Rankbox: An Adaptive Ranking System for Mining Complex Semantic Relationships Using User Feedback

Na Chen*
Computer Science Dept.
University of Southern California
Los Angeles, CA, USA
nchen@usc.edu

Viktor K. Prasanna
Ming Hsieh Dept. of Electrical Engineering
University of Southern California
Los Angeles, CA, USA
prasanna@usc.edu

Abstract

This paper presents Rankbox, an adaptive ranking system for mining complex relationships on the Semantic Web. Our objective is to provide an effective ranking method for complex relationship mining, which can 1) automatically personalize ranking results according to user preferences, 2) be continuously improved to more precisely capture user preferences, and 3) hide as many technical details from end users as possible. We observe that a user's opinions on search results carry important information regarding his interests and search intentions. Based on this observation, our system supports each user to give simple feedback about the current search results, and employs a machine-learning based ranking algorithm to learn the user's preferences from his feedback. A personalized ranking function is then generated and used to sort results of the user's subsequent queries. The user can keep teaching the system his preferences by giving feedback through several iterations until he is satisfied with the search results. Our system is implemented and deployed on a web server that can be easily accessed through web browsers. We evaluate our system on a large RDF knowledge base created from the Freebase linked-open-data. The experimental results demonstrate the effectiveness of our method compared to the state-of-the-art.

1. Introduction

Semantic Web [7] technologies are a standard, non-proprietary set of languages and tools that enable modeling, sharing, and reasoning about information. Words, terms and entities on the Semantic Web are connected through meaningful relationships, and thus enable a graph representation of knowledge with rich semantics. Relationship mining from the semantic data is a critical step towards getting useful semantic information for better integration, search and decision-making. One of the most fundamental tasks in relationship mining is to find complex semantic relationships between entities. Complex semantic relationship, also known as semantic association, is defined as an undirected path that connects two resource entities in an RDF graph [3]. A semantic association search seeks to obtain all meaningful relations between two given entities. Table 1 shows a few example results of a semantic association search.

As the volume of semantic data increases, a semantic association search is very likely to return too many results for a human user to digest. To ensure more relevant results shown first to the user, an effective *ranking* method has become an important necessity for semantic association mining. In addition, similar to web search, understanding user preferences is a key challenge in producing personalized semantic association search results. Different users can have different preferences in terms of personal interests and search intentions. Given that such preferences are difficult to be explicitly expressed in current semantic association query languages [6, 15], it is the *ranking* method's responsibility to cater for each individual user's specific preferences.

Many ranking methods have been proposed towards producing personalized semantic association search results. Some methods (*e.g.*, [4, 5, 14]) allow users to manually configure certain ranking parameters. But manual tuning requires users to have a good understanding of the ranking scheme, and can be very difficult for inexperienced users. To alleviate these problems, a learning-to-rank method was proposed in [8] to automatically capture a user's preferences by asking the user to assign ranks to his favorite search results. However, this method has two limitations. First, user labeling is a tedious and time-consuming task; a user has

*This work is supported by Chevron Corp. under the joint project, Center for Interactive Smart Oilfield Technologies (CiSoft), at the University of Southern California.

1	Harry Potter $\xrightarrow{married\_to}$ Ginny Weasley
2	Harry Potter $\xrightarrow{has\_child}$ Lily Luna Potter $\xrightarrow{has\_parent}$ Ginny Weasley
3	Harry Potter $\xrightarrow{is\_founder\_of}$ Dumbledore's Army $\xrightarrow{has\_member}$ Ginny Weasley
4	Harry Potter $\xrightarrow{has\_parent}$ James Potter $\xrightarrow{education}$ Hogwarts School $\xrightarrow{student\_graduate}$ Ginny Weasley

Table 1. Example results of a semantic association search between two fictional character *Harry Potter* **and** *Ginny Weasley*

to examine thousands of results during the labeling process. Second, the ranking function cannot be improved once it is learned from the user-labeled results, which may cause unsatisfactory ranking results if the training data is insufficient to cover all preferences of the user.

To address the above issues and challenges, we present Rankbox, an adaptive ranking system for mining semantic associations with three key features:

- **Personalized**: in Rankbox, each user has his own ranking function that represents his specific preferences.
- **Improvable**: Rankbox supports a refinement mechanism, allowing users to continuously teach the system their preferences.
- **End-user friendly**: users can access Rankbox from any digital devices with a web browser, and conveniently interact with the system through just a few clicks. Users no longer have to deal with manual parameter tuning or tedious labeling.

We make an important observation in this paper: users' opinions about search results can serve as a valuable source of their preferences. Based on this observation, our system allows each user to provide simple feedback about the current search results, learns the user's preferences from his feedback, and incorporates these preferences into a user-specific ranking function. In particular, the front end of our system is a web graphical user interface (GUI) that provides semantic association search results to the user, ranked by his current ranking function; meanwhile, it allows the user to send his subjective opinions on each result (*e.g.*, *like* or *dislike*) back to the system, by a few clicks on the web page. The back end of our system then automatically interprets the user's feedback and employs an adaptive learning-to-rank algorithm to continuously refine his ranking function. In general, the interaction between Rankbox and users iterates though a "searching-ranking-feedback-refinement" cycle. During our user study, most users are satisfied with the ranking results after a few iterations that usually takes only $20 \sim 50$ clicks. Users' burden is significantly reduced compared to examining thousands of results during the labeling process in [8].

