Augmenting Recommendation Systems Using a Model of Semantically-related Terms Extracted from User Behavior

Khalifeh AlJadda*, Mohammed Korayem**, Camilo Ortiz,
Chris Russell, David Bernal, Lamar Payson,
Scott Brown, and Trey Grainger

Search Relevancy and Recommendations, CareerBuilder
*Department of Computer Science, University of Georgia, GA
**School of Informatics & Computing, Indiana University, IN

khalifeh.aljadda, mohammed.korayem, camilo.ortiz@careerbuilder.com
chris.russell, david.bernal, lamar.payson@careerbuilder.com
scott.brown, trey.grainger@careerbuilder.com

ABSTRACT

Common difficulties like the cold-start problem and a lack of sufficient information about users due to their limited interactions have been major challenges for most recommender systems (RS). To overcome these challenges and many similar ones that result in low accuracy (precision and recall) recommendations, we propose a novel system that extracts semantically-related search keywords based on the aggregate behavioral data of many users. These semantically-related search keywords can be used to substantially increase the amount of knowledge about a specific user's interests based upon even a few searches and thus improve the accuracy of the RS. The proposed system is capable of mining aggregate user search logs to discover semantic relationships between key phrases in a manner that is language agnostic, human understandable, and virtually noise-free. These semantically related keywords are obtained by looking at the links between queries of similar users which, we believe, represent a largely untapped source for discovering latent semantic relationships between search terms.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Data Mining

General Terms

Semantic Discovery, Algorithm

1. INTRODUCTION

Recommender systems are widely used these days in many industries like e-commerce, video streaming, and job portals. Recommender systems automate the process of discovering the interests of a user and subsequently what is relevant to his/her needs [5, 6, 7].

CrowdRec 2014 6 Oct. 2014, Foster City, CA, USA

Many companies like Netflix[1], Amazon[2], CareerBuilder[3], etc. depend on recommender systems (RS) to help drive their revenue. For example, Netflix, a movie rental and video streaming web site, offered a prize (known as the Netflix prize) of 1 million dollars in 2006 for any recommendation algorithm that could beat their RS, named Cinematch[2]. Netflix, like many other websites, depends heavily on recommendations in order to keep their customers interested in their service. One of the major challenges for any RS is the cold-start problem [6], which occurs when there is a lack of information linking new users or items such that the RS is unable to determine how those users or items are related and is therefore unable to provide useful recommendations. One possible way to solve this is to perform classification or matching based upon the description (i.e. keywords in the text) of the items which could be recommended, but this approach tends to not perform as well as a collaborative filtering approach which links related items together based upon the collective intelligence of many users.

Instead of directly linking users and items, we propose a system that utilizes the wisdom of the crowd. This system incorporates a language-agnostic technique for discovering the intent of user searches by revealing the latent semantic relationships between the terms and phrases within keyword queries. Our hope is to express these relationships in common human language so that we can automatically augment recommendations when the only data available from a user is few search keywords extracted from the searches that he conducted. Mining the search history of millions of users allows us to discover the relationship between search terms and the most common meaning of each term. Once we know the semantic relationships between terms according to these users, we can then use those relationships to enhance the recommendation features in order to more accurately express the interest of each user.

Our use case for the proposed technique is for the recommendation system CareerBuilder, which operates the largest job board in the U.S. and has an extensive and growing global presence, with millions of job postings, more than 60

[1]http://www.netflix.com
[2]http://www.amazon.com
[3]http://www.careerbuilder.com

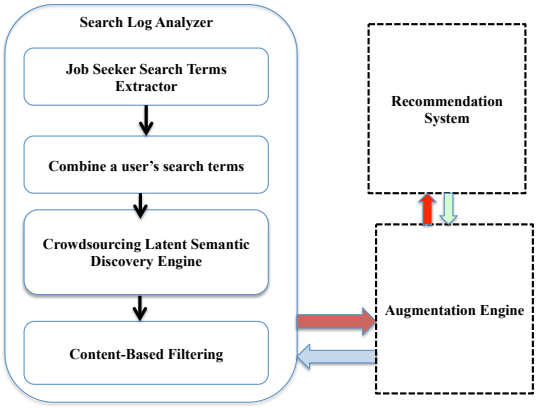

Figure 1: System Architecture

Table 1: Input data to PGMHD

UserID	Classification	Search Terms
user1	Java Developer	Java, Java Developer, C, Software Engineer
user2	Nurse	RN, Rigistered Nurse, Health Care
user3	.NET Developer	C#, ASP, VB, Software Engineer, SE
user4	Java Developer	Java, JEE, Struts, Software Engineer, SE
user5	Health Care	Health Care Rep, HealthCare

million actively-searchable resumes, over one billion searchable documents, and more than a million searches per hour. Our solution to this challenge makes use of the wisdom of the crowd to discover domain-specific relationships. Using the query logs of more than a billion user searches, we can discover the family of related keyword phrases for any particular term or phrase for which a search has been conducted. We are not attempting to fit these terms into an artificial taxonomy, instead we are discovering the existing relationships between terms according to our users.

