Learning to Rank Complex Semantic Relationships

Na Chen
Viktor K. Prasanna
University of Southern California, USA

ABSTRACT

This paper presents a novel ranking method for complex semantic relationship (semantic association) search based on user preferences. Our method employs a learning-to-rank algorithm to capture each user's preferences. Using this, it automatically constructs a personalized ranking function for the user. The ranking function is then used to sort the results of each subsequent query by the user. Query results that more closely match the user's preferences gain higher ranks. Our method is evaluated using a real-world RDF knowledge base created from Freebase linked-open-data. The experimental results show that our method significantly improves the ranking quality in terms of capturing user preferences, compared with the state-of-the-art.

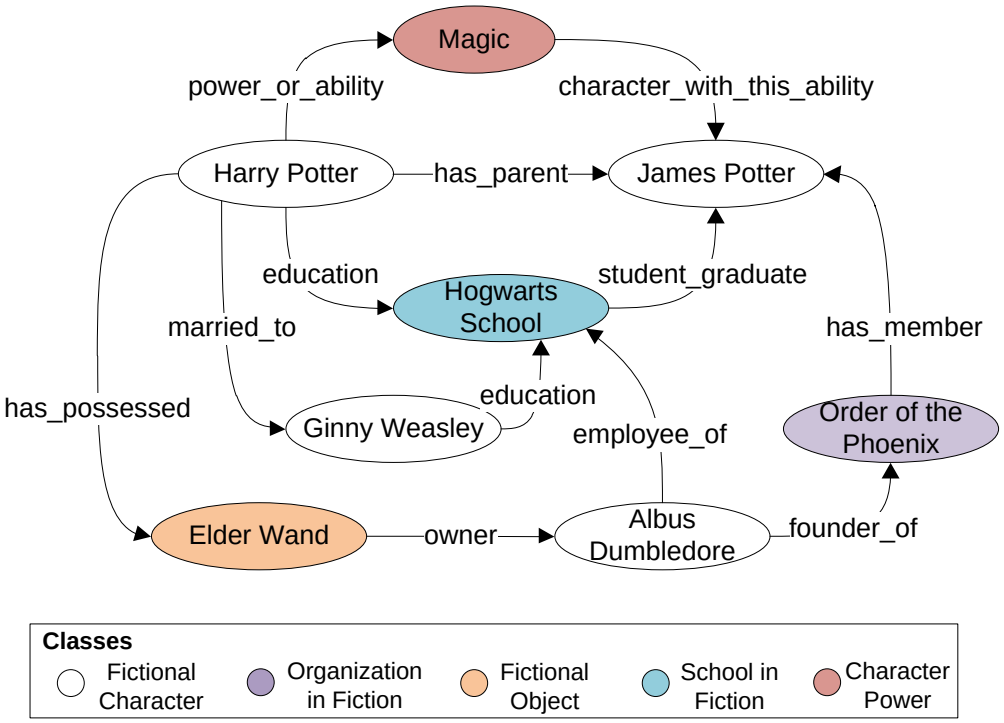
Keywords: semantic association, complex semantic relationship, learning to rank, user preferences, Freebase, Semantic Web.

INTRODUCTION

Semantic relationships are at the heart of ontologies. They connect words, terms and entities through meaning, and thus enable a graph representation of knowledge with rich semantics. Complex semantic relationship, also known as *semantic association* (Aleman-Meza *et al*., 2003), is a sequence of consecutive properties that link two resource entities; in RDF graphs, it is a path consisting of labeled edges that connects two entity nodes. Semantic association mining is a critical step towards getting useful semantic information for better integration, search and decision-making. A number of search techniques and query languages have been developed for discovering semantic associations, such as ρ-Queries (Anyanwu & Sheth, 2003) and SPARQLeR (Kochut & Janik, 2007). A semantic association search query consists of a pair of entities, and the results contain all the semantic associations between them.

With the amount, scale and complexity of ontologies growing rapidly, the number of semantic associations between a pair of entities is becoming increasingly overwhelming. Thus, a semantic association search is very likely to return too many results for a user to digest. For example, we parsed the entire *fictional_universe* domain of Freebase linked-open-data (Google, 2011) into an RDF knowledge base containing 192K resources and 411K properties. We observed that in such a knowledge base, even a simple query (*e.g*., between *Harry Potter* and *James Potter*) with a strict path length restriction (*e.g*., 10) returns thousands of semantic associations. Thus, an effective ranking technique is need for identifying the most relevant results.

Figure 1. A small fraction of the RDF knowledge base we created from Freebase data under the topic "fictional_universe". The color of each instance node denotes its class.



A fundamental challenge in semantic association ranking is to understand user preferences. Different users can have different preferences in terms of personal interests and search intentions. For example, Figure 1 is a small fraction of the RDF knowledge base we created from Freebase data (Google, 2011). Given a query "finding semantic associations between *Harry Potter* and *James Potter*", a few typical search results are listed in Table 1. Among these results, a user who is familiar with the fiction *Harry Potter* would find complicated relationships such as Result 4 more informative, while another user interested in the topic of superpower may want relationships like Result 5 to gain higher ranks. Given that such preferences are difficult to be explicitly expressed in current semantic association query languages (Anyanwu & Sheth, 2003; Kochut & Janik, 2007), it is the ranking method's responsibility to cater for each individual user's specific preferences.

Many ranking methods have been proposed towards personalized semantic association search, by allowing users to manually tune a predefined set of ranking metrics. For example, SemRank (Anyanwu *et al*., 2005) provides a slider bar for users to vary the rank mode between *Conventional* and *Discovery* via a graphical interface, and thus allows them to view their results through different lenses; while the ranking method in Aleman-Meza *et al*. (2005) allows users to specify weights of certain metrics. However, manually adjusting the ranking metrics is a subtle task, which requires a considerable amount of time and effort. In addition, manual tuning can become extremely difficult when the volume and the complexity of search results are overwhelming, which is very common in large knowledge bases.

Table 1. Typical results of the semantic association search between "Harry Potter" and "James Potter"

1	Harry Potter —*has_parent*→ James Potter
2	Harry Potter —*education*→ Hogwarts School —*student_graduate*→ James Potter
3	Harry Potter —*married_to*→ Ginny Weasley —*education*→ Hogwarts School —*student_graduate*→ James Potter
4	Harry Potter —*has_possessed*→ Elder Wand —*owner*→ Albus Dumbledore —*founder_of*→ Order of the Phoenix —*has_member*→ James Potter
5	Harry Potter —*power_or_ablity*→ Magic —*character_with_this_ablity*→ James Potter

The goal of our work is to automatically capture user preferences and effectively leverage these preferences to personalize semantic association search results. Our method is motivated by the notion that, the user who starts the semantic association query should be the best judge of the relevance of a query result. Thus, user assessments on the search results, as user-specific information, can serve as a valuable source of user preferences. In this paper, we present a machine-learning based ranking method to automatically capture a user's preferences from his assessments on the search results. Our method creates a personalized ranking function for each user based on the learned preferences. The ranking function is then used to improve the relevance of search results for the user's subsequent queries. In particular, our method allows each user to assess the results of a small set of randomly selected queries, by assigning ranks to a few of his favorite results. Each result semantic association is characterized by a set of quantitative features. We use an SVM-based learning-to-rank model to capture a user's preferences on these features from user-assessed results.

