Study of Query Interface Matching Based on Singular Value Decomposition

Heping GOU[1, a], Yongxia JING [1,b *] and Yuan ZHAO [2,c]

[1] Department of Information Technology, Qiongtai Teachers College, Haikou, China

[2] Computer Experiment Center, Naval Aeronautical and Astronautical University, Yantai, China

[a]gou_he_ping@163.com, [b]a_nice_day@163.com, [c]ascendtop@126.com

Keywords: Deep Web; query interface; interface matching; singular value decomposition

Abstract. Deep Web query interface matching is one of the key technologies to realize rapid Deep Web data query. On the study of query interface representation, feature extraction and weight calculation, the method based on singular value decomposition is presented. It transforms the query interface set into a feature-interface matrix, selects the final classification features and creates a feature vector matrix. In order to reduce the computational overhead of similarity computing between the different query interfaces, the singular value decomposition is adopted to truncate the vector matrix, which will build a low rank matrix to approximate the original feature vector matrix. This method realizes the fast matching between the original query interface and the target query interfaces, improving the efficiency of Deep Web online database query.

Introduction

Deep Web query interface matching and integration is to provide users with a unified access interface on online database, which can map the user queries from different domains to Deep Web online databases, realizing data retrieval in these heterogeneous online databases, for different Deep Web data sources, this is equivalent to the establishment of a common data query interface on them, implementing the faster and comprehensive retrieval. Therefore, the matching of Deep Web query interface is to establish the mapping relationship from the integrated interface attributes to each source interface attributes, which needs to solve the attribute matching between source query interface and target query interface. At present, there are lots of studies about matching on attribute text, such as Similarity Flooding (SF), SMatch, etc[1,2]. Association mining method was adopted to obtain the complex matching of correlation attributes in Reference[3], and priority selection strategy was adopted to get the matching of related attributes in reference [4,5], which obtained collection of related attributes through the clustering and designed a new attribute selection method based on the greedy strategy to remove the related error properties. However, these methods select feature attributes by using property tag text of query interface and seldom concentrate on the semantics of attributes, so the recognition rate is usually not good for complex matches. With the rapid expanding of Deep Web databases, the number of query interface on them becomes larger. It is significant to map the user query to different Deep Web databases quickly and comprehensively.

Deep Web query interface matching is to establish mapping relationship between the original query interface and the target query interface, according to the mapping relationship, the original query can be mapped to query interfaces in different domains, realizing large number of Deep Web online database queries. In order to achieve fast data query, the rapid mapping of query interface is one of the key techniques. Therefore, on the basis of previous research, query interfaces are expressed in XML schema tree structure, and then they are parsed into feature-interface matrix. The information gain and term frequency-inverse document frequency are adopted to implement weight calculation of each feature and convert query interfaces to feature vector matrix, then dimension reduction of the vector matrix is implemented using singular value decomposition algorithm. This method implements feature attribute reduction of the query interface, reduces the computational overhead and improves the efficiency of the query interface matching.

Singular Value Decomposition

Singular value decomposition (SVD) chiefly considers on the semantic association in dataset, data dimension reduction could be achieved based on these method[6].SVD is used to convert the original semantic space to another semantic space, which can resolve the semantic correspondence problem of matrix elements, there are two theorems about SVD:

Theorem 1(singular value decomposition for matrix) Given a data matrix A with m rows and n columns ($A\in R^{m\times n}$), there are m order orthogonal or unitary matrix $U(U\in R^{m\times m})$ and n order orthogonal or unitary matrix $V(V\in R^{n\times n})$,

$$A=U\Sigma V^T(\text{或}U\Sigma V^H) \tag{1}$$

Where $\Sigma=\begin{bmatrix}\Sigma_1 O\\ O\ O\end{bmatrix}$, $\Sigma_1=diag(d_1,d_2,...,d_r)$, the diagonal elements are arranged in the order $d_1\ge d_2\ge...\ge d_r\ge 0, r=rank(A)$, $d_1,d_2,...,d_r$ and $d_{r+1}=d_{r+2}=...=d_n=0$ referred to as the singular values of matrix A.

