Demo of P2Prec: a Social-based P2P Recommendation System

Fady Draidi, Esther Pacitti, Didier Parigot, Guillaume Verger
INRIA & LIRMM, Montpellier, France
{*Fady.Draidi@lirmm.fr, Esther.Pacitti@lirmm.fr, Didier.Parigot@inria.fr, Guillaume.Verger@inria.fr*}

Résumé. P2Prec est un système de recommandation P2P pour le partage de contenu à grande échelle, qui exploite à la fois les contenus et les aspects sociaux. L'idée est de recommander des documents de qualité portant sur des sujets demandés dans des requêtes (par mots-clés) et des contenus d'amis (ou collègues) experts des sujets. Nous avons réalisé un prototype de P2Prec avec SON, une plate-forme open source de développements de réseaux P2P. Dans cet article, nous décrivons la démonstration des services principaux de P2Prec (installation et initialisation de pairs P2Prec, dissémination (par gossip) des sujets d'intérêt entre amis, recherche par mots-clés) avec notre prototype.

1. Introduction

The general problem we address is large-scale content sharing for on-line communities. Consider, for instance, a scientific community (e.g., in bio-informatics, physics or environmental science) where community members are willing to share large amounts of documents (including images, experimental data, etc) stored in their local servers. Assume also that they don't want to lose control over their data at a central site. A promising solution is to organize community members in a peer-to-peer (P2P) overlay network, with the advantages of decentralized control, peer autonomy and scalability.

Locating contents based on contents ids in a P2P overlay network is now well-solved (see e.g. [2]). However, the problem with current P2P content-sharing systems is that the users themselves, i.e., their interest or expertise in specific topics, or their rankings of documents they have read, are simply ignored. In other words, what is missing is a recommendation service that, given a query, can recommend relevant documents by exploiting user information.

In [1], we proposed P2Prec, a social-based P2P recommendation system for large-scale content sharing. P2Prec leverages content-based and social-based recommendation over a P2P overlay network. The main idea is to recommend high quality documents related to query topics and contents hold by friends (or friends of friends), who are expert on the topics related to the query. Our recommendation model relies on a distributed graph, where each node represents a user (peer) labelled with the contents it stores and its topics of interests. Expertise is implicitly deduced based on the contents stored by a user. The topics each peer is interested in are automatically calculated by analyzing the documents the peer holds. Peers become relevant for a topic if they hold a certain number of highly rated documents on this topic. To exploit friendship links, we rely on Friend-Of-A-Friend (FOAF) descriptions (http://www.foaf-project.org). To disseminate information about experts, we propose new semantic-based gossip algorithms that provide scalability, robustness, simplicity and load balancing. In addition, we propose an efficient query routing algorithm that selects the best peers to recommend documents based on the gossip-

view entries and query topics. At the query's initiator, recommendations are selectively chosen based on similarity, rates and popularity or other recommendation criteria.

We have implemented a prototype of P2Prec using Data-Shared Overlay Network (SON) (http://www-sop.inria.fr/team/zenith/SON), an open source development platform for P2P networks using web services, JXTA and OSGi (http://www.osgi.org). SON components communicate by asynchronous message passing to provide weak coupling between system entities. To scale up and ease deployment, we rely on a Distributed Hash Table (DHT) for publishing and discovering services or data.

In this paper, we describe the demo of P2Prec's main services (installing and initializing P2Prec peers, gossiping topics of interest among friends, key-word querying for contents) using our prototype implemented as an application of SON.

2. Overview of P2Prec

P2Prec's recommendation model is expressed based on a graph $G = (D,U,E,T)$, where D is the set of shared documents, U is the set of users $u_1,...u_n$ corresponding to autonomous peers $p_1,...p_n$, E is the set of edges between the users such that there is an edge $e(u,v)$ if users u and v are friends, and T is the domain of topics. Each user $u \in U$ is associated with a set of topics of interest $T_u \subset T$, and a set of relevant topics $T_u^r \subset T_u$ extracted locally from the documents u has rated.

In our approach, we use Latent Dirichlet Allocation (LDA) to automatically model the topics in the system, which in turn are used to extract users' relevant topics of interest. LDA processing is done in two steps: training at a global level and inference at the local level. The global level is given to a bootstrap server (BS), where BS aggregates a sample set of M documents from P2Prec participant peers. Then BS executes the LDA classifier program to get a set $T = \{t_1,..t_k\}$ of topics, where k is the number of topics. Each topic $t \in T$ contains a set of z words, where z is the total number of the unique words in M, and each of these words is associated with a weight value between 0 and 1. At the local level, user u performs LDA locally to extract the topics of its local documents, using the same set of topics T that were previously generated at the global level.