Our main contributions include:

1. To the best of our knowledge, our method is the first learning-to-rank method for ranking semantic association search results. It automatically captures user preferences and generates personalized ranking results without any manual tuning.

2. We design and implement a learning-to-rank based semantic association search system to evaluate our method. Our evaluation over a large real-world RDF knowledge base shows that, our method significantly improves the ranking quality compared with the state-of-the-art.

RELATED WORK

Our work is related to the following research areas.

Ranking on the Semantic Web

Ranking knowledge on the Semantic Web has recently received a great amount of research interests. Many efforts have been made to address this problem from three different levels: ontology level, resource level and relationship level.

For ontology and resource level ranking, the objective is to determine the relevance of each individual ontology and resource respectively. For example, AKTiveRank (Alani *et al*., 2006) applies a number of analytic methods to rank ontologies based on how well they represent the given search terms. Harth et.al (2009) and Ding et.al (2005) use methods similar to PageRank (Page *et al*., 1999) to rank ontologies and resources by analyzing links and referrals between them.

In contrast, ranking methods at the relationship level aim to determine the relevance of semantic relationships between a pair of resources, such as SemRank (Anyanwu *et al*., 2005) and the method proposed in (Aleman-Meza et.al, 2005) as mentioned in Section *Introduction*. The latter approach is later implemented on a large RDF metabase by Halaschek et.al (2004). Another example for relationship-level ranking is NAGA (Kasneci *et al*., 2008), which ranks semantic associations based on three metrics: confidence, informativeness and compactness. NAGA employs two configurable parameters that can be manually tuned for better ranking quality.

Our method stays at the relationship level. Compared to other ranking methods for semantic relationships, we introduce a novel machine learning based ranking approach. Our method can provide personalized ranking results with minimal user interference, while others all require a certain amount of manual tuning to achieve such results.

Machine-learned Ranking

Machine-learned ranking (MLR), or learning to rank, is a type of supervised or semi-supervised machine learning problem, which automatically constructs a ranking model from training data (Liu, 2009). The input to MLR methods is a set of training data consisting of queries and ranked lists of results that match those queries. A ranking function is learned based on the training data, and then applied to rank results of unseen queries. The purpose of the ranking function is to produce a permutation of results of unseen queries in a way that is similar to rankings in the training data.

Existing algorithms for MLR problems can be classified into three categories: pointwise (*e.g*., Crammer & Singer (2001)), pairwise (*e.g*., Joachims (2002)), and listwise (*e.g*., Xia *et al*. (2008)) approaches. Pointwise approaches usually use regression algorithms to predict a score for each single query-result pair. In pairwise MLR methods, the ranking problem is approximated by 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 average number of inversions in ranking. Listwise algorithms generate ranking model by directly optimizing the ranking quality over all queries in the training data. The MLR method we use in this paper belongs to the pairwise category.

SEMANTIC ASSOCIATION FEATURES

To formulate the ranking problem, we first need a set of features to characterize semantic associations from various aspects. Prior semantic association ranking approaches (Aleman-Meza *et al*., 2005; Kasneci *et al*., 2008) use limited number of features (usually less than 10), and require users to manually specify the weights of the features. In order to get desirable results through manual specification, a user must have good understanding of the ranking scheme. However, for users who are not familiar with the ranking scheme, they are likely to get unsatisfactory results because of improperly tuned ranking criteria. In addition, as more features are used to comprehensively describe a semantic association, it will become very time-consuming and tedious, even for experienced users, to explicitly specify their preferences by manually tuning the weights of tens of features.

The goal of our learning-to-rank method is to automatically capture user preferences on association features, and generate a personalized ranking function for each user accordingly. Thus, users are released from the burden of manually tuning the parameters. In addition, our learning-to-rank method does not rely on a specific set of features. There is no limitation on the

number of features used to describe an association. In general, any association feature set (*e.g.* those proposed in prior research such as Aleman-Meza et.al (2005)) containing any number of features can be adopted by our method. Thus, our method can support comprehensive description of associations without incurring any overload to users. In this paper, we choose to use a relatively large feature set that can be directly calculated from semantic associations. These semantic association features are detailed in the following subsections.

Association Length

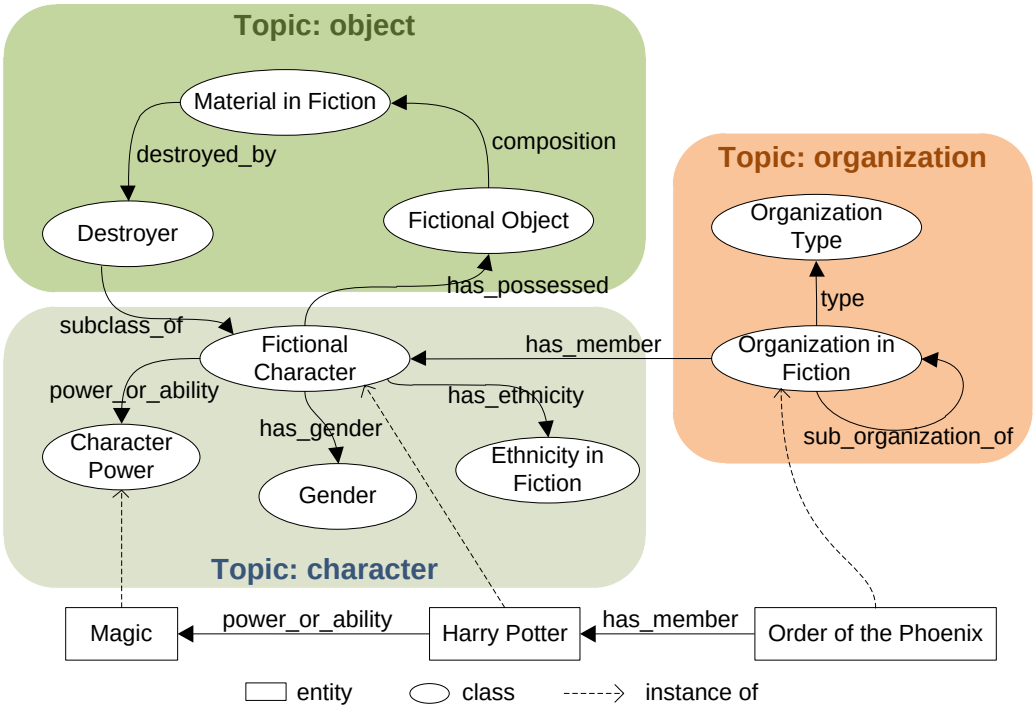
Association length is a metric for measuring the amount of properties contained in a semantic association. The longer a semantic association is, the more properties it contains. Assume A is a semantic association, $E_A = \{e \mid e \in A\}$ and $P_A = \{p \mid p \in A\}$ denote the entities (nodes) and properties (edges) in A respectively, the length of A is defined as the number of its properties:

$$L_A = \mid P_A \mid . \quad (1)$$

Topic Features

Topic features quantitatively characterize the topics covered by the entities of a semantic association. In a large RDF model, classes at the schema level can be categorized into several topic regions based on the knowledge domain they describe. Thus, the topic of an entity can be determined by the topic region of its corresponding schema-level class. For example, Figure 2 illustrates three topic regions in a small schema. Based on this region division, entities *Harry Potter* and *Magic* are abouth the topic of character, while the topic of the entity *Order of the Phoenix* is organization. Intuitively, the topic of a semantic association *A* is decided by the topics of its entities. Thus, we define the topic feature of *A* with respect to a topic S_i as

Figure 2. A small fraction of our Freebase knowledge base schema with three topic regions



$$C_A(S_i) = \frac{|E_i|}{L_A + 1}, \tag{2}$$

where $E_i = \{e \mid e \in E_A \wedge typeOf(e) \in S_i\}$ contains entities that belongs to topic S_i, and *typeOf*(e) denotes the class of entity e.

