Volume 5, Issue 4, 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Design and Implementation of Discovering Association Rule for Information Retrieval

[1]Hareesh Kumar G, [2]Sharath Kumara Y, [3]Bhanuprakasha D B
[1]Teaching Asst., Dept of CS &E, JNTUACEP, Pulivendula, Kadapa (Dt), India
[2]Dept. of CS&E, UBDT College of Engineering, Davangere, India
[3]Dept Of CS &E, VTUCPGS, Bengaluru, India

Abstract- Aliases are very important in information retrieval to retrieve complete information about a personal name from the web, as some of the web pages of the person may also be referred by his aliases. The aliases for a personal name are extracted by previously proposed alias extraction method. In information retrieval, the web search engine automatically expands the search query on a person name by tagging his aliases for complete information retrieval thereby improving recall in relation detection task and achieving a significant Mean Reciprocal Rank (MRR) of search engine. For the further substantial improvement on recall and Mean Reciprocal Rank (MRR) from the previously proposed methods, our proposed method will order the aliases based on their associations with the name using the definition of anchor texts-based co-occurrences between name and aliases in order to help the search engine tag the aliases according to the order of associations. The association orders will automatically be discovered by creating an anchor texts-based co-occurrence graph between name and aliases. Ranking SVM will be used to create connections between name and aliases in the graph by performing ranking on anchor texts-based co-occurrence measures. The hop distances between nodes in the graph will lead to have the associations between name and aliases. The hop distances will be found by mining the graph.

Keywords- Alias Extraction, Co-occurrence frequency,, Document Frequency, Hypergeometric Distributions, Log-Likelihood Ratio, Mean Reciprocal Rank, Support Vector Machine,.

I. INTRODUCTION

Information retrieval is the area where users might search for documents, information within documents and metadata from documents on the web. Many users query might include retrieval of documents for personal names. Many celebrities and experts from various fields are referred by their original names on web. Most of the queries to web search engines include person names. For example, people might use "Michel Jackson" as a query on search engine to know about him. The search engine might give the relevant documents met the information need of the user's query. Apparently celebrities and experts might about person names might also be created by aliases.

For example, a newspaper article might refer the persons using their original names, whereas a blogger might refer them using their nick names. The user will not be able to retrieve all information about a person if he only uses his personal name. To retrieve complete information about a person name, one might know about his aliases on the web. Various types of words are used as aliases on the web. Identifying aliases will be helpful in information retrieval. The aliases are extracted using previously proposed alias extraction method. The search engine expands the query on person names by tagging the extracted aliases to retrieve relevant web pages those are referred by original names as well as aliases thereby improving recall and MRR.

A. Existing System

The existing namesake disambiguation algorithm assumes the real name of a person to be given and does not attempt to disambiguate people who are referred only by aliases.

Disadvantages:

1) To low MRR and AP scores on all data sets.
2) To complex hub discounting measure.

B. Introduction To Proposed System

Proposed System focuses on the three problems of discovering association rule.

- How to extract aliases for the given name.
- How to find co-occurrence statistics between name and aliases.
- How to adapt the ranking model effectively and efficiently to rank aliases
- How to utilize mining algorithm to find association order between name and aliases.

The first problem is solved by using alias extraction algorithm.

Algorithm 1: ExtractPatterns(S)

```
comment: S is a set of (NAME, ALIAS) pairs
P ← null
for each (NAME,ALIAS) ∈ S
Do{
     D ← GetSnippets("NAME ∗ ALIAS")
        for each snippet d ∈ D
   P ← P + CreatePattern(d)
 }
Return (p)
```

Algorithm 2: ExtractCandidates(NAME,P)

```
comment: P is the set of patterns
C ← null
for each pattern p ∈ P
Do
{
   D ← GetSnippets("NAME p ∗ ")
          for each snippet d ∈ D
       {
              Do C ← C + GetNgrams(d,NAME, p)
        }
 }
Return (C)
```