Each user $u \in U$ maintains a FOAF file which contains a description of its personal information, and friendship network. Personal information includes the extracted topics of interest, where each topic of interest $t \in T_u$. Friends information includes friends' name, links (URI) to their FOAF files, relevant (topics of interest), and trust levels. The trust level between user u and a friend v, denoted by $trust(u,v)$, is a real value within [0, 1], and it represents how much user u has faith in its friend v.

To disseminate recommendation, we rely on gossip protocols [4] as follows. Each user keeps locally (in its FOAF file) its relevant friends and their corresponding topics of interests. At each gossip exchange, each user u checks its gossip *local-view* to enquire whether there is any relevant

user *v* that is similar to *u*, with respect to their topics of interests and friendship networks. If it is the case, a demand of friendship is launched among *u* and *v* and the respective FOAF files are updated accordingly. FOAF files are used to support users' queries. Whenever a user submits a key-word query, the FOAF file is used as a directory to redirect the query to the top-k most adequate friends by taking into account similarities, relevance, usefulness and trust.

Furthermore, each user *u* establishes new friendships with users that are *useful* to *u*'s demands, and if their friendship networks have high overlap with *u*'s friendship network. A user *v* is considered *useful* to a user *u*, if *v* is a relevant user and a certain amount of *v*'s relevant topics T_v^r are of interest for *u*. User *u* exploits its useful friends (of friends) for recommendations.

Finally, a key-word query *q* is associated with a TTL (Time To Live) and is routed recursively in a P2P top-k manner: once a query is submitted by *u*, it is forwarded to *u*'s top-k friends. When a query is received at any peer, it is again redirected to its top-k friends, until TTL is reached. Each user *v* that received the query provides recommendations to *u*. The response to a query *q* is a recommendation that has been provided in a ranked list, based on a function that ranks each document according to its relevance with *q*, its popularity, the similarity and trust between *q*'s initiator and responder *v*.

3. P2Prec Implementation

With SON, the development of a P2P application is done through the design and implementation of a set of components. Each component includes a non-functional code that provides the component services and a code component that provides the component logic (business code). The complex aspects of asynchronous distributed programming (non-functional code) are separated from code components and automatically generated. From a description of a component's services, the Component Generator (CG) automatically generates the non-functional code. Thus, the programmer does not deal with complex distributed programming aspects. The basic infrastructure of SON is composed of a Component Manager (CM), a Publishing and Discovery Component (PDC), and a Connection Component (CC). The PDC allows publishing or discovering components on different peers using a DHT. The CC provides connection between remote components on peers. The CM performs the creation of new component instances and the connections between them. To establish a connection between two components, the CM uses the services description of each component. The CM delegates the management of lists of remote components to the PDC. SON is implemented in Java on top of OSGi components that provide all basic services for the lifecycle of our components, in particular, the deployment services. The launching of a SON application is defined through an OSGi configuration, which describes the application components.

We developed P2Prec as a SON application with two components: the LDA component for the documents topics process and the P2Prec component for the recommendation process. For instance, the services of the P2Prec component are the services for passive and active propagation

through gossip services (*gossip* and *gossipAnswer* services) and the queries services (*query* and *queryAnswer* services). There are two OSGi configurations, the Bootstrap Server (BS) configuration and the Client (the peer) configuration, as shown in Figure 1.

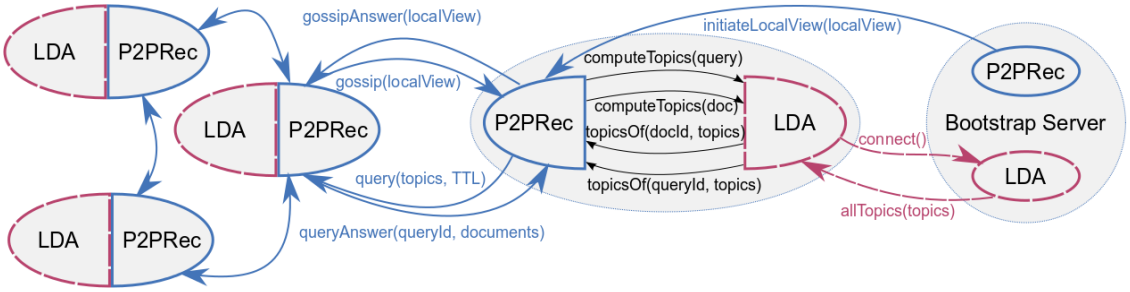

Figure 1: P2Prec Architecture

4. P2Prec Demonstration

In this section we describe how the P2Prec services cooperate using scenarios based on the Ohsumed documents corpus [3] that has been widely used in IR. It is a set of 348566 references from MEDLINE, the on-line medical information database, consisting of titles or abstracts from 270 medical journals over a five year period (1987-1991). We show how the application works, from the global installation to the utilization by an end-user.