2. SEARCH LOG ANALYZER

Our proposed system is applicable for websites and services that receive searches from massive numbers of users (i.e. millions), such as is the case with CareerBuilder.com. We propose a *search log analyzer* (SLA) that aims to discover latent semantic relationships among the users' search terms in order to build a semantic dictionary that can more expressively interpret a user's query intent which in turn provide more relevant results in both the search engine and RS. One important feature of the proposed SLA is that it is language agnostic, as there is no dependency on any language-specific natural language processing (NLP) techniques. This makes the system applicable on any website or system that receives a large number of searches, regardless of the list of languages the system supports.

2.1 Crowdsourced Latent Semantic Discovery Engine

The most significant component in the proposed SLA is

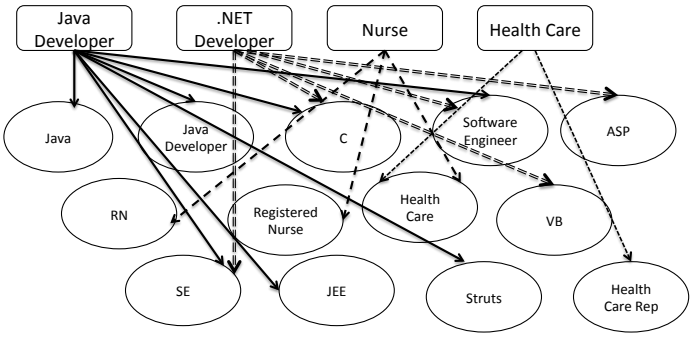

Figure 2: PGMHD representing the search log data

the crowdsourced latent semantic discovery engine. This engine is designed in a way that enables the utilization of crowdsourced wisdom. The model we used to represent the data is a probabilistic graphical model for massive hierarchical data (PGMHD) [1], which we designed and implemented to obtain a scalable variant of a Bayesian Network that is suitable for this kind of massive data. In order to represent search log data in this model, pre-processing phases are required as shown in Figure 1. The pre-processing phases include:

1. classify the users.
2. grab the query strings of their searches.
3. extract the search terms of those query strings.
4. aggregate the search terms of each user.
5. combine the user's classification with his search terms in one table to be used as input table to build the PGMHD.

Table 1 shows the processed data to be represented using PGMHD and Figure 2 shows an example of PGMHD representing search log data. The root nodes in this model represent the classification of the users who conducted searches. Additionally, the second level nodes represent the keywords used in the searches conducted by the users. An edge between a root node (user's classification) and a child node (search keyword) represents the usage of a search keyword by the users from a classification.

To obtain the estimates in our probabilistic model, we store the number of searches f_{ck} for a keyword k by users of class c at every edge that connects $c \rightarrow k$. This way, we can naturally estimate the joint probabilities

$$P(k,c) = \frac{f_{ck}}{\sum_{ij} f_{ij}},$$

and similarly, the conditional probabilities

$$P(k|c) = \frac{f_{ck}}{\sum_{j} f_{cj}}, \quad P(c|k) \hat{=} \frac{f_{ck}}{\sum_{i} f_{ik}},$$

required by the PGMHD. The SLA is implemented using the following technologies: HDFS [8], Map/Reduce jobs [3], Hive [9], and Solr[4].