The main contributions of this paper include:

1. We propose an adaptive ranking system for semantic association search. The system adapts to user preferences by continuously collecting and learning from user feedback. To the best of our knowledge, our system presents the first interactive learning-to-rank method for semantic association search.

2. Our system acts as a black box to end-users, as it hides all technical details of the ranking scheme from them. Rather than dealing with tedious parameter tuning or training data labeling, users only need to provide simple and intuitive feedback to the system if the current search results need improvement.

3. Our system can be easily accessed from any devices with a modern web browser. The evaluation based on a large link-open-data dataset demonstrates the advantages of our system compared to the state-of-the-art.

2. Related Work

We review the related work from the following two aspects:

2.1 Finding and Ranking Semantic Associations

Several query languages have been proposed to find semantic associations from the semantic data, such as ρ-queries [6] and SPARQLeR [15]. A semantic association query usually takes a pair of entities as input, and returns all undirected paths between the entities that satisfy certain length or property restrictions. With the rapid growth of semantic data, a semantic association search is very likely to return hundreds or even thousands of results.

Many research efforts have been made to identify relevant relationships from a large number of results returned from a semantic association query. An interactive relationship discovery approach is developed in [12] by allowing users to explore the visualization of search results and select the relevant ones; while SemRank [5] and SemDis [4] determine result relevance by adopting a ranking model with

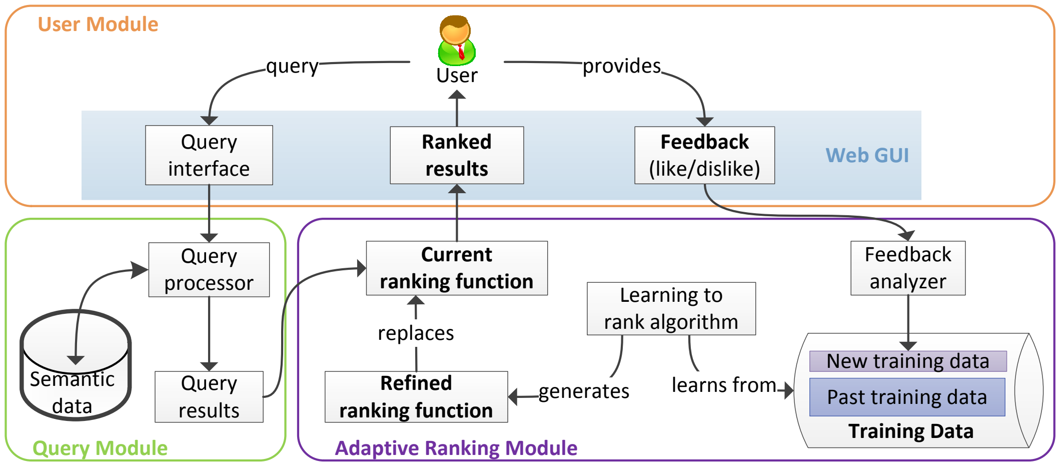


Figure 1. Our adaptive ranking system for semantic association search. Components in bold text are specific to each user.

user-configurable parameters. Different from these methods, our Rankbox system automatically generates ranking functions for determining result relevance by learning from user feedback, rather than relying on visualization or user-configured parameters.

2.2 Learning to Rank

Learning to Rank, also known as machine-learning based ranking, is a set of machine learning algorithms that can automatically construct a ranking model from the training data [16]. The training data consists of queries and ranked lists of results for those queries. A ranking model is learned from the training data, and then applied to rank the results of unseen queries. The objective of the ranking model is to sort the results of unseen queries in a way that is similar to the rankings in the training data.

Current learning-to-rank algorithms can be classified into three types: pointwise (*e.g.*, [9]), pairwise (*e.g.*, [13]), and listwise (*e.g.*, [18]). Pointwise approaches usually use regression algorithms to predict a score for each single query-result pair. Pairwise approaches attack the ranking problem as a classification problem: given a pair of query results, a binary classifier is learned to tell which result is better, with a goal to minimize the average number of inversions in ranking. Listwise algorithms produce ranking models by directly optimizing the ranking quality over all queries in the training data.

A learning-to-rank method for semantic association search is proposed in [8]. This method adopts an SVM-based pairwise algorithm to learn users' preferences from manually labeled training data. The key differences between this method and our Rankbox system include: 1) Rankbox does not require user-labeled training data of search ranks; 2) the ranking function can be continuously improved in our system; 3) Rankbox adopts pointwise learning-to-rank algorithms (detailed in Section 6).