In practice, the topic regions of schema-level classes can be determined either by letting user specify through an ontology visualization tool, or by analyzing the provenance of the classes.

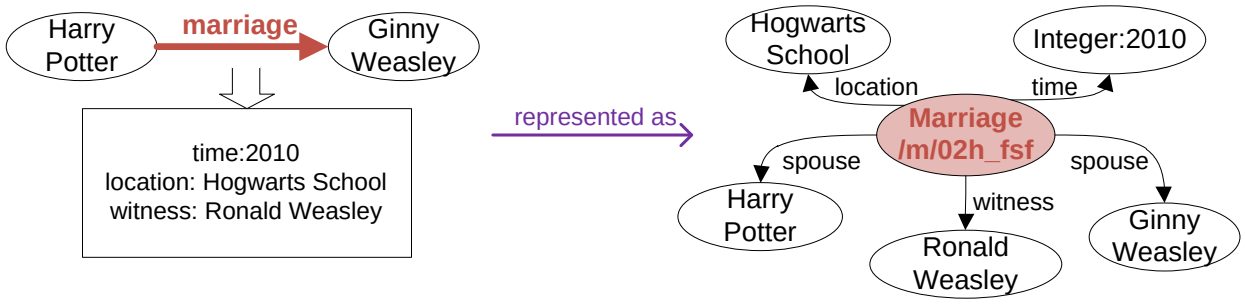
Relation Complexity

In a large linked dataset like Freebase, it is common that relations between two entities can be complex and have its own properties. A typical solution in ontology engineering is to create a relation node to represent such complex relation and assign properties to this node. Figure 3 illustrates an example of a complex relation node. We use the proportion of complex relation nodes to define relation complexity PC_A of a semantic associati on A:

$$PC_A = \frac{|M|}{L_A + 1}, \tag{3}$$

where $M = \{e \mid e \in E_A \wedge e$ is a complex relation node$\}$.

Figure 3. An example of a complex relation node



Property Frequency Features

Frequency of a property in a semantic association is an important hint about the rarity and commonness of the property. For example, in our Freebase dataset, for an organization *Order of the Phoenix*, it has eleven *has_member* properties and one *has_founder* property, which indicates that *has_member* is a common property while *has_founder* is rare. The frequency of each property can collectively decide the rarity and commonness of the entire semantic association. Given a property from e_x to e_y denoted by $p_i : (e_x, e_y)$, we define incoming frequency and outcoming frequency of p_i as:

$$f_{out}(p_i) = \frac{|P_i^{out}|}{d_{out}(e_x)}, \tag{4}$$

$$f_{in}(p_i) = \frac{|P_i^{in}|}{d_{in}(e_y)}, \tag{5}$$

where $P_i^{out} = \{p \mid p : (e_x, e) \wedge typeOf(p) = typeOf(p_i)\}$,
$P_i^{in} = \{p \mid p : (e, e_y) \wedge typeOf(p) = typeOf(p_i)\}$, e is an arbitrary entity and *typeOf* (*p*) denotes the class of property *p*; $d_{in}(e)$ and $d_{out}(e)$ denote the number of incoming and outgoing properties of e respectively.

Let *A* denote a semantic association between entities e_s and e_t, $d_{in}(e_s) = 0$, $d_{out}(e_t) = 0$, $PF_A = \{f(p) = f_{in}(p) + f_{out}(p) \mid p \in P_A\}$, we use four statistical features (average value, standard variance, minimum value and maximum value) to measure the overall property frequency of *A*:

$$\mathsf{F}_A^p = \{\mu(PF_A), \sigma(PF_A), \min(PF_A), \max(PF_A)\}. \quad (6)$$

Popularity Features

The number of incoming and outgoing properties of an entity can be viewed as a hint of its popularity (Aleman-Meza *et al.*, 2005). A semantic association with many popular entities is also likely to be popular. Assume $D_A = \{d(e) = d_{in}(e) + d_{out}(e) \mid e \in E_A\}$, similar to equation (6), we use the following statistical features to describe popularity of a semantic association A:

$$\mathsf{F}_A^e = \{\mu(D_A), \sigma(D_A), \min(D_A), \max(D_A)\}. \quad (7)$$

Feature Vector

Based on the features defined in the above sections, we define the feature vector of a semantic association A as

$$\mathbf{x}_A = (L_A, C_A(S_1), \ldots, C_A(S_k), PC_A, \mu(PF_A), \sigma(PF_A), \min(PF_A), \max(PF_A), \frac{\mu(D_A)}{\max(D_A)}, \frac{\sigma(D_A)}{\max(D_A)}, \frac{\min(D_A)}{\max(D_A)}) \quad (8)$$

where k is the number of subdomains in the schema. Each feature vector $\mathbf{x}$ corresponds to a point in the feature space X, which is a $(k+9)$-dimensional space.

LEARNING TO RANK

The Ranking Framework

The core of our system is a machine-learned ranking algorithm. The algorithm learns user preferences on the feature vector, and generates a personalized ranking function for each user. To collect training samples, for a specific user, we ask him to assign ranks to semantic associations returned from a query q_i, which belongs to a small set of random selected queries, $q_i \in Q$. The feature vector $\mathbf{x}_i$ of each semantic association search result is paired with its user-assigned rank r_i^*. A training sample consists of all such pairs given a random selected query q_i:

$$S(q_i) = \{(\mathbf{x}_1, r_1^*), (\mathbf{x}_2, r_2^*), \ldots, (\mathbf{x}_n, r_n^*)\}. \quad (9)$$

Given training data set $\mathsf{D} = \cup S(q_i)$, our objective is to learn a linear ranking function $h : X \mapsto \mathsf{R}$ which calculates a score from a feature vector

$$h(\mathbf{x}) = \mathbf{w}^T \cdot \mathbf{x}, \quad (10)$$

where $\mathbf{w}$ is a vector with the same length as $\mathbf{x}$. Vector $\mathbf{w}$ is learned for each particular user, and the corresponding personalized ranking function is used for the user's subsequent queries. Given a list of semantic associations returned from a query, we first calculate a score for each semantic

association by applying h over its feature vector; then sort the entire list based on the scores. Thus, a personalized ranking based on user preferences is produced.

The Ranking SVM Algorithm

We choose to employ a pairwise machine-learned ranking algorithm for the learning process, because: first, pairwise machine-learned ranking algorithms benefit from advanced binary classifiers, and thus are adequate to our problem; second, the distribution-skew problem (Liu, 2009) of the pairwise machine-learned ranking algorithms does not exist in our case, since the training samples in our training set are of a fixed size (detailed in Section *Evaluation*).