Singular value decomposition is employed to deal with compression of document vector space, which has two aspects: document compression and feature compression. For $m\times n$ matrix A, where each row represents a document, and each column represents a feature. The following is an explanation of the singular value decomposition applied in text categorization.

1) $n\times n$ matrix V is an orthogonal matrix(or unitary matrix), PostMultiply V by Eq.1 that $AV=U\Sigma$, its column vector is

$$Av_i=\begin{cases}d_iu_i, i=1,2,...,r\\ 0, i=r+1,r+2,...,n\end{cases} \tag{2}$$

So v_i is called the right singular vector of matrix A, V is called right singular matrix of matrix A. The $m\times n$ matrix A is compressed into $m\times r$ matrix U which means the column is compressed or the feature is compressed.

2) $m\times m$ matrix U is an orthogonal matrix(or unitary matrix), PreMultiply U^T by Eq.1 that $U^TA=\Sigma V^T$, its column vector is

$$u^TA=\begin{cases}d_iv_i^T, i=1,2,...,r\\ 0, i=r+1,r+2,...,n\end{cases} \tag{3}$$

So u_i is called the left singular vector of matrix A, U is called left singular matrix of matrix A. The $m\times n$ matrix A is compressed into $r\times n$ matrix V which means the row is compressed or the document is compressed.

Theorem 2 Let the singular value decomposition of matrix A ($A\in R^{m\times n}$) is $A=\sum_{i=1}^{r}d_iu_iv_i^T$, where $r=rank(A)$. If $k<r$ and $A_k=\sum_{i=1}^{k}d_iu_iv_i^T$, that is

$$A_k=(u_1,u_2,...,u_k)\begin{bmatrix}d_1 & & & \\ & d_2 & & \\ & & \cdots & \\ & & & d_k\end{bmatrix}\begin{bmatrix}v_1\\ v_2\\ \cdots\\ v_k\end{bmatrix}=U_k\Sigma_kV_k^T \tag{4}$$

Therefore, the application of SVD to the query interface matching system needs to create the matrix A of query interface feature vector, and truncate A according to Eq.4, use a low rank matrix A_k to approximate the original matrix A, which can decrease the query interface data dimension, reduce the computational overhead of similarity between query interfaces.

Feature Selection for Query Interface

Feature selection for query interface is to extract the keywords from the query interfaces, there are a lot of keywords extraction techniques, such as information gain(IG), mutual information and the chi-square statistic methods, etc. In the IG method, the significance of features depends on how much information they can bring, the more information is expressed, the features are more important. IG is used in our method for feature selection. For query interfaces in different domains, we choose the interface features which can identify different domains as final features, The term frequency–inverse document frequency(TF-IDF) is used to calculate the feature weight, these features have to be given different weight based on the frequency of the occurrence in different domains and the number of interfaces. According to the final features weight, the query interfaces can represent as feature vectors, and then the query interface feature vector matrix is built.

Query Interface Matching Algorithm Based on SVD

With the number of Deep Web databases increasing, the query interfaces on the databases became more complex and multiplex, when the interfaces are stored as feature vector matrix, the dimension of this feature vector matrix is growing, and the computational overhead of matching between interfaces is also increasing rapidly. In order to solve the high computational overhead, it is very useful to reduce the dimension of feature matrix by SVD and decrease the computational overhead between new query interface and other query interfaces, matching between the query interfaces can be achieved rapidly, there are five major steps involved in our method:

Step1：For query interface set I , information gain is adopted to implement feature selection, an interface feature set F was generated as final features .

Step2：Construct feature-interface matrix according to interface feature set F , and then calculate the weight of each feature based on TF-IDF, generate query interface vector matrix A .

Step3：According to Eq.4, the matrix A was decomposed, and a low-rank approximation matrix V_k replaces the original matrix A .