3. System Overview

Figure 1 illustrates an overview of our system. As the system front end, **user module** provides a web GUI that supports users to have three types of interactions with the system: initiating a semantic association search query through the query interface, getting ranked search results, and sending feedback about the current search results back to the system. **Query module** is part of the system back end. Given a semantic association query from the user, the semantic association query processor retrieves results from the semantic knowledge base. The core of our system is the **adaptive ranking module**, which analyzes the feedback from the user and incorporates the feedback into his ranking function. In particular, given a set of query results, the user's current ranking function is employed to sort the results. If the user is not satisfied with the sorted results, he can provide feedback to inform the system of what he *likes* or *dislikes*. The feedback is parsed by a feedback analyzer, and added as new training data for the user. A pointwise learning-to-rank algorithm is employed to learn the user's preferences from both the new and the past training data. A refined ranking function is then produced according to the user's current preferences, and replaces the user's current ranking function. The user can continuously provide feedback to improve the quality of his ranking function through several iterations. System components that incorporate user-specific information (*e.g.*, user preference, user feedback) are highlighted with bold text in Figure 1.

4. Semantic Association Search

Semantic associations are first introduced in [3], as a sequence of consecutive properties that link two resource entities in the RDF data model [17]. Semantic associations describe the existence and the meaning of relations between entities. As many researchers have observed, the discovery of semantic associations is a key in information retrieval on the Semantic Web.

A semantic association search takes an entity node pair $q = (e_i, e_j)$ as input query, and aims to find paths in the RDF graph that connect e_i and e_j through labeled edges. The result set is an unordered collection of semantic associations, denoted as $A(q)$. Table 1 shows a few example results for query $(HarryPotter, GinnyWeasley)$, in an RDF knowledge base we created from the Freebase data [11]. Note that the result set of a given semantic association query is solely determined by the entity pair and the RDF data model, and thus is identical to different users.

In our system, any state-of-the-art semantic association search algorithm (*e.g.*, ρ-queries [6] and SPARQLeR [15]) can serve as the query processor. For efficiency, we adopt a depth-limited search algorithm to find the result set $A(q)$ of a given query q under a certain path length restriction δ.

5. Ranking Search Results

In a large and complicated knowledge base such as Freebase, a semantic association search can return thousands of records even with a strict path length restriction (*e.g.*, $\delta = 10$). Therefore, an effective ranking method is required to sort the search results according to their relevance with respect to a user's preference. To formulate the ranking problem, we introduce a *ranking function* $f(a), a \in A(q)$, which determines a relevance score for each search result. By sorting these relevance scores over the entire search result set $A(q)$, more relevant results are shown to the user first.

In particular, we calculate a set of features for each semantic association $a \in A(q)$, and denote the feature vector as $\mathbf{x}(a)$. The ranking function is defined as a scaler product of the feature vector $\mathbf{x}(a)$ and a user-specific weight vector $\mathbf{w}_u$:

$$f_u(a) = \mathbf{x}(a) \cdot \mathbf{w}_u, \tag{1}$$

where $\mathbf{w}_u$ is a vector with the same length as $\mathbf{x}(a)$, and the subscript u denotes that the ranking function f_u together with the weight vector $\mathbf{w}_u$ specifically reflect user u's preferences. The semantic association feature vector $\mathbf{x}(a)$ and the user-specific weight vector $\mathbf{w}_u$ are detailed in the following subsections individually.

5.1 Semantic association features

An appropriate feature vector $\mathbf{x}(a)$ consists of a series of real numbers that characterize a semantic association from various perspectives, *e.g.*, length, contexts and popularity. These features are calculated automatically for each semantic association in the search result set. The calculation is performed in the back end of our system, and thus is completely invisible to the user. In general, any feature set (*e.g.*, those proposed in previous research such as [3, 8]) can be adopted to our system. Inspired by [8], our Rankbox system currently uses the following features to characterize each semantic association search result:

Association length $l(a)$ is the number of entities contained in a semantic association a.

Topic features quantitatively characterize the topics covered by the entities of a semantic association. In RDF models, schema-level classes and properties can be categorized into several topic regions based on the knowledge domain they describe. Thus, the topic of an entity is determined by the topic region of its corresponding schema-level class. For example, Figure 2 shows a simple RDF model with two topic regions at the schema level. Based on this region division, the topics of entities *Harry Potter* and *student* are determined as *character*, while the entity *Hogwarts School* is about the topic of *education*. We define the topic feature of semantic association a regarding topic i as

$$t_i(a) = \frac{|E_i|}{l(a)},$$

where E_i is the set of a's entities that cover topic i .

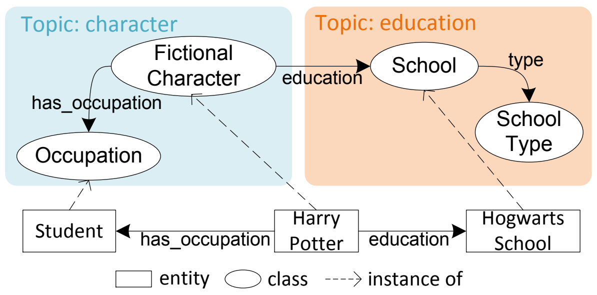


Figure 2. A simple RDF model with two topic regions at the schema level

The feature vector $\mathbf{x}(a)$ of semantic association a is then defined as

$$\mathbf{x}(a) = (l(a), t_1(a), t_2(a), \cdots, t_n(a)) \tag{2}$$

where n is the number of topics.

