable of obtaining expert knowledge and offering the best items to customers.

Privacy-preserving collaborative filtering (PPCF) has been receiving attention lately. CF schemes with privacy concerns have been proposed in the literature. Canny [11] proposes schemes for PPCF, where users control all of their own private data; a community of users can compute personalized recommendations without disclosing individual users' data. Polat [48] and Polat and Du [46] employ randomized perturbation techniques (RPT) for PPCF. In their schemes, before users send their private data to the server, they perturb their private data. Their solutions make it possible for servers to collect private data from users for CF purposes, without greatly compromising users' privacy requirements. Parameswaran [44] presents a data obfuscation technique that provides robust privacy protection with minimal loss in usability of the data. Data obfuscation is used to modify the value of the data items without distorting the usefulness of the data. He designs and implements a privacy-preserving shared CF framework using data obfuscation algorithm.

4.3. A New Hybrid Algorithm-based Collaborative Filtering

To offer CF services, data from many users is needed. Users give their ratings about many items. The CF system or the server creates a database, which is an *n x m* matrix, where *n* and *m* show the number of users and items, respectively. To determine similarities between users to find similar users, various similarity metrics can be used. Pearson correlation coefficient is commonly used for numerical ratings. it is also used, which is defined, as follows [10]:

$$w_{au} = \sum_{j} \frac{\left(v_{aj} - \mu_a\right)}{\sigma_a} \times \frac{\left(v_{uj} - \mu_u\right)}{\sigma_u}, \tag{4.1}$$

where w_{au} is the similarity weight between users *a* and *u*, σ_a and σ_u represent the standard deviations of users *a*'s and *u*'s ratings, respectively, μ_a and μ_u represent the mean ratings of users *a*'s and *u*'s ratings, respectively, and v_{aj} and v_{uj} represent ratings of users *a* and *u* for item *j*, respectively. The sum is over all items both users have rated. When *a* asks prediction for a target item *q*, she sends her known ratings and a query to the CF system. The system first finds similar users to *a* and uses their data to generate a prediction for *q* by employing a CF algorithm.

After data collection, trees for each user can be constructed off-line. To construct them, the idea of Markov models is used. In the proposed scheme, each user represents a state so that there are *n* states. Each state has *m* objects because there are *m* items. The user selected as the best similar user to *a* represents the initial state or the initial user. Various ways are proposed to determine the initial user. Each user's ratings for items represent observation probabilities in each state. Similarity values or weights between users represent transition probabilities.

4.3.1. Constructing Trees

The steps to construct the trees for each user *u* are, as follows:

a. Similarity weights between user *u* and each other users are calculated using Eq. (4.1). Note that ratings are known and nothing is done to find them.
b. The most *s* similar users to user *u* are found and they are removed from the database.
c. For each of the *s* users, find the best *s* similar users to them among the remaining ones.
d. The most *s* similar users are found iteratively to each user among the remaining ones until there is no remaining one in the database. The graph, constructed for each user, looks like a tree, in which each node's children represent the most similar users. $n = 1 + s + s^2 + s^3 + \ldots + s^y$, and *y* shows the number of levels and its value depends on *n* and *s*.
e. The optimum value of the number of nodes in a tree and the optimum value of *s* can be determined by running real data-based experiments.

4.3.2. Representing Trees

After constructing trees or graphs for each user, each tree is represented as a link list in a database. In such lists, users and the best similar users to them are stored. Each user is linked to the best similar users to her. Besides storing users, ratings of each user for items are also saved. And finally, similarity weights between users are saved. Note that similarities are stored for each link.

4.3.3. Finding Initial States

When *a* asks a prediction or a recommendation list, the first step is to find an initial state or the most similar user to her. Since recommendations will be computed based on the initial user and her link list, selecting the initial user or state is important. The way to select the initial user is vital for performance reasons. Moreover, it will affect the quality of the referrals, as well. The following methods are proposed to select it:

A. Compute similarities between *a* and each user in the database. For performance reasons, this is not desirable, because *n* similarity values must be calculated online.
B. After collecting *n* users' data, they can be clustered into various clusters. For this purpose, various clustering algorithms can be used. However, *k*-means algorithm is widely used to cluster users for CF purposes [37]. *K*-means clustering algorithm is used to cluster users into *k* clusters. When *a* asks referrals, first her cluster is found. To do this, the distances or similarities between *a* and each cluster center are computed. "*a*" is placed into the closest cluster. Since number of clusters is much less than the number of users, this method is more efficient then the first one. After placing *a* in a cluster, the initial user is selected using different methods, as follows:
    a. Select the initial user uniformly randomly among users in that cluster, where this method is called as MI.
    b. Find similarities between *a* and each user in that cluster, and select the best similar user to *a* as initial user, where this method is called as MII.
    c. Select the closest user to the cluster center as initial user, where this method is called as MIII. Note that the closest users can be found off-line.