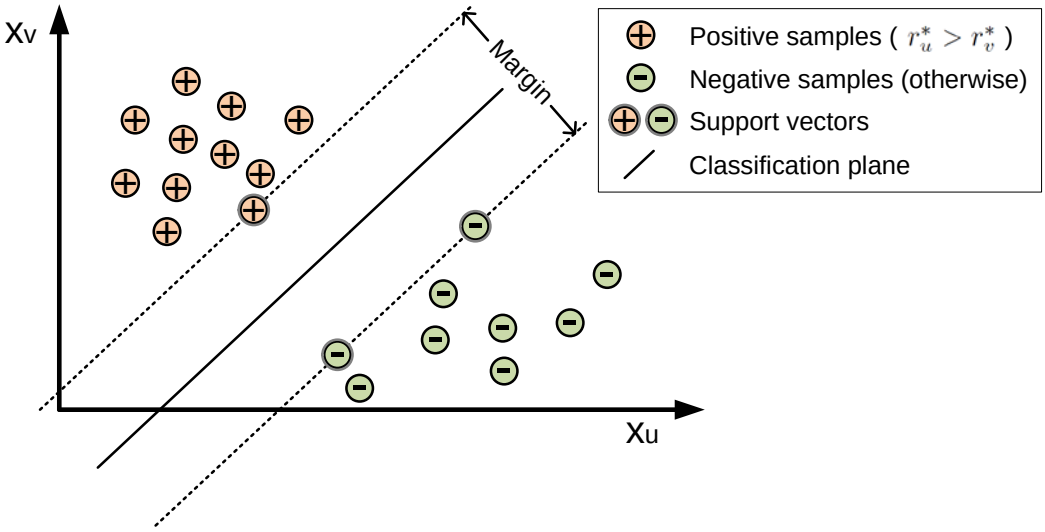
In particular, for a pair of user-ranked associations $(\mathbf{x}_u, r_u^*)$ and $(\mathbf{x}_v, r_v^*)$ from the same training sample $S(q_i)$, we compare their ranks and denote the preference value by

$$y_{u,v} = \begin{cases} 1 & \text{if } r_u^* > r_v^* \\ -1 & \text{otherwise} \end{cases} \tag{11}$$

Therefore, the ranking problem is reduced to a binary classification problem with target classification function $h^*(\mathbf{x}_u, \mathbf{x}_v) = y_{u,v} \in \{\pm 1\}$. We adopt a soft-margin support vector machine (SVM) algorithm (Cortes & Vapnik, 1995) to learn a linear classifier which separates positive samples and negative samples on space $X \times X$ as cleanly as possible, while maximizing the margin between them, as demonstrated in Figure 4.

In our ranking problem, positive samples and negative samples are always created in pairs, *i.e.*, $h^*(\mathbf{x}_u, \mathbf{x}_v) = -h^*(\mathbf{x}_v, \mathbf{x}_u)$; which are symmetric to hyperplane $L: \mathbf{x}_u = \mathbf{x}_v$. Therefore, the classification plane always contains hyperplane L and thus can be expressed in the form of $L_c: (\mathbf{w}, -\mathbf{w})^T \cdot (\mathbf{x}_u, \mathbf{x}_v) = 0$; where $\mathbf{w}$ is the only unknown variable which is learned by the SVM algorithm. The classification problem is equivalent to calculating $\mathbf{w} \cdot \mathbf{x}_u$ and $\mathbf{w} \cdot \mathbf{x}_v$ respectively and comparing these two scores. Ranking results are obtained by sorting $h(\mathbf{x}) = \mathbf{w}^T \cdot \mathbf{x}$ over all the semantic association search results.

Figure 4. Support vector machine algorithm: a linear classifier is learned to separate positive and negative samples with maximal margin.



User Preferences and Weight Vector

The linear ranking function h can be also viewed as a weighted sum of the feature vector. The weight vector $\mathbf{w}$ reflects the importance of each feature for a particular user. Since $\mathbf{w}$ is learned per user, different users may have different weight vectors. The weight of a particular feature represents the importance of this feature to the user, *i.e.*, the user's preference of this feature. Table 2 demonstrates the preferences of three different users, in which User 1 and User 2 are most interested in semantic associations with more complex relations, while User3 is most interested in results about a particular subdomain.

Table 2. Features sorted by importance in descending order for different users

User ID	Features sorted by importance (corresponding weights $\|w_i\|$)
1	$PC_A \succ C_A(S_3) \succ L_A \succ C_A(S_{35}) \succ \cdots$
2	$PC_A \succ \mu(D_A)/\max(D_A) \succ C_A(S_5) \succ C_A(S_{16}) \succ \cdots$
3	$C_A(S_{17}) \succ \mu(D_A)/\max(D_A) \succ C_A(S_3) \succ \cdots$

SYSTEM IMPLEMENTATION

To validate the effectiveness of our ranking method, we design and implement the semantic association search and ranking system as shown in Figure 5. It consists of five components:

1. **Data Module**: The original data of Freebase is in the form of .tsv files. We create a parser to automatically parse Freebase data into a large RDF knowledge base.
2. **User Module**: A user can have three types of interactions with the system: (1) initiating a semantic association query regarding a pair of resources; (2) viewing personalized ranked results of the query; (3) training the system to better understand his preferences by assigning ranks to his favorite results of a training query.
3. **Query Module**: Query processor takes the input pair of resources and retrieves all the semantic associations between them from the RDF knowledge base.
4. **Ranking Module**: Features calculator computes the features defined in Section *Semantic Association Features* for each query result. The personalized ranking function takes in query results with their features, and ranks them according to the user's preferences.
5. **Learning Module**: Training data consists of user-ranked results for a training query and features of these results. A few unranked results and their features are also selected to reduce the skew of the ranking function (detailed in Section *Evaluation*). We adopt a soft-margin support vector machine (Joachims, 2002) to implement our ranking SVM algorithm.