5.2 Learning $\mathbf{w}_u$ using LDA

The weight vector $\mathbf{w}_u$ in user u's ranking function f_u represents his preferences on the semantic association features. To automatically capture $\mathbf{w}_u$, our system collects two sets of feedback D_u^+ and D_u^- from user u, which contain semantic associations *liked* and *disliked* by him respectively. By representing each semantic association in D_u^+ and D_u^- using its feature vector, we get a set of positive samples X_u^+ and a set of negative samples X_u^-, in the k-dimensional feature space (k is the number of features in a feature vector):

$$X_u^+ = \{\mathbf{x}(a) | a \in D_u^+\}$$
$$X_u^- = \{\mathbf{x}(a) | a \in D_u^-\}$$

We then employ Linear Discriminant Analysis (LDA) [10] to learn $\mathbf{w}_u$ from X_u^+ and X_u^-.

LDA is a supervised machine learning algorithm for classifying two or more groups of objects. Intuitively, LDA seeks to maximize the separation between groups, while preserving as much the group discriminatory information as possible. Given X_u^+ and X_u^- as training data, LDA separates the positive samples and the negative samples by a discriminative hyperplane. After being projected onto the normal to the discriminant hyperplane, samples from the same group are projected very close to each other, while at the same time, the projection means are as farther apart as possible. Figure 3 illustrates this process in a 2-dimensional feature space.

Therefore, the projection of a semantic association feature vector onto the normal to the discriminant hyperplane can yield a quantitative preference value in terms of whether user u likes or dislikes the corresponding semantic association. We thus use the normal to the discriminant hyperplane as $\mathbf{w}_u$. According to Fisher's linear discriminant, $\mathbf{w}_u$ can be estimated as:

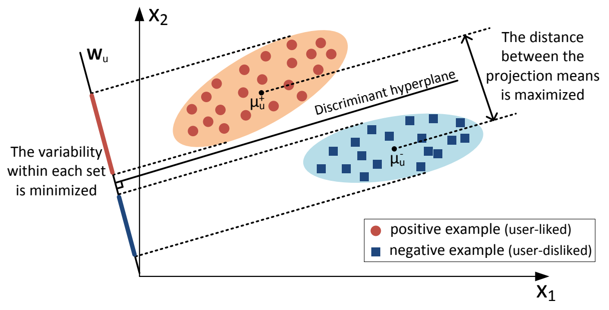
$$\mathbf{w}_u = (\Sigma_u^+ + \Sigma_u^-)^{-1}(\mu_u^+ - \mu_u^-),$$

where μ_u^+, μ_u^-, $\mathbf{\Sigma}_u^+$ and $\mathbf{\Sigma}_u^-$ are the means and the covariances of X_u^+ and X_u^- respectively.

6. Adaptively Learning User Preferences

In this section we detail how to collect user feedback and adapt the user's ranking function to his preferences based on the feedback. In particular, we develop an interactive learning-to-rank algorithm which iterates through a "searching-ranking-feedback-refining" cycle, as shown in Algorithm 1.

In the searching and the ranking phases, given a semantic association query q from user u, a set of search results $A(q)$ is retrieved from the RDF knowledge base using depth-limited search, and then sorted by the user's current ranking function f_u. If it is the first time the user uses our system, a default weight vector $\mathbf{w}_{default}$ is adopted to construct f_u.



X₂
X₁
w_u
The distance between the projection means is maximized
Discriminant hyperplane
μ⁺_u
μ⁻_u
The variability within each set is minimized
● positive example (user-liked)
■ negative example (user-disliked)

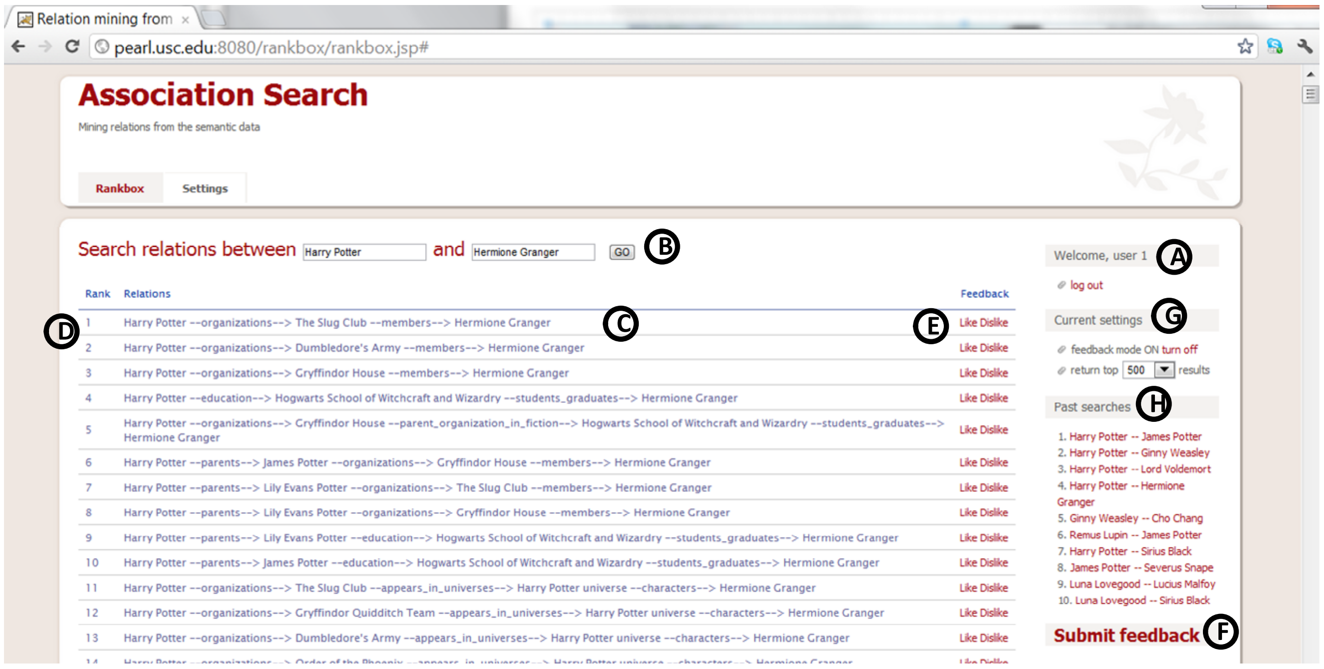
Figure 3. LDA maximizes the distance between the projection means while at the same time minimizing the scatter within the set (illustrated in a 2-dimensional space).