4.3.4. Computing Recommendations

The steps to generate referrals for *a* can be explained, as follows:

a. *a* sends her known ratings and a query to a CF system. The query contains the target item or items for which referral is sought. The system first places *a* into a cluster.
b. It then selects the initial user for *a* among the users in that cluster.
c. The data in the link list of the initial user is used to find referrals.
d. Since the link list contains *n* users' data, as explained previously, the optimum value of the number of users whose data to be used for CF should be determined. To improve the overall performance, the best-*N* neighbors can be chosen for providing recommendations and the optimum value of *N* can be determined experimentally.
e. Finally, the system considers those *N* users' data to find referrals. It can compute prediction for *a* on item *q* (p_{aq}), as follows [25], which one of the best memory-based CF algorithm, where v_{uq} is the rating of user *u* for *q* and *N* is the number of users involved in recommendation computation:

$$p_{aq} = \mu_a + \sigma_a \times \frac{\sum_{u=1}^{N} w_{au} \times \frac{(v_{uq} - \mu_u)}{\sigma_u}}{\sum_{u=1}^{N} w_{au}} \tag{4.2}$$

f. The system can find predictions for single items, as explained above. Moreover, it can find recommendation lists, as well. To find top-*R* recommendations, *a* asks referrals for her *R* unrated items. The server finds predictions for all *R* items, sorts them decreasingly, and provides the sorted list to *a* as top-*R* recommendations list.

4.4. Producing Recommendations With Privacy on New Hybrid Algorithm

E-commerce sites can trivially offer CF services if privacy is not a concern. However, it happens to be difficult to achieve the same tasks without greatly jeopardizing users' privacy. It is a challenge to give a clear-cut definition for privacy. However, it is abled to be defined, as follows: Users including active users do not want others to learn their ratings or preferences about products they bought or showed interest. Moreover, it might be more damaging to disclose which products users bought or not purchased. Therefore, privacy means, in this

context, preventing data collectors or CF systems from learning true ratings and rated and/or unrated items.

Randomization techniques are proposed to protect users' privacy. Such techniques are used to achieve privacy [4]. To disguise a value x, a simple method is to add a random value r to it. $x + r$, rather than x alone, will appear in the database, where r is a random value drawn from some distribution with mean (μ) being 0. If aggregate data is interested in, certain computations are able to be performed using masked data. Since CF is based on aggregate data, CF services are able to be achieved still on disguised data. Users' ratings are disguised in such a way that the server can only know the range of the data, and such range is broad enough to protect privacy.

4.4.1. Data Perturbation

Users might disguise their data in the same way for consistently masked data. However, since privacy concerns might vary between users, they might decide to perturb their data differently to achieve various levels of privacy. It is still possible to estimate aggregate data from variably masked data [49]. Moreover, inconsistent data perturbation improves privacy. Users including active users mask their data, as follows:

a. They first decide the random number distribution. They can use either uniform or Gaussian distribution to generate random numbers.
b. They decide the level of perturbation or the standard deviation (σ) of random numbers.
c. They then decide the amount of data to be masked. Users might decide to disguise all of the ratings or some of them, all of the empty cells or some of them; or all of the cells of their ratings vectors or some of them.
d. Finally, they generate random numbers using some distribution with $\mu = 0$ and σ**.** Number of random numbers depends on the number of cells to be perturbed. They add those random numbers to corresponding cells to be masked.

After data disguising, users send masked data to the server, which creates a disguised user-item matrix, A'. As shown by Polat and Du [49], uniform and

Gaussian distributions give similar results with respect to accuracy and privacy. With increasing σ, randomness increases; that makes accuracy worse while privacy improves. Users select the cells to disguise based on how much data they want to mask and how many empty cells they want to hide to prevent the server from learning rated and/or unrated items.

4.4.2. Recommendations Computation with Privacy Concerns

As seen from Eq. (4.1) and Eq. (4.2), the server is able to provide CF services using z-score values rather than ratings. The recommendations can be calculated, as follows:

$$p_{aq} = \mu_a + \sigma_a \times \frac{\sum_{u=1}^{N} w_{au} \times z_{uq}}{\sum_{u=1}^{N} w_{au}} = \mu_a + \sigma_a \times P_{aq}, \tag{4.3}$$

where z-scores and w_{au} values based on z-scores can be computed, as follows:

$$z_{uj} = \frac{v_{uj} - \mu_u}{\sigma_u} \text{ and } w_{au} = \sum_{j} z_{aj} \times z_{uj}, \tag{4.4}$$

where j shows the commonly rated items between a and user u, μ_u and σ_u are mean vote and standard deviation of user u's ratings, respectively, v_{uj} is user u's vote for item j, and z_{uj} is user u's z-score for item j.

Users normalize their ratings, mask them, and send the perturbed z-scores to the server. The server first finds a's cluster and initial user. It estimates P'_{aq} from masked data and sends it to a. Since μ_a (mean rating of a's ratings) and σ_a (standard deviation of a's ratings) are known by a, she can estimates p'_{aq} after she receives P'_{aq} from the server. To generate top-R recommendations, either the server or a does not need to de-normalize P'_{aq} values. Since μ_a and σ_a are used for de-normalization and the sorted list is needed other than numerical rating values, the sorted list can be found based on P'_{aq} values.