Step4：Estimating the similarity between a query interface vector v_{I_r} and each row vector of matrix V_k . The similarity is computed by

$$S(v_{I_r}, T_i) = \sqrt{\sum_{l=1}^{n} (x_{il} - x_{jl})^2} \tag{5}$$

Where x_{il} and x_{jl} is respectively refers to the lth attribute of v_{I_r} and $T_i (T_i \in V_k)$.

Step5：According to the similarity, the query interface T_r is mapped to the query interface set in a particular domain, and the matching relationship is established.

Table 1 The Matrix A about Query Interface

	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}
Model	0.58	0.58	0	0.58	0	0	0	0	0	0	0	0
Format	0	0	0	0	0.71	0	0	0.71	0	0	0	0
Year	0.58	0.58	0	0.58	0	0	0	0	0	0	0	0
Actor	0	0	0	0	0	0	0	0	0.5	0.5	0.5	0.5
ISBN	0	0	0	0	0	0.71	0	0.71	0	0	0	0
Price	0	0	0	0.5	0.5	0.5	0.5	0	0	0	0	0
Author	0	0	0	0	0	0.58	0.58	0.58	0	0	0	0
Title	0	0	0	0	0	0.38	0.38	0.38	0.38	0.38	0.38	0.38
Director	0	0	0	0	0	0	0	0	0.5	0.5	0.5	0.5
Make	0.5	0.5	0.5	0.5	0	0	0	0	0	0	0	0

In order to verify the effectiveness of the proposed method, we selected the UIUC's BAMM query interface data set. This data set has four domains, 211 query interfaces, we selected 12 query interface texts which include in the three different domains, the distribution is :

Automobiles{T0,T1,T2,T3},Books{T4,T5,T6,T7},Movies{T8,T9,T10,T11}

The query interface feature set F is gotten by feature selection.

F={Model, Format, Year, Actor, ISBN, Price, Author, Title, Director, Make}

Then we need to construct feature-interface matrix A and compute weight of each feature in matrix A. The matrix A with TF-IDF weight is shown in Table 1.

In this analysis, rank of matrix A is 7, so we chose the first seven singular values(k=7), and truncated the original matrix, the left singular matrix U_k and the right singular matrix V_k are respectively shown in Table 2 and Table 3.

Table 2 The U_k of SVD for Matrix A

Model	-0.21	-0.52	0.11	-0.02	-0.09	0.386	-0.1
Format	-0.26	0.081	-0.3	-0.9	0.08	-0.03	-0.17
Year	-0.21	-0.52	0.11	-0.02	-0.09	0.386	-0.1
Actor	-0.28	0.214	0.521	-0.09	0.029	0.019	0.112
ISBN	-0.37	0.131	-0.36	0.063	-0.52	0.13	0.656
Price	-0.35	-0.1	-0.23	0.183	0.82	0.02	0.325
Author	-0.4	0.132	-0.36	0.343	-0.12	-0.09	-0.55
Title	-0.47	0.248	0.159	0.156	-0.05	-0.05	-0.28
Director	-0.28	0.214	0.521	-0.09	0.029	0.019	0.112
Make	-0.2	-0.5	0.107	-0.03	-0.13	-0.82	0.106

Table 3 The V_k of SVD for Matrix A

T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}
-0.19	-0.19	-0.06	-0.29	-0.2	-0.47	-0.32	-0.47	-0.25	-0.25	-0.25	-0.25
-0.5	-0.5	-0.15	-0.53	0.005	0.126	0.071	0.189	0.182	0.182	0.182	0.182
0.121	0.121	0.036	0.043	-0.22	-0.34	-0.18	-0.41	0.389	0.389	0.389	0.389
-0.05	-0.05	-0.02	0.063	-0.65	0.473	0.42	-0.4	-0.04	-0.04	-0.04	-0.04
-0.22	-0.22	-0.08	0.313	0.601	-0.06	0.417	-0.51	0.011	0.011	0.011	0.011
0.086	0.086	-0.97	0.11	-0.02	0.073	-0.15	0.005	0.004	0.004	0.004	0.004
-0.17	-0.17	0.143	0.271	0.118	0.551	-0.7	-0.21	0.021	0.021	0.021	0.021

We select a query interface T_r ={Make,Model,Year,Style} which belongs to Automobiles, execute the query on Deep Web online database which include 12 query interfaces mentioned above. Query processing as follows:

1) Query interface T_r is converted to the word frequency text according to feature set F which obtained in training step.