[4]http://lucene.apache.org/solr

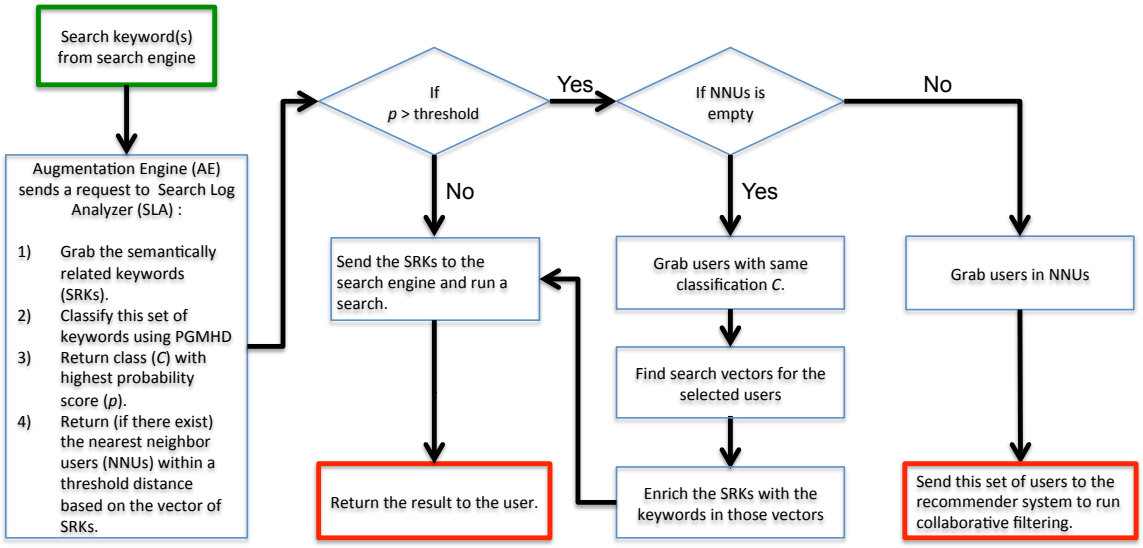

Figure 3: Augmentation Engine Architecture

Table 2: Sample Results using Solr for content-based filtering (CBF)

Term	Before Solr	Removed Terms (Outlier)	Final List
data scientist	machine learning, data analyst, data mining, analytics, big data, statistics, ...	data analyst, data mining, analytics, statistics	machine learning, big data
cashier	retail, retail cashier, customer service, cashiers , receptionist , cashier jobs, teller, ...	receptionist, teller	retail, retail cashier, customer service, cashiers, cashier jobs
collections Rep	collector, collections specialist, call center rep, credit and collections, ...	call center rep	collector, collections specialist, credit and collections
repair tech	repair technician, maintenance technician, call center	maintenance technician, call center	repair technician
front end development	web developer, productivity, Monster	productivity, monster	web developer

2.2 Content-based Filtering

In our content-based post-filtering phase, we apply the search engine [4] itself as part of the SLA for post-filtering. Since our ultimate goal is to find the most related search terms that have actual semantic similarity, we examine all of the relationships for each term. For each relationship we run a query using the original term and another query using the related term. We compare the two resulting document sets and look for intersection. If there are too few intersections we consider the relationship to be invalid and remove it. Table 2 shows how the content-based filtering improves the accuracy of the discovered semantic relationships between search terms.

3. AUGMENTATION ENGINE

There are several approaches to follow in order to improve the accuracy of the RS using the proposed SLA. In our design, the *augmentation engine* (AE) decides which combination of approaches to follow in order to improve the recommendations generated by the RS. Our implementation of the AE focuses on deciding which combination of the following three augmentations (to the RS) should be used:

1. a *search query augmentation*,
2. a *user classification augmentation*, or
3. a *nearest neighbor augmentation*.

These three augmentation approaches try to tackle the special "cold-start" cases when the only information you have about a user are the search queries that he/she performs through the search engine. The next three subsections describe in detail each of the augmentation approaches mentioned above.

3.1 Search Query Augmentation

In the case when the RS is not able to recommend items to a user (possibly due to the "cold-start" problem), it is possible to use the search engine to assist the RS based on the user's search keywords. However, it might be the case that the search keywords from the user are not enough for the search engine to retrieve a significant amount of items to feed to the RS. In such a case, the semantically-related keywords represent a crucial source of information to augment the search query. The *search query augmentation* that we propose uses the semantically-related keywords obtained

from the SLA to enhance the search queries that will be used on the search engine that feeds the RS.

3.2 User Classification Augmentation

In the previous subsection, we presented a simple augmentation approach that only enriches the search query for the search engine with the semantically related search keywords, ignoring any information that can be obtained about the users who share similar searching behavior. To go beyond that basic search query augmentation, we propose to utilize the search keywords to classify the users with the PGMHD model [1] used within the SLA. Given a set of search keywords K and a set C of pre-defined classifications of the users, PGMHD calculates the probability scores p_c for every class of user $c \in C$. The *user classification augmentation* that we propose can be combined with the *search query augmentation* described in the last subsection to refine the decision space in order to return more precise results which fall within the same classification as the user.