4.4.3. Estimation from Perturbed Data

The scalar product and sum computations need to be conducted to estimate w_{au} and P_{aq} values. They are based on aggregate data rather than individual data items. As shown by Polat [48], it is still possible to estimate such values with

decent accuracy from masked data even if data is perturbed differently. Let *A* and *B* be the original vectors and given, where $A = (a_1, \ldots, a_n)$ and $B = (b_1, \ldots, b_n)$. *A* is disguised by $F = (f_1, \ldots, f_n)$, and *B* is disguised by $G = (g_1, \ldots, g_n)$, where f_is and g_is are random values drawn from some distribution with μ being 0. Let $A' = A + F$ and $B' = B + G$ be the masked data that are known; the scalar product of *A* and *B* can be estimated from *A'* and *B',* as follows:

$$A' \cdot B' = \sum_{i=1}^{n} \left(a_i b_i + a_i g_i + b_i f_i + f_i g_i\right) = \sum_{i=1}^{n} a_i b_i + \sum_{i=1}^{n} a_i g_i + \sum_{i=1}^{n} b_i f_i + \sum_{i=1}^{n} f_i g_i \quad (4.5)$$

Since random vectors are independent and random values drawn from some distribution with $\mu = 0$, and $E(A \cdot G) = A \cdot E(G) = A \cdot 0 = 0$; similarly $E(B \cdot F) = 0$; and $E(F \cdot G) = E(F) \cdot E(G) = 0 \cdot 0 = 0$, where *E* represents expected value. Thus, it can roughly that $A' \cdot B' \approx \sum_{i=1}^{n} a_i b_i$. The sum of *A* can be estimated from *A'*, as follows*:*

$$\sum_{i=1}^{n} a_i' = \sum_{i=1}^{n} (a_i + f_i) = \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} f_i . \quad (4.6)$$

4.5. Evaluation of the Proposed Schemes' Overall Performance

To evaluate schemes, various experiments are performed using real data sets. As explained before, there are different factors that might affect the overall performance of the proposed schemes. Schemes are analyzed in terms of accuracy and performance based on experiment results. Moreover, they are also analyzed in terms of privacy and additional costs. Although there are different data sets collected for CF purposes, Jester and MovieLens Public (MLP) datasets are used because the results based on them can be generalized. Jester is a web-based joke referral system (http://goldberg.berkely.edu/jester-data/). It has 100 jokes and records of 17,988 users. The ratings are continuous and range from -10 to +10. Almost 50% of the ratings are available. MLP consists of ratings for approximately 1,682 movies made by 943 users. Each user has rated at least 20 movies. Ratings are made on a 5-star scale. It is collected by the GroupLens Research Project (www.cs.umn.edu/research/Grouplens) at the University of Minnesota.

Mean absolute errors (MAE) and normalized mean absolute errors (NMAE) are used as evaluation criteria because they are commonly used ones in the literature [41]. The MAE and the NMAE should be minimized. The lower them, the more accurate results are. If $p_1, p_2, \ldots, p_d$ are predicted values from undisguised data, and $p'_1, p'_2, \ldots, p'_d$ are predicted values from disguised data, then $E = \{\xi_1, \xi_2, \ldots, \xi_d\} = \{p'_1 - p_1, p'_2 - p_2, \ldots, p'_d - p_d\}$ represents errors. Therefore, the MAE and the NMAE can be computed, as follows:

$$MAE = \overline{E} = \frac{\sum_{i=1}^{d} |\xi_i|}{d} \text{ and } NMAE = \frac{MAE}{\xi_{\max} - \xi_{\min}}, \tag{4.7}$$

where ξ_{max} and ξ_{min} represent the maximum and minimum errors, respectively.

Users are selected randomly who rated at least 50 items from Jester and used randomly selected 900 users from MLP to use in experiments. Then they are divided into training and test set. For Jester, 2,000 and 500 users are randomly selected for train and test sets, respectively. For MLP, 700 and 150 users are choosen randomly for training and testing, respectively. For each test user, five rated items are randomly selected as test items. Their ratings are withheld and are tried to predict their values using schemes. After comparing predictions with true withheld ratings, the MAE and the NMAE are computed for all test data and displayed final values.

4.5.1. Experiments for Evaluating New Algorithm

First, trials are performed to asses the overall performance of proposed scheme. As explained before, the number of best neighbors (*N*), the number of similar users to each user (*s*), the number of clusters (*k*), and the methods to select the initial users can affect results. Besides evaluating the schemes in terms of accuracy, they are also evaluated in terms of performance. For this purpose, total time (*T*) is calculated in seconds, necessary to generate referrals. To compare results with conventional memory-based CF algorithms, predictions are computed for the same test and train data using the algorithm proposed by Herlocker et al. [25]. MATLAB R006b program is used and experiments are executed on a computer, which is 1.80 GHz with 1.23 GB RAM. Each experiment set is

executed 10 times and displayed the overall averages. The following experiments are performed to assess how various factors affect the overall performance of schemes:

Experiment 4.5.1.1.: Given 2,000 and 700 train users for Jester and MLP, respectively, experiments are conducted with changing *N* values to show how varying *N* affects the results. *s* at 2 and *k* at 20 are fixed, where MII method is used to select the initial users. 500 and 150 users are used as testing for Jester and MLP, respectively, while predicting ratings for five withheld items for each test user. Overall averages of MAE, NMAE, and *T* are computed; and are displayed in Table 4.1.

Table 4. 1 Overall Performance with Varying *N* Values

	Jester					**MLP**			
N	**100**	**200**	**400**	**800**	**1,600**	**100**	**200**	**400**	**700**
MAE	3.7176	3.7331	3.6740	3.6721	3.6705	0.8612	0.8504	0.8603	0.8684
NMAE	0.2040	0.2033	0.1967	0.1942	0.1939	0.2459	0.2383	0.2362	0.2305
T	5.9282	5.9670	6.1605	6.4893	7.0504	6.7497	6.8020	6.8200	6.9430

As seen from Table 4.1, with increasing *N*, *T* also increases. *T* shows the total time in seconds spent to generate 2,500 and 750 recommendations from Jester and MLP, respectively. Although various *N* values achieve similar MAE and NMAE values, 800 and 200 are chosen as optimum values for Jester and MLP, respectively.