Algorithm 1: InteracitveLtR for user u

```
Initialization: $\mathbf{w}_u \leftarrow \mathbf{w}_{default}, D_u^+ \leftarrow \emptyset, D_u^- \leftarrow \emptyset$
Data: an RDF knowledge base $M_{rdf}$
foreach query $q$ from user $u$ do
    find $A(q)$ by DepthLimitedSearch ($M_{rdf}, q$)
    for $a \in A(q)$ do
        calculate feature vector $\mathbf{x}(a)$
        $f_u(a) \leftarrow \mathbf{x}(a) \cdot \mathbf{w}_u$
    sort $A(q)$ by $f_u(a)$
    get $u$'s feedback $F_u^+$ (results liked by $u$) and $F_u^-$ (results disliked by $u$)
    if $F_u^+ \neq \emptyset$ or $F_u^- \neq \emptyset$ then
        $D_u^+ \leftarrow D_u^+ \bigcup F_u^+$
        $D_u^- \leftarrow D_u^- \bigcup F_u^-$
        apply LDA on ($D_u^+, D_u^-$) to learn a new $\mathbf{w}_u$
```

In the feedback phase, if user u thinks the ordering of $A(q)$ produced by his current f_u needs improvement, our system allows him to give two types of simple and intuitive comments on each semantic association search result: *like* and *dislike*. The system maintains two feedback sets D_u^+ and D_u^- for user u. These sets accumulatively collect all the results liked or disliked by user u from both his past and present feedback phases. Each result commented by user u in the present feedback phase is added to either the positive feedback set D_u^+ or the negative feedback set D_u^-, based on the comment type (*liked* or *disliked*).

In the refining phase, the goal is to refine user u's current ranking function f_u based on the updated feedback sets D_u^+ and D_u^-. Thus, Linear Discriminant Analysis (LDA) is



Association Search

Mining relations from the semantic data

Rankbox Settings

Search relations between Harry Potter and Hermione Granger GO

Rank	Relations	Feedback
1	Harry Potter --organizations--> The Slug Club --members--> Hermione Granger	Like Dislike
2	Harry Potter --organizations--> Dumbledore's Army --members--> Hermione Granger	Like Dislike
3	Harry Potter --organizations--> Gryffindor House --members--> Hermione Granger	Like Dislike
4	Harry Potter --education--> Hogwarts School of Witchcraft and Wizardry --students_graduates--> Hermione Granger	Like Dislike
5	Harry Potter --organizations--> Gryffindor House --parent_organization_in_fiction--> Hogwarts School of Witchcraft and Wizardry --students_graduates--> Hermione Granger	Like Dislike
6	Harry Potter --parents--> James Potter --organizations--> Gryffindor House --members--> Hermione Granger	Like Dislike
7	Harry Potter --parents--> Lily Evans Potter --organizations--> The Slug Club --members--> Hermione Granger	Like Dislike
8	Harry Potter --parents--> Lily Evans Potter --organizations--> Gryffindor House --members--> Hermione Granger	Like Dislike
9	Harry Potter --parents--> Lily Evans Potter --education--> Hogwarts School of Witchcraft and Wizardry --students_graduates--> Hermione Granger	Like Dislike
10	Harry Potter --parents--> James Potter --education--> Hogwarts School of Witchcraft and Wizardry --students_graduates--> Hermione Granger	Like Dislike
11	Harry Potter --organizations--> The Slug Club --appears_in_universes--> Harry Potter universe --characters--> Hermione Granger	Like Dislike
12	Harry Potter --organizations--> Gryffindor Quidditch Team --appears_in_universes--> Harry Potter universe --characters--> Hermione Granger	Like Dislike
13	Harry Potter --organizations--> Dumbledore's Army --appears_in_universes--> Harry Potter universe --characters--> Hermione Granger	Like Dislike
14	Harry Potter --organizations--> Order of the Phoenix --appears_in_universes--> Harry Potter universe --characters--> Hermione Granger	Like Dislike