Given a set S of (NAME, ALIAS) pairs, the function ExtractPatterns returns a list of lexical patterns that frequently connect names and their aliases in web-snippets. For each (NAME, ALIAS) pair in S, the GetSnippets function downloads snippets from a web search engine for the query "NAME * ALIAS". Then, from each snippet, the CreatePattern function extracts the sequence of words that appear between the name and the alias. Results of our preliminary experiments demonstrated that consideration of words that fall outside the name and the alias in snippets did not improve performance. Finally, the real name and the alias in the snippet are respectively replaced by two variables [NAME] and [ALIAS] to create patterns. For example, from the snippet we extract the pattern [NAME] aka [ALIAS]. We repeat the process described above for the reversed query, "ALIAS * NAME" to extract patterns in which the alias precedes the name.

Once a set of lexical patterns is extracted, we use the patterns to extract candidate aliases for a given name as portrayed . Given a name, NAME and a set, P of lexical patterns, the function ExtractCandidates returns a list of candidate aliases for the name. We associate the given name with each pattern, p in the set of patterns, P and produce queries of the form: "NAME p *". Then the GetSnippets function downloads a set of snippets for the query. Finally, the GetNgrams function extracts continuous sequences of words (n-grams) from the beginning of the part that matches the wildcard operator *. Experimentally, we selected up to 5-grams as candidate aliases. Moreover, we removed candidates that contain only stop words such as a, an, and the. For example, assuming that we retrieved the snippet for the query "Will Smith aka *", the procedure described above extracts the fresh and the fresh prince as candidate aliases.

C. Advantages of Proposed System

I. ***Low cost***

- In this system, the services which ever the user needs will be provided for free of cost.
- Maintenance cost of the application will be less as it uses minimal system resources ex. hardware and software.

II. ***Privacy***

- In proposed system, we overcome the problem of location privacy by implementing authentication.
- New users should register for first time in order to access the application.

III. ***Improved performance***

- Achieving a significant Mean Reciprocal Rank (MRR) of search engine
- Substantial improvement on recall.

II. SYSTEM DESIGN

The proposed method is outlined in Fig. 1 and comprises two main components: pattern extraction, and alias extraction and ranking. Using a seed list of name-alias pairs, we first extract lexical patterns that are frequently used to convey information related to aliases on the web. The extracted patterns are then used to find candidate aliases for a given name. We define various ranking scores using the hyperlink structure on the web and page counts retrieved from a search engine to identify the correct aliases among the extracted candidates

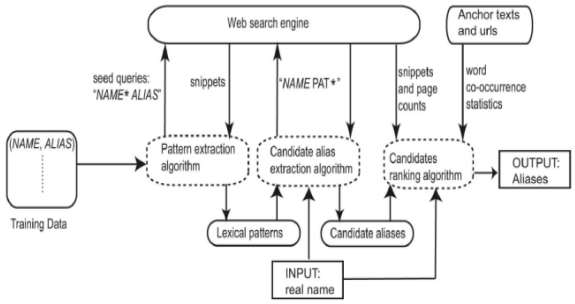


Fig I: Architecture of extracting aliases from the web.