Experiment 4.5.1.2.: Experiments are performed to determine the optimum value of *s*. Other factors are fixed; and varied *s* from 1 to 5 for both data sets. *N* is set at its optimum values determined in the previous experiments and *k* at 20. To select initial users, MII is employed. The same test set is used. After computing overall averages, final outcomes are demonstrated in Table 4.2 for both data sets.

Although the results are very similar for all *s* values, after comparing overall performances in terms of MAEs and NMAEs, 3 and 2 as optimum values of *s* are decided to be selected for Jester and MLP, respectively. In the following experiments, they are used as optimum *s* values. Note that *s* values determine which similar users' data to be used for prediction generation. However, as seen

from Table 4. 2, schemes are robust against various *s* values and are able to provide high quality referrals based on different *s* values.

Table 4. 2 Overall Performance with Varying *s* Values

	Jester				
s	**1**	**2**	**3**	**4**	**5**
MAE	3.6694	3.6721	3.6576	3.6356	3.6637
NMAE	0.1960	0.1942	0.1929	0.1940	0.1943
	MLP				
s	**1**	**2**	**3**	**4**	**5**
MAE	0.8746	0.8504	0.8676	0.8755	0.8645
NMAE	0.2376	0.2383	0.2370	0.2431	0.2308

Experiment 4.5.1.3.: Then trials are executed to determine the best value of *k*. Although there have been experiments run to determine *k* in the literature, it might take different values for different applications. *k* is varied from 10 to 40, where MII is used to decide the initial users. The optimum values of *s* and *N* are used for both data sets. Since the results are similar for both data sets, results are showed in Table 4. 3 for Jester only.

As seen from Table 4.3, *T* improves with increasing *k* values because it takes less time to find the initial user. Number of users in each cluster decreases with increasing *k*. Accuracy becomes better with increasing *k* from 10 to 25, however, it worsens after that. The similar results are obtained for MLP, as well. Therefore, 25 is selected as optimum value of *k* for both data sets.

Table 4. 3 Overall Performance with Varying *k* Values

k	**10**	**15**	**20**	**25**	**30**	**35**	**40**
MAE	3.6697	3.6602	3.6576	3.6207	3.6584	3.6705	3.6819
NMAE	0.1941	0.1931	0.1929	0.1894	0.1956	0.1959	0.1981
T	7.0660	6.8775	6.6216	6.4972	6.4394	6.3595	6.2823

Experiment 4.5.1.4.: After showing affects of *N*, *s*, and *k* values; and determining their optimum values, experiments are performed to show how different initial user selection methods affect results. Trials are executed for all three methods and displayed the final outcomes in Table 4.4, where the optimum values of *N*, *s*, and *k* are employed.

As seen from Table 4.4, *T* takes different values for different initial user selection methods. There are various reasons for this case. The most important

factor determining *T* is the initial user selection method. As seen from Table 4, *T* is the best for MIII for both data sets because it takes the least time to select the initial user. In MIII, the initial user can be determined off-line. *T* is better for MI than for MII because the initial user in MI is randomly selected. However, to determine the initial user using MII, the similarities between *a* and each user in *a*'s cluster are found. Although *T* is the worst for MII, it achieves the best accuracy compared to other methods. Therefore, MII is selected as the best method to decide the initial users.

Table 4. 4 Overall Performances with Various Initial User Selection Methods

	Jester			**MLP**		
	MI	**MII**	**MIII**	**MI**	**MII**	**MIII**
MAE	3.7662	3.6207	3.7277	0.8816	0.8472	0.8695
NMAE	0.1950	0.1894	0.2064	0.2506	0.2376	0.2427
T	5.9046	6.4972	5.8671	6.7623	6.7841	6.7612

After determining the optimum values for *N*, *s*, and *k*; and deciding the best initial user selection method, experiments are performed to compare proposed algorithm with the conventional algorithm (called Base Algorithm) proposed by Herlocker et al. [25]. The same train and the test sets for both data sets are used for both algorithms. Experiments are performed and displayed the overall average outcomes in Table 4. 5.

Table 4. 5. Proposed Algorithm (A1) vs. Base Algorithm (A2)

	Jester		**MLP**	
	A1	**A2**	**A1**	**A2**
MAE	3.6207	3.4975	0.8472	0.8458
NMAE	0.1894	0.2075	0.2376	0.2450
T	6.4972	15.234	6.7841	34.901

As seen from Table 4.5, proposed scheme is much better than the conventional algorithm in terms of *T*. Since scheme is a hybrid- memory and model-based, algorithm, some computations are conducted off-line. Algorithm is 2.34 and 5.14 times faster than the A2 for Jester and MLP, respectively. For MLP, scheme is almost as good as the A2 in terms of MAE and NMAE. Although scheme has worse performance than the A2 in terms of accuracy, scheme is still able to generate high quality recommendations. Online time required to offer

many recommendations to many customers is vital for the success of both the CF systems and e-commerce sites. Therefore, using proposed scheme provides competitive edge to those sites using it over the ones using the conventional algorithms.