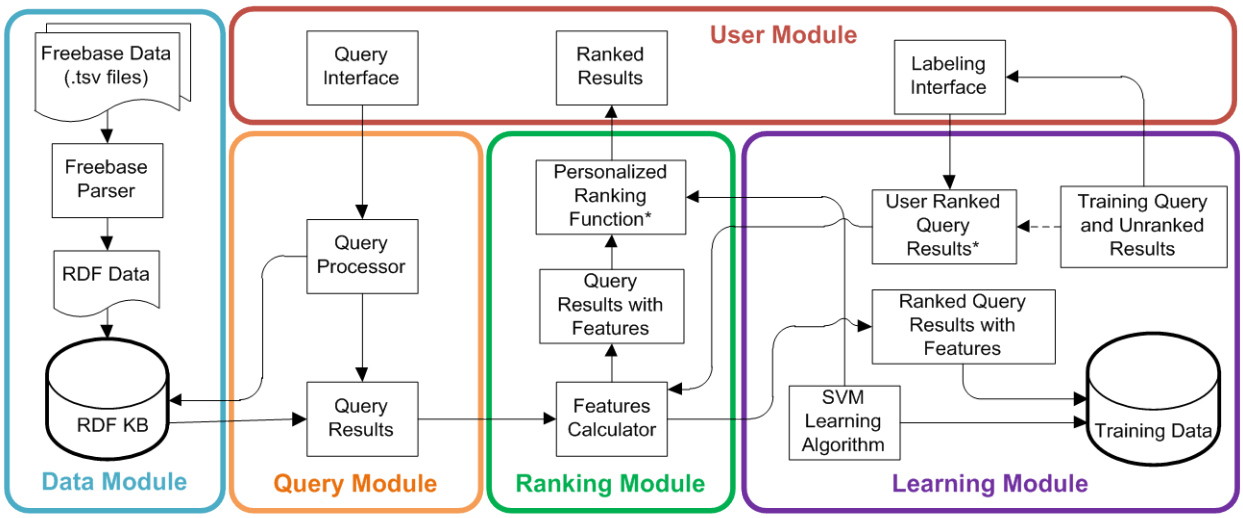
EVALUATION

The objective of incorporating learning capability into our ranking method is to improve the relevance of semantic association search results for each individual user. To this end, we evaluate the per-user and the overall ranking quality of our method on the ranking system described in Section *Implementation*. In particular, we compare the ranking quality of our method to two other methods under various qualitative and quantitative metrics.

Experimental Setup

Benchmarks Our experiments were conducted on a consumer level laptop (Intel i-7 CPU 1.60GHz with 6GB memory). Our dataset is an RDF knowledge base we created from the entire

Figure 5. Architecture of our ranking system (∗ : specific to each user.)



fictional_universe domain of Freebase linked-open-data. The RDF knowledge base covers information about all types of fictional works, especially the characters and organizations that appear in them. Our dataset contains 192K entities (185K regular entity nodes and 7K complex relation nodes) and 411K properties. The schema of our RDF knowledge base is available at Freebase Schema Explorer (2011). In addition, our dataset covers 36 topics that describe the *fictional_universe* domain from various aspects. Table 3 shows the top 10 topics with the most number of entities.

Table 3. Top 10 topics with the most number of entities

Topic	Fictional character	Work of fiction	Romantic involvement	Fictional setting	Character creator
# of instances	150, 832	22,551	3,179	2,149	1,685
Topic	Sibling relationship	Fictional organization	Employment tenure	Person in fiction	Character species
# of instances	1,561	1,247	1,166	1,097	1,064

For each semantic association search query q_i, our search engine employs a depth limited search algorithm to find the semantic associations that satisfy q_i with a length smaller than a given threshold δ. The search engine then outputs the first K results as the result set $T(q_i)$. In our experiments, we retrieve the first 2000 results of which the length is smaller than 10, *i.e.*, $\delta = 10$ and $K = 2000$.

Ranking methods compared We compare our method (**LtR**) with two other approaches: **baseline** and **LtR_CA**. The **baseline** approach is presented in (Aleman-Meza *et al*., 2005). This approach defines a set of semantic association features, and requires users to manually assign weights to the features. We adopt the recommended weight assignment used in its official implementation (SemDis Project, 2011). Our feature set and the feature set in the baseline

approach both use association length and topic features. But for topic features, we analyze the provenance information of classes to decide topic regions, rather than letting the users specify them. The rest of the features in the two sets are all different. To ensure the soundness of our evaluation, we also test our learning-to-rank framework with the feature set used in the baseline approach. This method is denoted as **LtR_CA**, in which feature weights are learned using our SVM ranking model.

Note that we do not compare with SemRank (Anyanwu *et al*., 2005) because: first, the ranking criteria used in SemRank is not based on features of semantic associations, while our method (LtR), the baseline approach, and LtR_CA are all feature-based methods; second, SemRank aims at providing a tool for users to look through different lenses between *Conventional* and *Discovering*, while this work is focused on capturing different user's preferences. Thus, we have not included SemRank in the comparison.

Training queries and test queries We invited 20 granduate students to participate in the evaluation. Given that the fiction *Harry Potter* and its characters are known to most of our participants, we designed a query set containing 30 queries between the characters in *Harry Potter*. The entire query set and detailed statistics can be found in Table 5. For each user, 10 queries $q_1 \sim q_{10}$ are randomly selected from the query set. The user is instructed to label the corresponding result sets $T(q_1) \sim T(q_{10})$. In particular, for each result set, the user needs to read through the entire set, pick 10 most interesting results, and assign them rank $1 \sim 10$ based on his preferences. In order consider the effect of unranked semantic associations, M unranked results are randomly selected and assigned rank 11, which creates $10M$ more pairs to ensure that a user-ranked result is more important than an unranked one. These pairs reward the pairs between the top-10 ranked results, but should not dominate the training set. Thus, we choose a small M in our experiments ($M = 5$). *I.e*., each labeled result set $S(q_i)$ contains 15 results. Finally, from the10 queries $q_1 \sim q_{10}$, we randomly choose 5 as training queries, and use the rest 5 as test queries.

Ranking Results

Figure 6 contains four screenshots taken from our implementation. It illustrates the ranking results of our method and the baseline for two users on the same query (*i.e.*, search semantic associations between *Ginny Weasley* and *Cho Chang*). Each screenshot shows the top six ranking results from one method for one user. Our method demonstrates a significant advantage in capturing different users' preferences. For User 1, the first six results of our method successfully capture his top six favorite semantic associations; while for the baseline, only two of his ten favorite results with relative low ranks are shown in the screenshot. For User 2, none of her favorite ten results is captured by the baseline, but our method is still able to capture her top two favorite results at Rank1 and Rank 2.

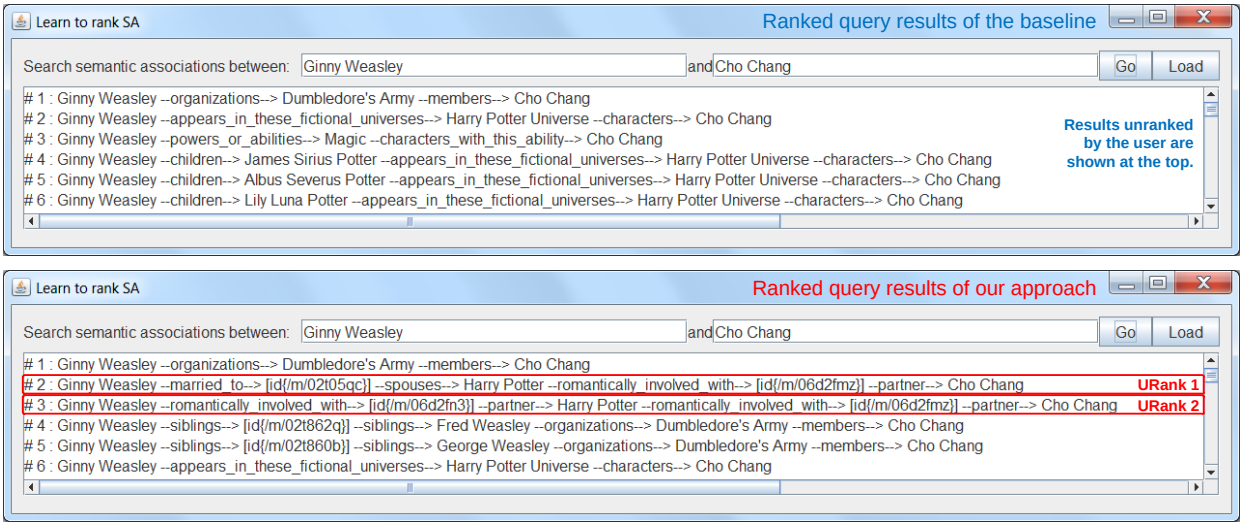
Comparison 1: Ranking Quality per User

We compare the ranking quality of our method (**LtR**), the **baseline** approach and **LtR_CA**, in terms of efficiency and capturing user preferences for each individual user, based on the following metrics:

1. Time complexity The training process takes only a few seconds and it is just performed once

Figure 6. Ranking results of our method and the baseline. The number "#i" ahead of each result is the rank produced by the corresponding ranking approach. URank denotes user-assigned rank (ground truth). According to our experiment setup, only the user's most favorite 10 results have URank values.