Installation. In order to run a P2Prec peer properly, any user (at a peer) has to connect first to the Bootstrap Server (BS). Therefore we define a place the BS will run on. Every peer in the system will know its IP address. As the BS and any peer offer the same kind of services, we have defined two OSGi configurations for running P2Prec components: one as a BS, and one as a standard peer that will connect to the BS.

Initialization. Each peer consists of a LDA part coupled with a Communication part (called P2Prec). As the demonstration starts, the BS is created, and so are several peers (30 of them). Each peer sends some of its documents, which are arbitrarily distributed among all peers, to the BS to perform LDA on a sample of all documents and to define the set of topics used in the network. Next, the BS informs all connected peers about the topics that are present in the network, and each peer indexes its own documents with the set of topics. Each peer is given an initial FOAF, which determines its friends in the network, and provides it information about them. It can now start gossiping with other peers, and the user belonging to the peer can send queries to discover documents.

Gossiping. The gossip service is at the heart of P2Prec, and is transparent to the end-user. While peers exchange gossiping messages, the system recommends new friendships to users. For the sake of the demonstration, we developed an interface showing what is internally happening during gossiping (see Figure 2). The interface shows the current friends of the user, the gossiping messages sent and received by the peer, the gossip local-view that permits to find friends, etc.

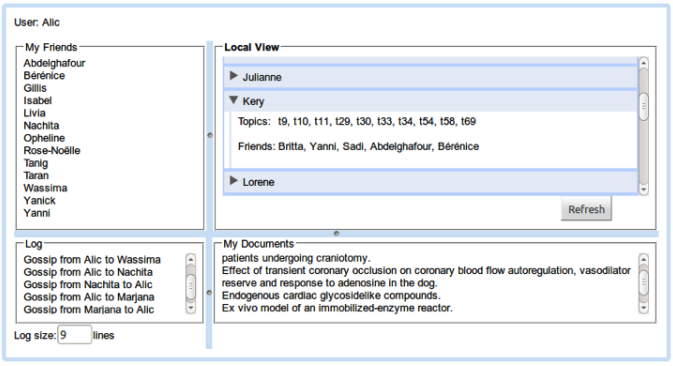

Figure 2: P2Prec Gossip Interface

Querying. Spreading information with gossip to make new friends has one aim: being able to answer queries accurately when a user searches for documents. This is where the query service is needed. The user is able to send a query for getting documents recommendations from her friends. The local LDA of the user translates the query into a set of relevant topics, and the peer sends them through the query service to the user's friends. Each friend may recommend documents depending on the similarity in terms of topics and the rate of the document. The query hops to friends of friends as many times as its TTL allows, the results being returned during the journey. Figure 3 shows the result returned to the user after a query is sent.

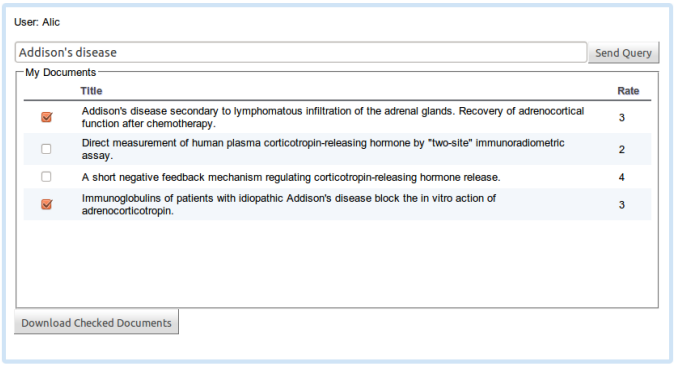

Figure 3: P2Prec Query Interface

5. References

1 Draidi F., Pacitti E., Kemme B. P2Prec: a P2P Recommendation System for Large-scale Data Sharing. Journal of Transactions on Large-Scale Data and Knowledge-Centered Systems (TLDKS), Springer, LNCS 6790(3), to appear, 2011.

2 El Dick M., Pacitti E., Akbarinia R., Kemme, B. Building a peer-to-peer content distribution network with high performance, scalability and robustness. Information Systems. 36(2): 222-247 (2011).

3 Hersh W.R., Buckley C., Leone T., Hickam D.H., Ohsumed: An interactive retrieval evaluation and new large test collection for research. ACM SIGIR, 192-201,1994.

4 Jelasity M., Voulgaris S., Guerraoui R., Kermarrec A.M., VanSteen M. Gossip-based peer sampling. ACM Trans. On Computer Systems, 25(3), 2007.