4.5.2. Experiments for Evaluating New Algorithm with Privacy

Privacy protection measures affect the quality of predictions. Due to random numbers added to true data, accuracy is expected to become worse. To show how privacy-preserving measures affect accuracy, experiments are performed. In these trials, the optimum values of *N*, *s*, and *k* are used; and employed the best method to select initial users. There are various privacy factors that might affect accuracy. Several experiments have been conducted by Polat [48] to show their effects on accuracy. With increasing available data or number of best users (*N*), the quality of referrals computed with privacy concerns slightly improves. Since randomness increases with increasing level of perturbation, accuracy lessens with increasing σ. Using uniform or Gaussian distributions to generate random numbers gives similar results. The effects of number of cells to be masked have been also investigated. It is wanted to be shown how much accuracy goes down due to privacy concerns while generating predictions using scheme, rather than showing how various privacy factors affects accuracy. Thus, the following trials are executed:

Experiment 4.5.2.: Trials are performed using both data sets. For data disguising, random numbers are generated using Gaussian distribution with μ being 0 and σ being 1 and 2. Random numbers are added to randomly selected 50% and 25% of the cells in each user's ratings vector for Jester and MLP, respectively. Since different random numbers are generated each time, data disguising is executed 100 times. After calculating overall averages,are displayed in Table 4.6.

Table 4. 6. Overall Performance of Privacy-Preserving Schemes

	Jester			**MLP**		
	Original	**Perturbed Data**		**Original**	**Perturbed Data**	
	Data	**σ = 1**	**σ = 2**	**Data**	**σ = 1**	**σ = 2**
MAE	3.6207	3.7164	4.2133	0.8472	0.9033	0.9924
NMAE	0.1894	0.1976	0.2144	0.2376	0.2379	0.2509

As seen from Table 4.6, accuracy becomes worse for both data sets when data is disguised. However, results on perturbed data are still promising compared to the results on original data. It is still possible to generate recommendations with decent accuracy using scheme without jeopardizing users' privacy. As expected, the results worsen with increasing σ values due to escalating randomness. When σ is 1, the accuracy losses due to privacy measures are 2.6% and 6.6% for Jester and MLP, respectively.

How privacy affects the quality of recommendations are showed by running trials. Privacy protection measures introduce additional costs. Such costs might be extra storage, communication, or computation costs. Due to privacy concerns, schemes do not cause any supplementary storage costs. Since off-line costs are not critical, online extra communication and computation costs are investigated. The users are able to normalize their ratings and mask them off-line based on their privacy concerns. Since users might insert bogus data into their empty cells, more data is involved in the scalar product and the sum computations. However, the server is able to provide predictions from masked data without conducting extra work to get rid of the effects of random data. Therefore, it can be said that the online additional computation costs are negligible. Although privacy concerns do not increase the number of communications required to generate referrals online, the amount of data to be sent off-line increases because random numbers are added to both ratings and empty cells. In conclusion, supplementary costs due to privacy concerns are negligible.

Proposed schemes should protect users' including active users' privacy. The server wants to learn the true ratings and rated items. However, since users disguise their z-scores instead of true ratings, it becomes difficult for the server deriving ratings from masked z-scores, because it does not know the mean ratings and the standard deviations of ratings. The server can guess the distribution of random numbers with probability ½ for one user. It also needs to guess the standard deviations of random numbers because users select them uniformly randomly over a range let say $[0, \sigma_r]$. The server can guess them in an interval. If

the interval length L is called, then the probability of guessing the correct interval for the server is 1 out of L/σ_r for one user. After guessing the distributions and the standard deviations of random numbers, it should guess the disguised number of cells and the masked cells. Note that there are m items. Suppose that a user u has rated m_{ur} items and the remaining $m_{ue} = m - m_{ur}$ items are not rated. Assume that she decides to hide $x_u\%$ of her empty cells besides masking all of her ratings and the number of empty cells in the masked data vector is m'_u. For one user, the probability of guessing the correct x_u is 1 out of $(98 - (m'_u/m) * 100)$, assuming that each user has rated at least two items. After guessing x_u, the server can find out m_{ue}. Then the probability of guessing the masked empty cells for one user is 1 out of $C^{m-m'_u}_{m_{ue}-m'_u}$, where C^f_g represents the number of ways picking g unordered outcomes from f possibilities. The users can decide the parameters of data perturbation on how much accuracy and privacy they want. The privacy level introduced by random numbers can be calculated using the formula given by Agrawal and Aggarwal [3]. They propose a privacy measure, which takes into account the distribution of original data, based on the differential entropy of a random variable. Uniform and Gaussian random number distributions produce similar privacy levels. For Gaussian distribution with σ being 1, the privacy level is 2.9222, while it is 3.6962 when σ being 2. For a general random variable X, the privacy level denotes the length of the interval, over which a uniformly distributed random variable has the same uncertainty as X.

4.6. Conclusions

A new algorithm is presented to generate accurate recommendations efficiently. New algorithm is a hybrid- memory and model-based, algorithm. It takes advantage of both memory and model-based CF algorithms. As seen from real data-based trials results, the time required to offer many referrals using the proposed scheme is much less than the one using the conventional algorithm. It is imperative to provide many predictions to many customers in a limited time. New algorithm helps e-commerce sites achieve such goal. In terms of accuracy, method is almost as good as the conventional ones. By sacrificing little on accuracy, e-commerce sites are able to improve T if they use scheme. The

proposed scheme introduces extra storage cost because trees are created for each user and store them to improve the online time. Since time rather than storage cost is the major concern, time is improved by introducing extra storage cost. How to achieve predictions on algorithm while preserving users' privacy is also investigated. It is shown that it is still possible to generate accurate referrals on scheme without jeopardizing users' privacy. Scheme does not introduce additional costs due to privacy concerns. In conclusion, e-commerce sites can use proposed scheme to offer accurate referrals efficiently while preserving users' privacy in order to increase their sales and profits. It is believed that customers will prefer those sites providing accurate predictions efficiently without violating their privacy.