(a) Ranked query result for ***User 1***: ranking results of the baseline (top) and our approach(bottom)



(b) Ranked query result for ***User 2*** ranking results of the baseline (top) and our approach(bottom)

for each user. *I.e.*, once the personalized ranking function is learned, it performs as fast as a weighted sum function for all the subsequent queries. In the testing phase, for all three methods, most of the time is consumed on path finding, which takes a couple of seconds. The additional overhead of feature analysis and ranking is very small.

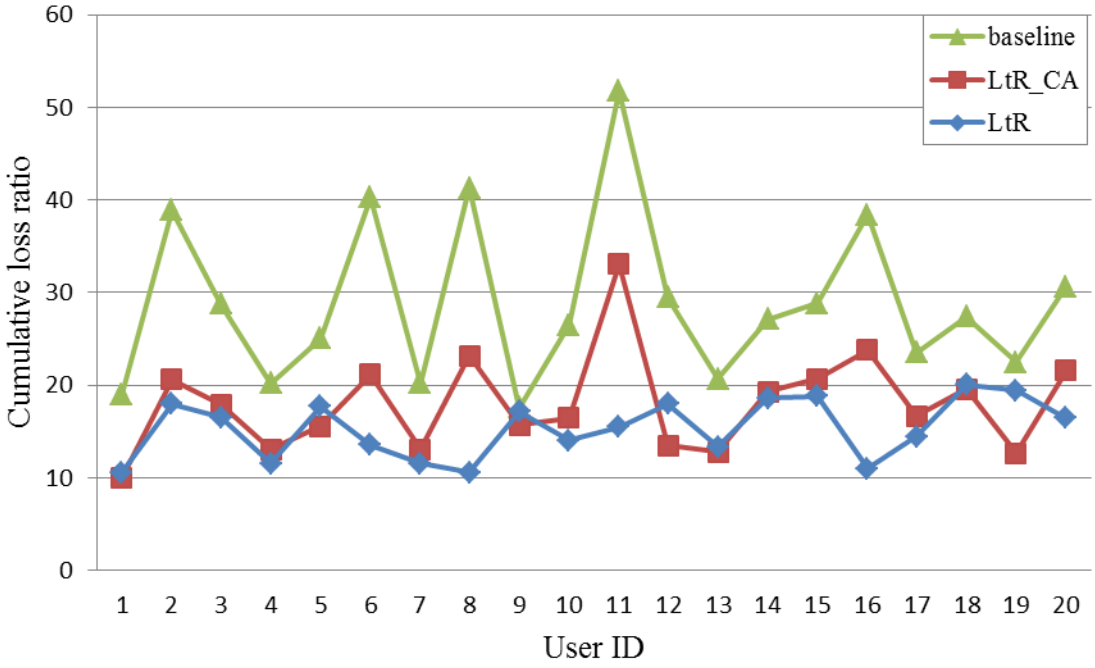
2. Cumulative loss ratio

$$r_{loss} = \frac{1}{N_{\mathrm{L}}} \cdot \mathrm{L} = \frac{1}{N_{\mathrm{L}}} \sum_{(u,v)} L(h; \mathbf{x}_u, \mathbf{x}_v, y_{u,v}) = \frac{1}{N_{\mathrm{L}}} \sum_{(u,v)} \frac{|\, h(\mathbf{x}_u, \mathbf{x}_v) - y_{u,v} \,|}{2} \tag{12}$$

where L is the cumulative loss of a user's ranking function, and N_{L} denotes the number of all comparable pairs. In particular, L counts the number of swapped pairs between ground truths (user-assigned ranks) and ranks produced by the ranking function, over all the test queries. Cumulative loss ratio r_{loss} can be regarded as the false alarm rate of the linear classifier. We use r_{loss} to measure the quality of user preferences capture. The smaller r_{loss} is, the more accurate a ranking function is at capturing user preferences.

Figure 7 illustrates the cumulative loss ratio of all three methods for each user. The loss ratios of both LtR and LtR_CA are always better than that of the baseline approach. In general, as shown in Table 4, the average loss ratio of LtR is better than that of LtR_CA. However, we find that LtR is not always better than LtR_CA for all users. This is because some users are not interested in our exclusive features such as property complexity. Instead, they are more interested in the common features of our method and Aleman-Meza *et al*. (2005), *e.g.*, context features. For these users, both methods show similar loss ratio because the same learning-to-rank algorithm is applied.

Figure 7: Cumulative loss ratio for each of the 20 users. This figure indicates that our method (LtR) and LtR_CA always performs better than the baseline in capturing user preferences.



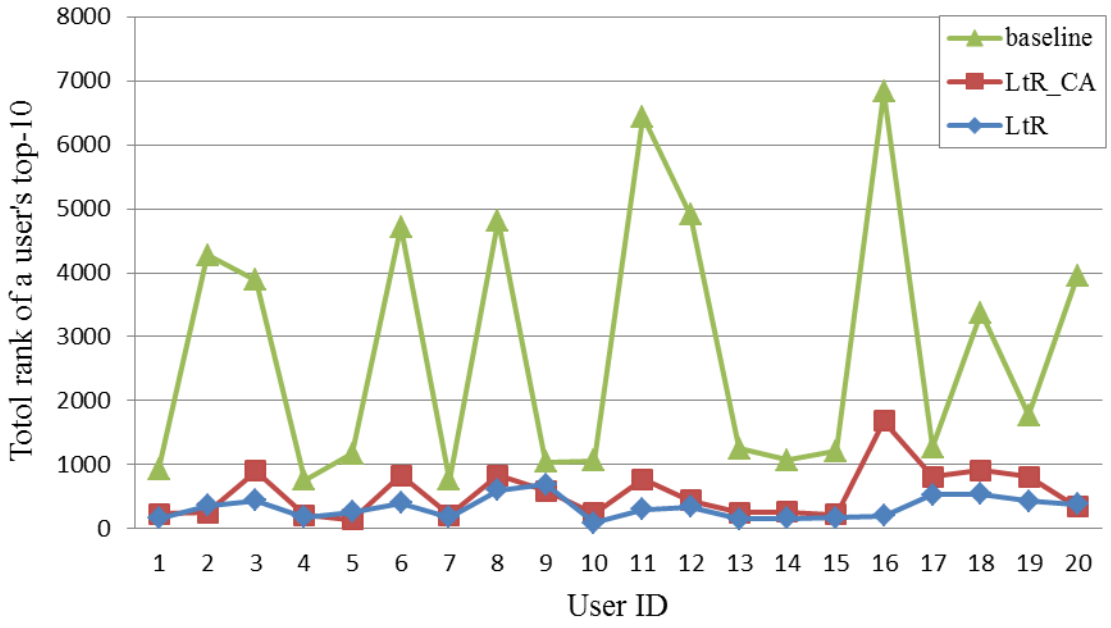
3. Total rank of a user's top-10

$$\mathsf{S} = \sum_{s \in U} rank(s) \tag{13}$$

where U denotes a set that contains a user's top 10 favorite semantic associations search results for a given query. This measurement is the number of results a user needs to examine in order to retrieve all of his 10 favorite results. It assesses the effectiveness of a ranking approach in identifying a user's most interested results. The lower S is, the more effective the ranking function is.