A. Extracting Lexical Patterns from Snippets

Many modern search engines provide a brief text snippet for each search result by selecting the text that appears in the web page in the proximity of the query. Such snippets provide valuable information related to the local context of the query. For names and aliases, snippets convey useful semantic clues that can be used to extract lexical patterns that are frequently used to express aliases of a name. For example, consider the snippet returned by Google2 for the query "Will Smith _ The Fresh Prince." Here, we use the wildcard operator _ to perform a NEAR query and it matches with one or more words in a snippet. In Fig. I the snippet contains aka (i.e., also known as), which indicates the fact that fresh prince is an alias for Will Smith. In addition to aka, numerous clues exist such as nicknamed, alias, real name is nee, which are used on the web to represent aliases of a name. Consequently, we propose the shallow pattern extraction method illustrated in Fig. I to capture the various ways in which information about aliases of names is expressed on the web. Lexico-syntactic patterns have been used in numerous related tasks such as extracting hypernyms and meronyms . Given a set S of (NAME, ALIAS) pairs, the function ExtractPatterns returns a list of lexical patterns that frequently connect names and their aliases in web snippets. For each (NAME, ALIAS) pair in S, the GetSnippets function downloads snippets from a web search engine for the query "NAME _ ALIAS." Then, from each snippet, the Create-Pattern function extracts the sequence of words that appear between the name and the alias. Results of our preliminary experiments demonstrated that consideration of words that fall outside the name and the alias in snippets did not improve performance. Finally, the real name and the alias in the snippet are, respectively, replaced by two variables [NAME] and [ALIAS] to create patterns. Our definition of lexical patterns includes patterns that contain words as well as symbols such as punctuation markers. For example, from the snippet shown in Fig. I, we extract the pattern [NAME], aka [ALIAS]. We repeat the process described above for the reversed query, "ALIAS _ NAME" to extract patterns in which the alias precedes the name. In our experiments, we limit the number of matched words with "_" to a maximum of five words. Because snippets returned by web search engines are very short in length compared to the corresponding source documents, increasing the matching window beyond five words did not produce any additional lexical patterns. Once a set of lexical patterns is extracted, we use the patterns to extract candidate aliases for a given name as portrayed in Fig. 4. Given a name, NAME and a set, P of lexical patterns, the function ExtractCandidates returns a list of candidate aliases for the name. We associate the given name with each pattern, p in the set of patterns, P and produce queries of the form: "NAME p_." Then, the GetSnippets function downloads a set of snippets for the query. Finally, the GetNgrams function extracts continuous sequences of words (n-grams) from the beginning of the part that matches the wildcard operator _ . Experimentally, we selected up to five grams as candidate aliases. Moreover, we removed candidates that contain only stop words such as a, an, and the. For example, assuming that we retrieved the snippet in Fig. I for the query "Will Smith aka_," the procedure described above extracts the fresh and the fresh prince as candidate aliases. For efficiency reasons, we limit the number of snippets downloaded by the function GetSnippets to a maximum of 100 in both Algorithm 1 and 2. In Google it is possible to retrieve 100 snippets by issuing only a single query by setting the search parameter num to 100, thereby reducing the number queries required in practice.

B. Ranking of Candidates

Considering the noise in web snippets, candidates extracted by the shallow lexical patterns might include some invalid aliases. From among these candidates, we must identify those, which are most likely to be correct aliases of a given name. We model this problem of alias recognition as one of ranking candidates with respect to a given name such that the candidates, who are most likely to be correct aliases are assigned a higher rank. First, we define various ranking scores to measure the association between a name and a candidate alias using three different approaches: lexical pattern frequency, word co-occurrences in an anchor text graph, and page counts on the web. Next, we describe the those three approaches in detail**.**

C. Lexical Pattern Frequency

In Section 3.1 we presented an algorithm to extract numerous lexical patterns that are used to describe aliases of a personal name. As we will see later in Section 4, the proposed pattern extraction algorithm can extract a large number of lexical patterns. If the personal name under consideration and a candidate alias occur in many lexical patterns, then it can be considered as a good alias for the personal name. Consequently, we rank a set of candidate aliases in the descending order of the number of different lexical patterns in which they appear with a name. The lexical pattern frequency of an alias is analogous to the document frequency (DF) popularly used in information retrieval.

D. Co-Occurrences in Anchor Texts

Anchor texts have been studied extensively in information retrieval and have been used in various tasks such as synonym extraction, query translation in cross-language information retrieval, and ranking and classification of web pages. Anchor texts are particularly attractive because they not only contain concise texts, but also provide links that can be considered as expressing a citation. We revisit anchor texts to measure the association between a name and its aliases on the web. Anchor texts pointing to a url provide useful semantic clues related to the resource represented by the url. For example, if the majority of inbound anchor texts of a url contain a personal name, it is likely that the remainder of the inbound anchor texts contain information about aliases of the name. Here, we use the term inbound anchor texts to refer the set of anchor texts pointing to the same url. We define a name p and a candidate alias x as co- occurring, if p and x appear in two different inbound anchor texts of a url. Moreover, we define co-occurrence frequency (CF) as the number of different urls in which they co-occur. It is noteworthy that we do not consider co-occurrences of an alias and a name in the same anchor text. For example, consider the Picture of Results VI shows a picture of Dr.Rajkumar being linked to by four different anchor texts. According to our definition of co-occurrence, of Dr.Rajkumar, and fresh prince are considered as co-occurring To measure the strength of association between a name and a candidate alias, using Table 1, we define nine popular co-occurrence statistics: CF, tfidf measure (tfidf), chisquared measure (CS), Log-likelihood Ratio (LLR), hypergeometric distributions (HG), cosine measure (cosine), overlap measure (overlap), and Dice coefficient (Dice). Next, we describe the computation of those association measures in detail.