5. CONCLUSIONS AND FUTURE WORK

In this thesis, firstly, it is shown that it is possible to provide predictions based on horizontally or vertically distributed HMMs between two parties without violating their privacy. In addition to preserving privacy, providing predictions with decent accuracy efficiently is also imperative for the success of HMMs. Therefore, the proposed schemes are analyzed in terms of accuracy and performance. Fortunately, distributed HMMs-based schemes with privacy accomplish the same accuracy as the ones without privacy concerns. On the other hand, performance decreases with privacy concerns because privacy, accuracy, and efficiency are inversely proportional. Although horizontally or vertically distributed HMMs are investigated, such model partition might be hybrid. It is more likely that the proposed schemes can be modified in such a way to provide predictions on hybrid distributed HMMs with privacy. There still remains work to be done to study hybrid distributed HMMs-based forecasting with privacy. Although two-party schemes are scrutinized only, the model might be partitioned among more than two parties. The proposed schemes can be expanded to multi-party schemes. In that case, communication bottlenecks will become a major issue. How the proposed schemes can be expanded to multi-party methods should be deeply explored.

Secondly, distributed HMMs-based prediction with privacy is presented. Only horizontal or vertical partition are considered. The partition might be hybrid. How to provide similar HMM-based services will be studied when the model is hybrid distributed between two parties without jeopardizing the model owners' privacy. Although it is assumed that the model is partitioned between two parties, it might be distributed between more than two parties. Proposed solutions can be extended to multi-party schemes. However, detail analysis should be done and solutions should be proposed to overcome the bottlenecks that might occur in multi-party schemes.

Finally, a new algorithm is presented to generate accurate recommendations efficiently. The algorithm is a hybrid (memory- and model-based) algorithm. It takes advantage of both memory- and model-based CF algorithms. As seen from real data-based trials results, the time required to offer

many referrals using the proposed scheme is much less than the one using the conventional algorithm. It is imperative to provide many predictions to many customers in a limited time. In terms of accuracy, the new method is almost as good as the conventional one. The proposed scheme introduces extra storage costs because it needs to create trees for each user and stores them to improve online time. Since time rather than storage cost is the major concern, time is improved by introducing extra storage cost. How to achieve predictions based on the new algorithm while preserving users' privacy is also investigated. It is shown that it is still possible to generate accurate referrals using the new scheme without jeopardizing users' privacy. Whether further improvements can be done to the proposed scheme or not should be deeply investigated. Moreover, how to improve accuracy while conducting different memory-based algorithms to calculate predictions after finding the model should also be investigated.

REFERENCES

[1] Abrash, V., Franco, H., Cohen, M., Morgan, N. and Konig, Y., "Connectionist gender adaptation in a hybrid neural network/hidden Markov model speech recognition system", Proceedings of the International Conference on Spoken Language Processing, 911-914, Banff, Canada, 1992.

[2] Abrash, V., Cohen, M., Franco, H. and Arima, I., "Incorporating linguistic features in a hybrid HMM/MLP speech recognizer," Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, **2(II)**, 673-676, Adelaide, Australia, 1994.

[3] Agrawal, D. and Aggarwal, C. C., "On the design and quantification of privacy-preserving data mining algorithms," Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART, 247-255, Santa Barbara, CA, USA, 2001.

[4] Agrawal, R. and Srikant, R., "Privacy-preserving data mining," Proceedings of the ACM SIGMOD International Conference on Management of Data, 439-450, Dallas, Texas,USA, 2000.

[5] Ahn, H., Kim. K., and Han I., "Hybrid genetic algorithms and case-based reasoning systems for customer classification," Expert Systems, **23 (3)**, 127-144, 2006.

[6] Beausang, J. F., Zurla, C., Manzo, C., Dunlap, D., Finzi, L. and Nelson, P., "DNA looping kinetics analyzed using diffusive hidden Markov model," Biophysical Journal, **92 (8),** L64-L66, 2007.

[7] Begletier, R., El-Yaniv. R., and Yona, G., "On prediction using variable order Markov models," Journal of Artificial Intelligence Research, **22:**385-421, 2004.

[8] Benaloh, J., "Dense probabilistic encryption," Proceedings of the Workshop on Selected Areas of Cryptography, 120-128, Kingston, Ontario, Canada, 1994.

[9] Birney, E., "Hidden Markov models in biological sequence analysis," IBM Journal of Research and Development in Technology, **45 (3/4),** 449-454, 2001.

[10] Breese, J. S., Heckerman, D., and Kadie, K., "Empirical analysis of predictive algorithms for collaborative filtering," Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, 43-52, Madison, Wisconsin, USA, 1998.

[11] Canny, J., "Collaborative filtering with privacy," Proceedings of the IEEE Symposium on Security and Privacy, 45-57, Oakland, California, USA, 2002.

[12] Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., and Zhu, M., "Tools for privacy-preserving distributed data mining," SIGKDD Explorations, **4(2)**, 28-34, 2003.

[13] Cranor, L.F., Reagle, J., and Ackerman, M.S., "Beyond concern: Understanding net users' attitudes about online privacy," Available from, http://www.research.att.com/resources/trs/TRs/99/99.4/99.4.3/report.htm.