3.3 Nearest Neighbor Augmentation

We can further improve the precision of the recommendations obtained from the user classification augmentation presented in the previous subsection by considering a specific subset of the users from the class $c_{\max}$ based on a k–nearest neighbor approach. More specifically, by representing each user as vector of its search keywords, we can define a distance (e.g., Hamming, Euclidean, etc.) between a given user and other users, so as to discover the most similar/nearest ones. The *nearest neighbor augmentation* that we propose enables the RS to use a collaborative filtering (CF) approach in a more precise way by using the most similar users from the class $c_{\max}$.

3.4 Augmentation Engine Architecture

Let K be the set of most relevant search keywords from the user for which the RS will provide recommendations, and C be the set of pre-defined classifications of the users. Then, the proposed AE operates as follows (see Figure 3). First, based on the set of search keyword(s) K, the AE sends a request to the SLA to obtain: i) a set R of semantically related keywords; ii) the probability scores p_c obtained from a PGMHD classification of the keywords, for every class of user $c \in C$; and iii) the set of nearest neighbor users (NNUs) from the class $c_{\max}$ within a threshold distance. If the maximum probability score is larger than a given threshold $\alpha > 0$, i.e.

$$\max_{c \in C} p_c = p_{c_{\max}} > \alpha, \tag{1}$$

and the set of NNUs is not empty, a *nearest neighbor augmentation* is chosen. If (1) is satisfied but the set of NNUs is empty, a *user classification augmentation* is applied (to refine the decision space), and a *search query augmentation* is applied within the decision space corresponding with the user's classification. Otherwise, if (1) is not satisfied, a *search query augmentation* is chosen without restricting the decision space to a specific classification corresponding to the user.

4. EXPERIMENT AND RESULTS

We have implemented the Search Log Analyzer (SLA) using Hadoop Map/Reduce framework. The SLA was applied to analyze 1.6 billion search logs obtained from CareerBuilder.com. After applying the content-based filtering, the discovered related search terms had an error rate of at most 2%. Table 3 shows sample results of the SLA. Currently, we are implementing tests on the RS that incorporate the proposed AE to evaluate the impact of the proposed system at CareerBuilder.com.

Table 3: Results of SLA

Term	Related Terms
hadoop	big data, hadoop developer, OBIEE, Java, Python
registered nurse	rn registered nurse, rn, registered nurse manager, nurse, nursing, director of nursing
data mining	machine learning, data scientist, analytics, business intellegence, statistical analyst
Solr	lucene, hadoop, java
Software Engineer	software developer, programmer, .net developer, web developer, software
big data	nosql, data science, machine learning, hadoop, teradata
Realtor	realtor assistant, real estate, real estate sales, sales, real estate agent
Data Scientist	machine learning, data analyst, data mining, analytics, big data
Plumbing	plumber, plumbing apprentice, plumbing maintenance, plumbing sales, maintenance
Agile	scrum, project manager, agile coach, pmiacp, scrum master

5. CONCLUSIONS

In this paper we propose a novel system to augment recommendations generated by a RS when the only available information about a user for recommendations is a small set of searched keywords. The proposed system relies on a model that discovers the semantic relationships between search keywords using aggregate behavioral data from millions of users. Moreover, the proposed system is language-agnostic and could be integrated with most modern recommendation engines.

6. ACKNOWLEDGMENTS

We would like to greatly thank the Big Data and Data Science teams at CareerBuilder for their support and feedback during implementation of the Search Log Analyzer. Also, many thanks to the Search Development Group at CareerBuilder for their help with integrating the proposed system within the recommendation engine.

7. REFERENCES

[1] K. AlJadda, M. Korayem, C. Ortiz, T. Grainger, J. A. Miller, and W. S. York. Pgmhd: A scalable probabilistic graphical model for massive hierarchical data problems. *CoRR*, abs/1407.5656, 2014.

[2] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[3] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[4] T. Grainger and T. Potter. *Solr in Action*. Manning Publications Co., 2014.

[5] J. A. Konstan. Introduction to recommender systems: Algorithms and evaluation. *ACM Transactions on Information Systems (TOIS)*, 22(1):1–4, 2004.

[6] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *Proceedings of the third ACM conference on Recommender systems*, pages 21–28. ACM, 2009.

[7] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[8] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–10. IEEE, 2010.

[9] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy. Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, 2(2):1626–1629, 2009.