Flow Chart Diagram for Design and Implementation of Discovering Association Rule for Information Retrieval.

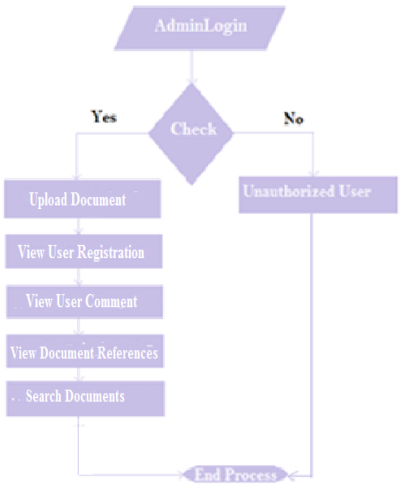


Fig II: Flow chart diagram for admin side.

Operations provided to administrator by application are:

- **Upload document**: Administrator can able to upload or create a new document by entering details like name, tag, etc.., For example, if administrator wants to create new document he/she can enter name as original name, tag as aliases and content of the document then submit these details to create new document.
- **View user registrations**: Administrator can able to view the registered user information contain name, email, phone number and gender user which is given by the user.
- **View user comment**: Administrator can view the comment of the user by entering the document name.
- **View document references**: Administrator can view the number of references to the particular document.
- **Search document**: Administrator can search document by entering document name.

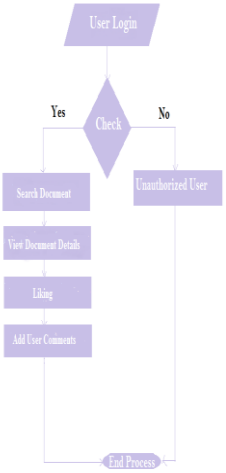


Fig III: Flow chart diagram for user side.

The following operations are going to provide by the application to the user.

- **Search document**: User can search document by entering document name in our implementation image is the document and name of the image is given as input.
- **View document details**: User can viewer details by entering the document name or image name in a search engine. For example if image name is Dr.Rajkumar. User enter name in search engine and he will get all the details of the image.
- **Liking:** User can like the document or image.
- **Add user comment**: The user can also add comment to the specific document.

Use Case diagram for Design and Implementation Discovering Association Rule for Information Retrieval.

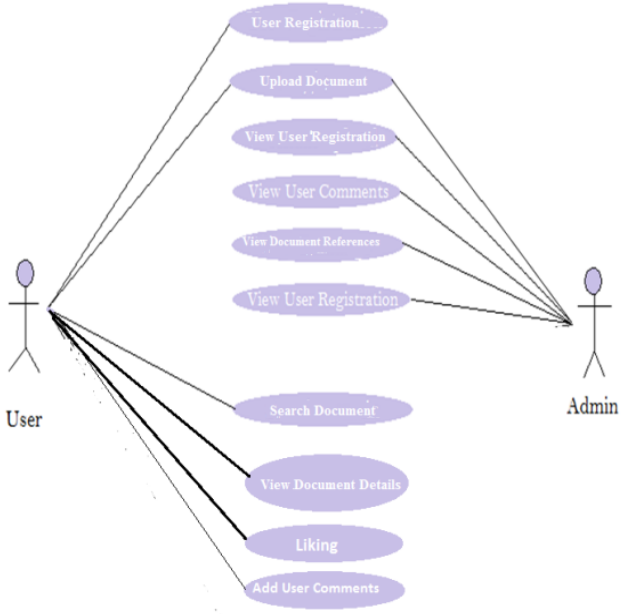


Fig IV: UseCase diagram.