[14] Cranor, L. F., "'I didn't buy it for myself' privacy and e-commerce personalization," Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society, 111-117, Washington, DC, USA, 2003.

[15] Dugad R. and Desai, U. B., "A tutorial on hidden Markov models," Technical Report, SPANN-96.1, Indian Institute of Technology-Bombay, 1996.

[16] Fox, M., Ghallab, M., Infantes G. and Long, D., "Robot introspection through learned hidden Markov models," Artificial Intelligence, **170 (2)**, 59–113, 2006.

[17] Franco, H., Weintraub, M. and Cohen, M., "Context modeling in a hybrid HMM-Neural net speech recognition system," Proceedings of the International Conference on Acoustics, Speech and Signal Processing, 2089-2092, Houston, USA, 1997.

[18] Goldberg, K., Roeder, T., Gupta, D., and Perkins, C., "Eigentaste: A constant time collaborative filtering algorithm," Information Retrieval, **4 (2)**, 133-151, 2001.

[19] Goldberg, D., Nichols, D., and Oki, B. M., "Using collaborative filtering to weave an Information Tapestry," Communications of ACM, **35 (12)**, 61-70, 1992.

[20] González, A. M., Roque, A. M. S., and García- González, J., "Modeling and forecasting electricity prices with input/output hidden Markov models," IEEE Transactions on Power Systems, **20 (1)**, 13-24, 2005.

[21] Green, P.J., Noad, R. and Smart, N.P., "Further hidden Markov model cryptanalysis," Lecture Notes in Computer Science, **3659**, 61-74, 2005.

[22] Hamilton, J.D., "Regime-switching models," Technical Report, University of California, San Diego, 2005.

[23] Hassan, R. and Nath, B., "Stock market forecasting using hidden Markov model: A new approach," Proceedings of the 5th International Conference on Intelligent Systems Design and Applications**,** 192-196**,** Wroclaw, Poland, 2005.

[24] Henderson, J., Salzberg, S., and Fasman, K.,"Finding genes in human DNA with a hidden Markov model," Journal of Computational Biology, **4 (2),** 127-141, 1996.

[25] Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. T., "An algorithmic framework for collaborative filtering," Proceedings of the 22nd Annual International ACM SIGIR Conference, 230-237, Berkeley, California, USA, 1999.

[26] Husmeier, D. and McGuire, G., "Detecting recombination in 4-Taxa DNA sequence alignments with Bayesian hidden Markov models and Markov chain Monte Carlo," Molecular Biology and Evolution, **20 (3)**, 315-337, 2003.

[27] Jagannathan, G, and Wright, R. N., "Privacy-preserving distributed *k*-means clustering over arbitrarily partitioned data," Proceeding of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, 593-599, Chicago, USA, 2005.

[28] Jiang, H., Li, X., and Liu, C., "Large margin hidden Markov models for speech recognition, audio, speech, and language processing," IEEE Transaction on Audio, Speech, and Language Processing**, 14 (5)**, 1584-1595, 2006.

[29] Kaleli, C. and Polat, H., "Providing naïve Bayesian classifier-based private recommendations on partitioned data," Lecture Notes in Artificial Intelligence, **4702**, 515-522, 2007.

[30] Kantarcioglu, M. and Clifton, C., "Privacy-preserving distributed mining of association rules on horizontally partitioned data," IEEE Transactions on Knowledge and Data Engineering, **16 (9)**, 1026-1037, 2004.

[31] Kantarcioglu, M. and Clifton, C., "Privately computing a distributed k-nn classifier," In Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'04), Pisa, Italy, September 20-24, 2004.

[32] Kantarcioglu, M. and Vaidya, J. S., "Privacy-preserving naive Bayes classifier for horizontally partitioned data," In Proceedings of the IEEE ICDM Workshop on Privacy-Preserving Data Mining, 3-9, Melbourne, FL, USA, November 19- 22, 2003.

[33] Karlof, C. and Wagner, D., "Hidden Markov model cryptanalysis," Lecture Notes in Computer Science, **2779**, 17-34, 2003.

[34] Laxman S., Sastry, P. S., and Unnikrishnan, K. P., "Discovering frequent episodes and learning hidden Markov models: A formal connection," IEEE Transactions on Knowledge and Data Engineering, **17 (11)**, 1505-1517, 2005.

[35] Lee, W., "Applying domain knowledge and social information to product analysis and recommendations: An agent-based decision support system," Expert Systems, **21 (3)**, 138-148, 2004.

[36] Lin, W., Orgun, M.A., and Williams, G. J., "Multilevels hidden Markov models for temporal data mining," Proceedings of the KDD Workshop on Temporal Data Mining, San Francisco, CA, USA, 2001.

[37] Marlin, B., "Collaborative filtering: A machine learning perspective," M.Sc. Thesis, University of Toronto, 1994.

[38] Merugu, S. and Ghosh, J., "Privacy-preserving distributed clustering using generative models," Proceedings of the 3rd IEEE International Conference on Data Mining, 211-218, Melbourne, Florida, USA, 2003.

[39] Miyahara, K. and Pazzani, M.J., "Improvement of collaborative filtering with the simple Bayesian classifier," IPSJ Journal, **43 (11)**, 3429-3437, 2002.

[40] Naccache, D. and Stern, J., "A new public key cryptosystem based on higher residues," Proceedings of the 5th ACM Conference on Computer and Communication Security, 59-66, San Francisco, CA, USA, 1998.