2) T_r is represented as a query interface vector

v_{I_r} = (0.5773502691896257,0.0,0.5773502691896257, 0.0,0.0,0.0,0.0,0.0,0.0,0.5)

3) Interface feature vector v_{I_r} was converted into v_{I_r}' according to Eq.4.

v_{I_r}'=(-0.192053, -0.5033301, 0.1205235, -0.0468365,-0.2156479, 0.0860665,-0.1674284)

4) The similarity $S(v_{I_r}',T_i)$ between v_{I_r}' and interface $T_i (i = 0,1,...,11)$ in v_k is calculated according to Eq.5.

Table 4 The Similarity between T_r and T_i

	T_0	T_1	T_2	T_3	T_4	T_5
T_r	1.43E-15	1.43E-15	1.1821	0.6508	1.2521	1.451
	T_6	T_7	T_8	T_9	T_{10}	T_{11}
T_r	1.2634	1.41	1.0531	1.0531	1.0531	1.0531

According to interface similarity(distance) showed in Table 4, if the similarity is greater than a given threshold d , in this paper we select the $d = 1$, then the interface will be removed.

$$T_r = \{T_0, T_1, T_3\}$$

Because T_0, T_1, T_3 are belong to Automobiles ,the query interface T_r will be mapped to the Automobiles domain, establishing matching relationship between different query interfaces on query interface set T_0, T_1, T_3, which enables user get data in different Deep Web databases quickly.

Conclusion

Due to increasing number of Deep Web online database, it is difficult to query comprehensive data, in order to achieve rapid data retrieval and comprehensive query, we need to establish the mapping between the source query interface and the target query interfaces, the matching of query interface is one of the key techniques to realize the mapping between different query interfaces. On the basis of the SVD algorithm, the feature vector matrix is decomposed and create a low rank matrix to approximate the original matrix in this paper, which can achieve the dimension reduction of feature vector, and reduce the computational overhead of the similarity between the query interface vectors, which also will in turn reduce the mapping time between query interfaces, realizing the rapid data query in the target Deep Web database.

Acknowledgments

The Project was supported by Natural Science Foundation of Hainan Province, China(No. 20156241) ,Scientific Research Foundation of the Higher Education Institutions of Hainan Province, China(No. Hnky2015-72) and the Research Project of Qiongtai Teachers College(No. qtky201404).

References

[1] Do H H, Rahm E. Matching large schemas: Approaches and evaluation. Information Systems 2007, 32(6),pp,857–885.

[2] Giunchiglia F, Shvaiko P ,Yatskevich M. an algorithm and an implementation of semantic matching. In Proceedings of the European Semantic Web Symposium, LNCS 3053, 2004,pp, 61-75.

[3] Jie Liu, Nianbin Wang, Fujiang Liu. Complex synonymous matchings based on correlation mining. Proc of the Interoperability for Enterprise Software and Applications China(IESA). Washington, DC:IEEE Computer Society,2009,pp,175-179.

[4] Zhongtian He, Jun Hong, Bell David A. Schema matching across query interfaces on the Deep Web. Proc of the 25th British National Conf on Databases(BNCOD), LNCS 5071, 2008,pp,51-62.

[5] Zhongtian He, Jun Hong, Bell David A. A prioritized collective selection strategy for schema matching across query interfaces. LNCS 5588:Proc of the 26th British National Conf on Databases (BNCOD).Berlin:Springer, 2009,pp,21-32.

[6] Xianda Zhang. Matrix analysis and applications. Beijing: Tsinghua University Press, 2013,pp,288-296.