Operations provided to administrator by application are:

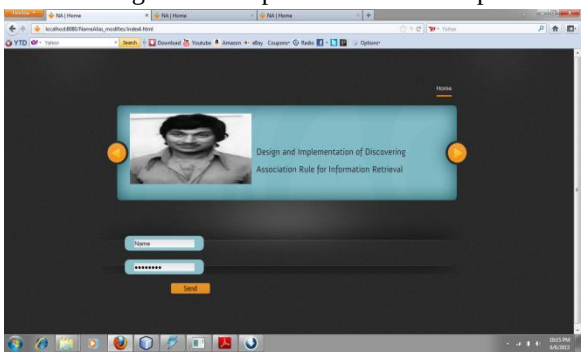
- Upload document.
- View user registration.
- View user comment.
- View document references.
- Search documents.

Operations provided to user by application are:

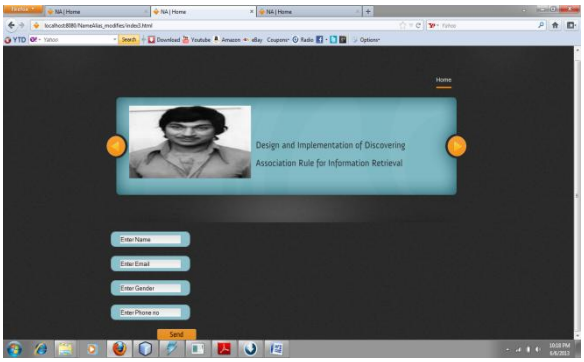
- Search Document.
- View document.
- Liking.
- Add user comment.

III. RESULTS AND DISCUSSION

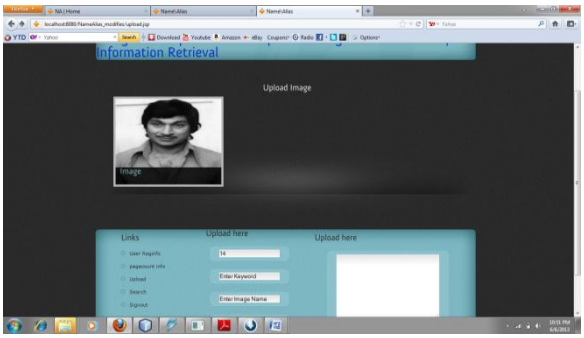
The results and discussions of ranking model adaptation for domain specific search is as follows.



I. Admin Login

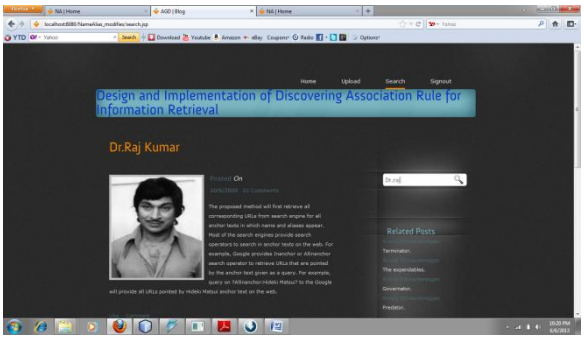


II.User Registration Page

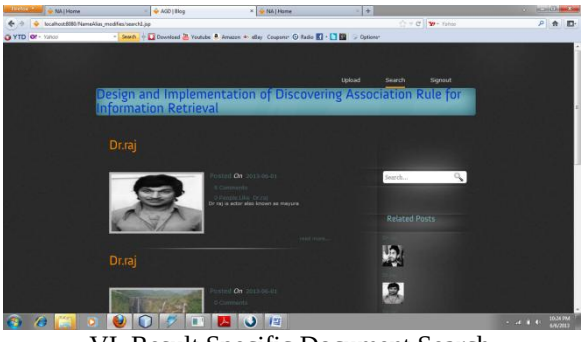


III. Document Upload

C) IV.User Login



V. Searching Document Specifically



VI. Result Specific Document Search

IV. CONCLUSION

The proposed method will compute anchor texts-based co-occurrences among the given personal name and aliases, and will create a word co-occurrence graph by making connections between nodes representing name and aliases in the graph based on their first order associations with each other. The graph mining algorithm to find out the hop distances between nodes will be used to identify the association orders between name and aliases.