Welcome, user 1

log out

Current settings

feedback mode ON turn off

return top 500 results

Past searches

1. Harry Potter -- James Potter
2. Harry Potter -- Ginny Weasley
3. Harry Potter -- Lord Voldemort
4. Harry Potter -- Hermione Granger
5. Ginny Weasley -- Cho Chang
6. Remus Lupin -- James Potter
7. Harry Potter -- Sirius Black
8. James Potter -- Severus Snape
9. Luna Lovegood -- Lucius Malfoy
10. Luna Lovegood -- Sirius Black

Submit feedback

Figure 4. The web graphical user interface of Rankbox

employed to learn a new weight vector $\mathbf{w}_u$ from the updated D_u^+ and D_u^-. The ranking function f_u is updated with the new $\mathbf{w}_u$, and used to rank the results of the subsequent queries from user u. A user can continuously refine his ranking function by going through the "searching-ranking-feedback-refining" cycle for many iterations, until he is satisfied with the ranking of search results.

7. System Implementation

Our Rankbox system is developed and deployed on a Tomcat 7 web server. Rankbox can be easily accessed from any digital device with a modern web browser. No software installation is required on the client side. On the server side, the system maintains a semantic association query processor, an RDF knowledge base, and user profiles. Each user profile consists of the user's ranking function and all the history feedback from the user.

The interactions between users and the system go through the friendly web GUI illustrated in Figure 4. The major components of this GUI include: (A) a login bar: a user must login to use and update his profile; (B) a search bar to start a semantic association query; (C) search results; (D) result ranks produced by the user's current ranking function; (E) click on *like* or *dislike* to comment on each result; (F) click to submit user feedback to the system; (G) a configuration bar to view and change current GUI settings, *e.g.*, turn on/off the feedback mode; (H) shortcuts to start the user's past search queries.

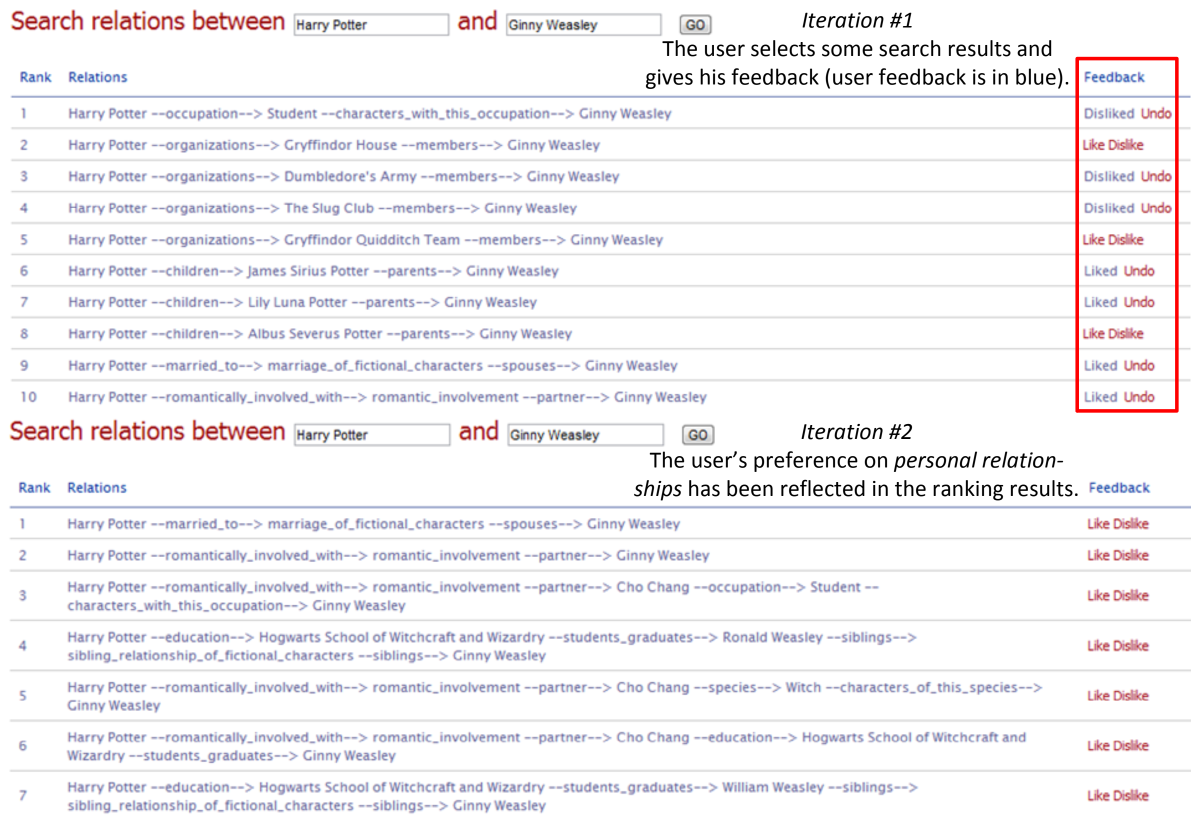
8. Experimental Results

8.1 Experimental setup

Dataset: Our dataset is an RDF knowledge base we created from the *fictional_universe* domain of the Freebase linked-open-data [11]. The RDF knowledge base covers information about all types of fictional works, especially the characters and organizations that appear in them. In particular, our dataset contains 36 topics, 340K instances and 590K properties. The schema of our RDF knowledge base is available at [1].