Figure 8 illustrates the total ranks of 20 individual users' top 10 favorite results. Our method is always more effective in capturing a user's top 10 favorite results than the baseline, and demonstrates better effectiveness to LtR_CA for most users.

Figure 8: Total rank of each user's top 10 favorite results: before having seen all of his 10 most favorite results, the number of results a user needs to examine in our method (LtR) is far less than that in the baseline.



Comparison 2: Overall Ranking Quality

In addition to the evaluation on an individual user basis, we also evaluate the overall ranking quality of the three methods. We adopt two standard information retrieval metrics: precision@ k and $nDCG_k$ (Järvelin & Kekäläinen, 2000). In order to make quantitative evaluation based on these metrics, we randomly create 20 additional queries. Each user is asked to judge the top 10 records that each of the three methods generated for him, by explicitly assign each record a score on a six point scale ranging from 0(Bad) to 5(Perfect). The record is considered to be *relevant* with a label of 3(Good) or better; to be *non-relevant* otherwise. We collect 12,000 records with user judgements, and evaluate the three methods using the following metrics:

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

nDCG $_k$, the *normalized discounted cumulative gain* at position k is

$$nDCG_k = M_k \sum_{i=1}^{k} \frac{2^{r_i} - 1}{\log_2(1+i)}, \tag{14}$$

where M_k is a normalization factor to ensure $nDCG_k$ of a perfect ordering to be 1. $nDCG_k$ is designed specifically for evaluating ranking results. It rewards relevant records ranked at the top more heavily than those ranked lower (Agichtein *et al.*, 2006).

Figure 9 and Figure 10 show that among the three methods, our method (LtR) always has the

best precision and *nDCG* for the top 10 ranking results.

In addition, we compute the averages of *cumulative loss ratio* and *total rank of a user's top 10*, as another two metrics to assess the overall ranking quality. Table 4 contains the evaluation results of all four metrics discussed in this section, showing that our method has the best overall ranking quality under all these metrics.

Figure 9. Precision@k for all three methods

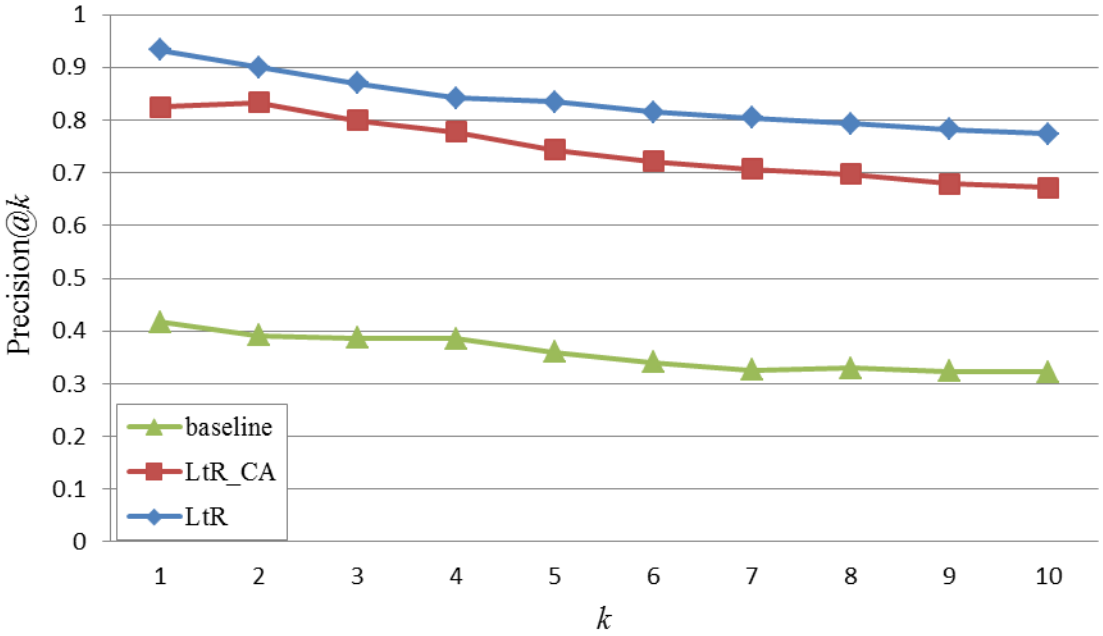


Figure 10. $nDCG_k$ for all three methods

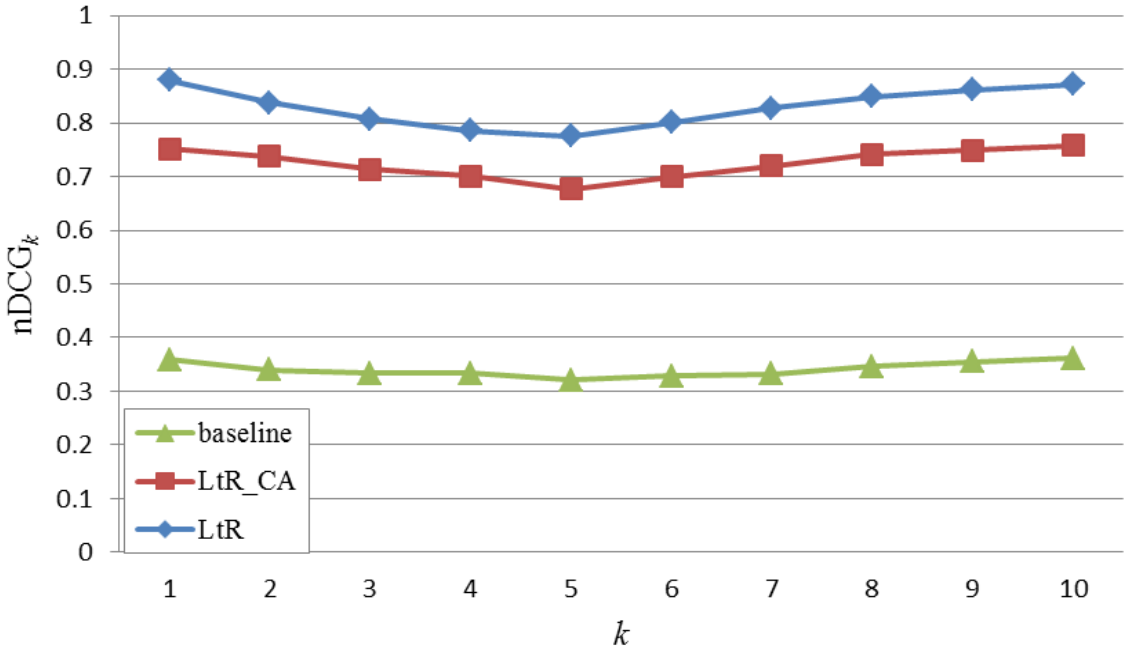


Table 4: Data of average cumulative loss ratio, average total rank of users' top-10, precision@10, and NDCG@10. Our method outperforms the other two methods under all four metrics.