[41] O'Connor, M. and Herlocker, J. L., "Clustering items for collaborative filtering," Proceedings of the ACM SIGIR'99 Workshop on Recommender Systems, 1999.

[42] Oliveira, S. R. M. and Zaiane, O. R., "Achieving privacy preservation when sharing data for clustering," In Proceedings of the International Workshop on Secure Data Management in a Connected World (SDM'04) in conjunction with VLDB 2004, Toronto, Canada, August 2004.

[43] Paillier, P., "Public-key cryptosystems based on composite degree residue classes," Lecture Notes in Computer Science, **1592**, 223-238, 1999.

[44] Parameswaran, R., "A robust data obfuscation approach for privacy-preserving collaborative filtering," PhD Thesis, Georgia Institute of Technology, 2006.

[45] Pennock, D. M., Horvitz, E., Lawrence, S., and Giles, C. L., "Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach". Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, 473-480, Stanford, CA, USA, 2000.

[46] Polat, H. and Du, W., "Privacy-preserving collaborative filtering on vertically partitioned data," Lecture Notes in Computer Science, **3721**, 651-658, 2005.

[47] Polat, H. and Du, W., "Privacy-preserving top-*N* recommendation on horizontally partitioned data," Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence, 725-731, Paris, France, 2005.

[48] Polat, H., "Privacy-preserving collaborative filtering," Ph.D. Thesis. Syracuse University, 2006.

[49] Polat, H. and Du, W., "Effects of inconsistently masked data using RPT on CF with privacy," Proceedings of the 22nd ACM SAC E-commerce Technologies, 649-653, Seoul, South Korea, 2007.

[50] Polat, H. and Du, W., "Privacy-preserving top-N recommendation on partitioned data," Journal of the American Society for Information Science and Technology, **59**(7):1093-1108, 2008.

[51] Rabiner, L. R., "A tutorial on hidden Markov models and selected applications in speech recognition," Proceedings of the IEEE, **77 (2)**, 257-286, 1989.

[52] Sanil, A. P., Karr, A. F., Lin, X. and Peiter, J. P., "Privacy-preserving regression modeling via distributed computation," In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 677-682, Seattle, WA, USA, August 22-25, 2004.

[53] Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. T., "Application of dimensionality reduction in recommender system: A case study," Proceedings of the ACM WebKDD 2000 Web Mining for E-commerce Workshop, 682-693, Boston, MA, USA, 2000.

[54] Savage, J., Marquez, E., Lepe-Casillas, F., and Morales, M. A., "Hidden Markov models and vector quantization for mobile robot localization," Robotics and Applications, IASTED, 498-032, Boston, MA, USA, 2005.

[55] Schafer, J. B., Konstan, J. A., and Riedl, J. T., "Recommender systems in e-commerce," Proceedings of the 1st ACM Conference on Electronic Commerce, 158-166, Denver, CO, USA, 1999.

[56] Shatkay, H., "Learning geometrically-constrained hidden Markov models for robot navigation: Bridging the topological-geometrical gap," Jounal of AI Research , **16,** 167–207, 2002.

[57] Skounakis, M., Craven, M. and Ray, S., "Hierarchical hidden Markov models for information extraction," Proceedings of the 18th International Joint Conference on Artificial Intelligence, Acapulco, Mexico, 2003.

[58] Thayer, K.M. and Beveridge, D. L., "Hidden Markov models from molecular dynamics simulations on DNA," PNAS of the United States of America, **99 (13)**, 8642-8647, 2002.

[59] Tso, B. and Chang, P. Y., "Mining free-structured information based on hidden Markov models," Expert Systems with Applications, **32 (1)**, 97-102, 2007.

[60] Ungar, L. H. and Foster, D. P., "Clustering methods for collaborative filtering," Proceedings of the Workshop on Recommendation Systems at the 15th National Conference on Artificial Intelligence, Menlo Park, CA, USA, 1998.

[61] Vaidya, J. S. and Clifton, C., "Privacy-preserving naïve Bayes classifier for vertically partitioned data," Proceedings of the 2004 SIAM Conference on Data Mining, 522-526, Lake Buena Vista, FL, USA, 2004.

[62] Vaidya, J. S., "Privacy-preserving data mining over vertically partitioned data," PhD thesis, Purdue University, West Lafayette, IN, USA, 2004.

[63] Verykios, V. S., Bertino, E., Fovino, I. N., Provenza, L. P., Saygin, Y., and Theodoridis, Y., "State-of-the-art in privacy-preserving data mining," SIGMOD Record, **3 (1)**, 50-57, 2004.

[64] Vlasenko, B. and Wendemuth, A., "Tuning hidden Markov model for speech emotion recognition," Proceedings of the 33rd German Annual Conference on Acoustics, Stuttgart, Germany, 2007.

[65] Wright, R. and Yang, Z., "Privacy-preserving Bayesian network structure computation on distributed heterogeneous data," In Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 703-718, Seattle, WA, USA, August 22-25, 2004.

[66] Yau, W. C., Kumar, D. K. and Weghorn, H., "Visual speech recognition using motion features and hidden Markov models," Lecture Notes in Computer Science, **4673,** 832-839, 2007.

[67] Zhang, Y., "Prediction of financial time series with hidden Markov models," M.Sc. Thesis, Simon Fraser University, 2004.

[68] Zijian, Y., "Estimation of Markov regime-switching model," Technical Report, CCFEA Project, 2004.