Ranking methods compared: We compare Rankbox with two state-of-the-art semantic association ranking methods: SemDis and SVMLtR. SemDis refers to the approach proposed in [4], which needs users to manually configure the weights of several ranking metrics. We adopt the recommended weight configuration of its official implementation (the SemDis Project [2]). SVMLtR [8] is an SVM-based learning to rank method for semantic association search. SVMLtR requires users to examine thousands of results and assign ranks to his 10 favorite results.

Users and test queries: We invited 20 human users to participate in our user study. Given that the fiction *Harry Potter* and its characters are known to most of our participants, we designed 20 test queries between the characters in *Harry Potter* (Table 2) for each user to go through.



Search relations between Harry Potter and Ginny Weasley GO

Iteration #1
The user selects some search results and gives his feedback (user feedback is in blue).

Rank	Relations	Feedback
1	Harry Potter --occupation--> Student --characters_with_this_occupation--> Ginny Weasley	Disliked Undo
2	Harry Potter --organizations--> Gryffindor House --members--> Ginny Weasley	Like Dislike
3	Harry Potter --organizations--> Dumbledore's Army --members--> Ginny Weasley	Disliked Undo
4	Harry Potter --organizations--> The Slug Club --members--> Ginny Weasley	Disliked Undo
5	Harry Potter --organizations--> Gryffindor Quidditch Team --members--> Ginny Weasley	Like Dislike
6	Harry Potter --children--> James Sirius Potter --parents--> Ginny Weasley	Liked Undo
7	Harry Potter --children--> Lily Luna Potter --parents--> Ginny Weasley	Liked Undo
8	Harry Potter --children--> Albus Severus Potter --parents--> Ginny Weasley	Like Dislike
9	Harry Potter --married_to--> marriage_of_fictional_characters --spouses--> Ginny Weasley	Liked Undo
10	Harry Potter --romantically_involved_with--> romantic_involvement --partner--> Ginny Weasley	Liked Undo

Search relations between Harry Potter and Ginny Weasley GO

Iteration #2
The user's preference on *personal relationships* has been reflected in the ranking results.

Rank	Relations	Feedback
1	Harry Potter --married_to--> marriage_of_fictional_characters --spouses--> Ginny Weasley	Like Dislike
2	Harry Potter --romantically_involved_with--> romantic_involvement --partner--> Ginny Weasley	Like Dislike
3	Harry Potter --romantically_involved_with--> romantic_involvement --partner--> Cho Chang --occupation--> Student --characters_with_this_occupation--> Ginny Weasley	Like Dislike
4	Harry Potter --education--> Hogwarts School of Witchcraft and Wizardry --students_graduates--> Ronald Weasley --siblings--> sibling_relationship_of_fictional_characters --siblings--> Ginny Weasley	Like Dislike
5	Harry Potter --romantically_involved_with--> romantic_involvement --partner--> Cho Chang --species--> Witch --characters_of_this_species--> Ginny Weasley	Like Dislike
6	Harry Potter --romantically_involved_with--> romantic_involvement --partner--> Cho Chang --education--> Hogwarts School of Witchcraft and Wizardry --students_graduates--> Ginny Weasley	Like Dislike
7	Harry Potter --education--> Hogwarts School of Witchcraft and Wizardry --students_graduates--> William Weasley --siblings--> sibling_relationship_of_fictional_characters --siblings--> Ginny Weasley	Like Dislike

Figure 5. A user's feedback can significantly improve the quality of his ranking function.

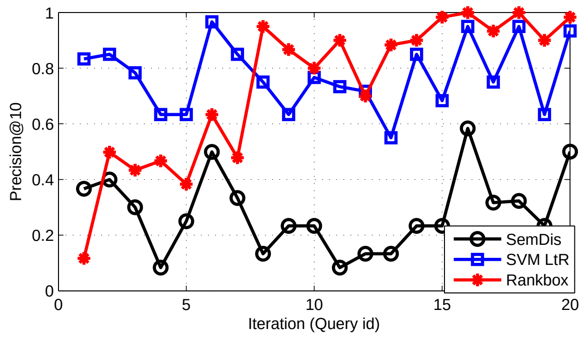


Figure 6. Precision@10 per iteration

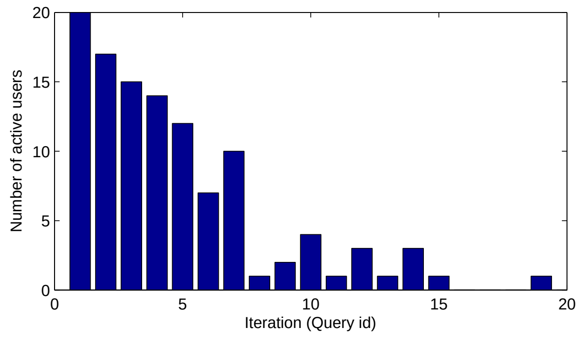


Figure 7. The number of active users per iteration

8.2 Evaluations

Time complexity: The most time-consuming operation in our system is semantic association search, which usually takes 15∼20 seconds to retrieve 500 results per query. The feature calculation, LDA, and sorting all complete within a couple of seconds. Experimental results are measured when the Rankbox server is deployed on a computer with Intel i-7 CPU 1.60GHz and 6GB memory.