	Avg. loss ratio	Avg. total rank@10	Precision@10	NDCG@10
Baseline	28.91%	2807	32.25%	36.12%
LtR_CA	17.92%	550	67.17%	75.73%
LtR (our method)	**15.44%**	**336**	**77.42%**	**87.17%**

CONCLUSION AND FUTURE WORK

In this paper, we presented a learning-to-rank method to rank semantic association search results. We use a feature vector to characterize each semantic association. A personalized ranking function is automatically created for each user by learning his preferences on these features. We evaluated the per-user and the overall ranking quality of our method under various qualitative and quantitative metrics using a real-world data set. Compared with the-state-of-the-art, our method has demonstrated significant advantage in terms of precision and effectiveness.

Our future work will mainly include three directions. First, we will develop an interactive learning-to-rank system to allow users to continuously improve the ranking results. In particular, the system allows users to provide simple feedback about the current ranking results, learns the users' current preferences from the feedback, and incorporates these preferences into the ranking function. Second, we will use larger training and test sets for the evaluation. Finally, we plan to adopt other machine-learned ranking algorithms to rank semantic associations.

APPENDIX

Table 5. Query set and result statistics

Query	Entity 1	Entity 2	Min. L_A	Max. L_A	# of results
1	Albus Dumbledore	James Potter	2	5	1852
2	Albus Dumbledore	Hermione Granger	2	5	1700
3	Draco Malfoy	Fred Weasley	2	5	2000*
4	Fred Weasley	Lord Voldemort	2	5	2000
5	George Weasley	Fred Weasley	2	5	2000
6	George Weasley	Albus Dumbledore	2	5	2000
7	Ginny Weasley	Cho Chang	2	5	2000
8	Ginny Weasley	George Weasley	2	5	2000
9	Harry Potter	James Potter	1	5	2000
10	Harry Potter	Ginny Weasley	2	5	2000
11	Harry Potter	Lord Voldemort	2	5	2000
12	Harry Potter	Hermione Granger	2	5	2000
13	Harry Potter	Sirius Black	2	5	2000
14	James Potter	Severus Snape	2	5	2000
15	James Potter	Lucius Malfoy	3	5	2000
16	James Potter	Lord Voldemort	2	5	2000

17	James Potter	Lucius Malfoy	3	5	2000
18	Lily Evans Potter	Neville Longbottom	2	5	2000
19	Lord Voldemort	James Potter	2	5	1856
20	Lord Voldemort	Ginny Weasley	2	5	1827
21	Luna Lovegood	Lucius Malfoy	4	6	1603
22	Luna Lovegood	Sirius Black	4	6	2000
23	Luna Lovegood	Fred Weasley	2	5	1291
24	Remus Lupin	James Potter	2	5	2000
25	Ronald Weasley	Cho Chang	2	5	1912
26	Ronald Weasley	Ginny Weasley	2	5	1832
27	Severus Snape	Ginny Weasley	2	5	1650
28	Sirius Black	Remus Lupin	2	6	1555
29	Tom Riddle Sr.	Lily Evans Potter	3	5	502
30	Tom Riddle Sr.	Remus Lupin	3	5	400

*: In our experiments, we retrieve the first 2000 results with length smaller than 10.

REFERENCES

Freebase Schema Explorer (2011). Retrieved February 22, 2011, from http://schemaviz.freebaseapps.com/?domain=/fictional_universe.

SemDis Project (2011). Retrieved March 10, 2011, from http://lsdis.cs.uga.edu:8080/rankingah/.

Agichtein, E., Brill, E., & Dumais, S. (2006). Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 19-26).

Alani, H., Brewster, C., & Shadbolt, N. (2006). Ranking ontologies with aktiverank. In *Proceedings of the International Semantic Web Conference, ISWC2006* (pp. 5-9).

Aleman-Meza, B., Halaschek, C., Arpinar, I., & Sheth, A. (2003). Context-aware semantic association ranking. In *Semantic Web and Databases Workshop Proceedings* (pp. 33-50).

Aleman-Meza, B., Halaschek-Wiener, C., Arpinar, I. B., Ramakrishnan, C., & Sheth, A. (2005). Ranking complex relationships on the semantic web. *IEEE Internet Computing* (pp. 37-44).

Anyanwu, K., Maduko, A., & Sheth, A. (2005). Semrank: Ranking complex relationship search results on the semantic web. In *Proceedings of the 14th international conference on World Wide Web* (pp. 117-127).

Anyanwu, K., & Sheth, A. (2003). ρ-queries, enabling querying for semanticassociations on the semantic web. In *Proceedings of the 12th International World Wide Web Conference* (pp. 690-699).

Cortes, C., & Vapnik, V. N. (1995). Support-vector networks. *Machine Learning Journal* (pp. 273-297).

Crammer, K., & Singer, Y. (2001). Pranking with ranking. In *Advances in Neural Information Processing Systems* (pp. 641-647).

Ding, L., Pan, R., Finin, T., Joshi, A., Peng, Y., & Kolari, P. (2005). Finding and ranking knowledge on the semantic web. In *Proceedings of the 4th International Semantic Web Conference* (pp. 156-170).

Google (2011). Freebase data dumps. Retrieved February 22, 2011, from http://download.freebase.com/datadumps/.

Halaschek, C., Aleman-Meza, B., Arpinar, I. B., & Sheth, A. (2004). Discovering and ranking semantic associations over a large rdf metabase. In *Proceedings of 30th International Conference on Very Large Data Bases* (pp. 1317-1320).

Harth, A., Kinsella, S., & Decker, S. (2009). Using naming authority to rank data and ontologies for web search. In *Proceedings of the 8th International Semantic Web Conference* (pp. 277-292).

Järvelin, K., & Kekäläinen, J. (2000). Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 41-48).

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 133-142).

Kasneci, G., Suchanek, F. M., Ifrim, G., Ramanath, M., & Weikum, G. (2008). Naga: Searching and ranking knowledge. In *Proceedings of the 24th International Conference on Data Engineering* (pp. 953-962).

Kochut, K. J., & Janik, M. (2007). Sparqler: Extended sparql for semantic association discovery. In *Proceedings of the 4th European Semantic Web Conference* (pp. 145-159).

Liu, T.-Y. (2009). Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* (pp. 225-331).

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report. Stanford InfoLab.

Xia, F., Liu, T.-Y., Wang, J., Zhang, W., & Li, H. (2008). Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning* (pp. 1192-1199).