Ranking SVM will be used to rank the anchor texts according to the co-occurrence statistics in order to identify the anchor texts in the first order associations. The web search engine can expand the query on a personal name by tagging aliases in the order of their associations with name to retrieve all relevant results thereby improving recall and achieving a substantial MRR compared to that of previously proposed methods

Mining potential information about person identity in emails is one of the popular research topics in email mining. We should focuses on mining name aliases of a user from emails. The methods used in the Alias Authority Ranking Module to rank the authority of name aliases of a user are presented in detail, which are based on email communication relation analysis and morphologically similar alias clustering. Experiment results show that the proposed system can efficiently extract name aliases and find the authoritative aliases of a user.

REFERENCES

[1] Rama Subbu Lakshmi B, Jayabhaduri R *"Automatic Discovery of Association Orders between Name and Aliases from the Web using Anchor Texts-based Co-occurrences"* International Journal of Computer Applications (0975 – 8887) Volume 41– No.19, March 2012.

[2] J. Artiles, J. Gonzalo , and F. Verdejo, *" A Testbed for People Searching Strategies in the WWW"*, 2005.

[3] R. Guha and A. Garg, " Disambiguating People in Search" technical report, Stanford Univ., 2004.

[4] D.Bollegala, Y. Matsuo, and M. Ishizuka , *"Automatic Discovery of Personal Name Aliases from the Web"* IEEE Transactions on Knowledge and Data Engineering, June 2011.

[5] Y. Matsuo, and M. Ishizuka*,"Keyword Extraction from a Single Document using Word Co-occurrence Statistical Information*", International Journal on Artificial Intelligence Tools, 2004.

[6] W. Lu, L. Chien and H. Lee, *"Anchor Text Mining for Translation of Web Queries: A Transitive Translation Approach"* ACM Transactions on Information Systems, Aprill 2004.

[7] Z. Liu, W. Yu, Y. Deng, Y. Wang, and Z. Bian*," A Feature selection Method for Document Clustering based on Part-of-Speech and Word Co-occurrence"* Proceedings of 7th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 10), Aug 2010.

[8] F. Figueiredo, L. Rocha, T. Couto, T. Salles, M.A. Gonclaves, and W. Meira Jr, *"Word Co-occurrence Features for Text Classification"* July 2011.

[9] G. Salton and C. Buckley, *"Term-Weighting Approaches in Automatic Text Retrieval"* Information processing and Management, 1988.

[10] T. Dunning, *"Accurate Methods for the Statistics of Surprise and Coincidence"* Computational Linguistics, 1993.

[11] K. Church and P. Hanks, *"Word Association Norms, Mutual Information and Lexicography"* Computational Linguistics,1991.

[12] T. Hisamitsu and Y. Niwa, *"Topic-Word Selection Based on Combinatorial Probability"* Proc. Natural Language Processing Pacific-Rim Symp, 2001.

[13] F.Smadja*, "Retrieveing Collocations from Text: Xtract"* Computational Liguistics, 1993.

[14] T. Joachims, *" Optimizing Search Engines using Clickthrough Data"* proc. ACM SIGKDD '02, 2002.

[15] D. Chakrabarti and C. Faloutsos , *"Graph Mining: Laws, Generators, and Algorithms"* ACM Computing Surveys, Vol. 38, March 2006, Article 2.

[16] C.C. Agarwal and H. Wang, *" Graph Data Management and Mining : A Survey of Algorithms and Applications"* DOI 10.1007/978-1-4419-6045-0_2,@ Springler Science+Business Media, LLC 2010.