Qualitative results: Figure 5 shows an example in which ranking quality can be significantly improved after incorporating user feedback on only 8 results.

User Study: To quantitatively evaluate our system, we ask each user to interact with the system for 20 iterations. In the i-th iteration, each user is instructed to give feedback to at most 10 results. In addition, he is asked to judge the relevance of top-10 results by labeling each as *relevant* or *irrelavant*. These labels are used to evaluate the precision

Query id	Entity 1	Entity 2
1	Ronald Weasley	Cho Chang
2	Professor Albus Dumbledore	James Potter
3	Lily Evans Potter	Neville Longbottom
4	Tom Riddle Sr.	Lily Evans Potter
5	Sirius Black	Remus Lupin
6	Ginny Weasley	George Weasley
7	Professor Severus S-nape	Ginny Weasley
8	Fred Weasley	Lord Voldemort
9	James Potter	Lucius Malfoy
10	Lord Voldemort	James Potter
11	Luna Lovegood	Fred Weasley
12	Draco Malfoy	Fred Weasley
13	Tom Riddle Sr.	Remus Lupin
14	Lord Voldemort	Ginny Weasley
15	Professor Albus Dumbledore	Hermione Granger
16	Ronald Weasley	Ginny Weasley
17	James Potter	Lord Voldemort
18	George Weasley	Fred Weasley
19	James Potter	Lucius Malfoy
20	George Weasley	Professor Albus Dumbledore

Table 2. Test queries

of our system. We employ the following two evaluation metrics:

Precision@10 refers to the ratio of records ranked in the top 10 results that are labeled as *relevant*.

Number of active users@query k is the number of users who give feedback at query k, *i.e.*, in the k-th iteration.

Figure 6 and Figure 7 illustrate the evaluation results using the above metrics. Compared to the other two approaches, Rankbox is the only method that can continuously improve the ranking precision through iterations. After 7 iterations, Rankbox achieves the best precision@10 among all three methods. In addition, as the ranking quality increases, more and more users stop giving feedback to the system. They only become active once they feel that some ranking results are “not good”. Such results usually come from a “difficult query” such as Query 7, 10 and 12.

9 Conclusion

We proposed Rankbox, an adaptive ranking system for mining semantic associations from large-scale semantic data. The core of our system is an interactive learning-to-rank algorithm, which automatically captures user preferences from his feedback about the search results. A user can continuously improve the ranking quality by sending more feedback to the system. Our Rankbox system can be accessed from any devices with a web browser. The evaluation results demonstrated the effectiveness and advantages of our system.

References

[1] Freebase schema explorer. `http://schemaviz.freebaseapps.com/?domain=/fictional_universe`.

[2] Semdis project. `http://lsdis.cs.uga.edu:8080/rankingah/`.

[3] B. Aleman-Meza, C. Halaschek, I. Arpinar, and A. Sheth. Context-aware semantic association ranking. In *SWDB’03: 33-50, Berlin, Germany*, 2003.

[4] B. Aleman-Meza, C. Halaschek-Wiener, I. B. Arpinar, C. Ramakrishnan, and A. Sheth. Ranking complex relationships on the semantic web. *IEEE Internet Computing*, 2005.

[5] K. Anyanwu, A. Maduko, and A. Sheth. Semrank: Ranking complex relationship search results on the semantic web. In *the 14th International World Wide Web Conference*, 2005.

[6] K. Anyanwu and A. Sheth. ρ-queries,enabling querying for semantic associations on the semantic web. In *the 12th International World Wide Web Conference*, 2003.

[7] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[8] N. Chen and V. K. Prasanna. Learning to rank complex semantic relationships. Technical report, University of Southern California, November 2011. Available at `http://www-scf.usc.edu/~nchen/paper/ltr.pdf`.

[9] K. Crammer and Y. Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems 14*, pages 641–647, 2001.

[10] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 1999.

[11] Google. Freebase data dumps. `http://download.freebase.com/datadumps/`, 2012.

[12] P. Heim, S. Lohmann, and T. Stegemann. Interactive relationship discovery via the semantic web. In *The Semantic Web: Research and Applications*. 2010.

[13] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

[14] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. Naga: Searching and ranking knowledge. *Data Engineering, International Conference on*, 0:953–962, 2008.

[15] K. J. Kochut and M. Janik. Sparqler: Extended sparql for semantic association discovery. In *4th European conference on The Semantic Web: Research and Applications*, 2007.

[16] T.-Y. Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 2009.

[17] F. Manola and E. Miller. Rdf primer. In *W3C Recommendation*, 2004.

[18] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, 2008.