Design of a Recommendation Model Considering Semantic Analysis

Umasankar Das
Silicon Institute of Technology
Bhubaneswar, Odisha

Girija Prasad Mohapatra
Silicon Institute of Technology
Bhubaneswar, Odisha

Vinay Kumar
Wipro Technologies
India

ABSTRACT
The Social networking site is increasingly used as a channel for reaching end users. Personalized Recommender system can work on participatory media content and enhance CMC (computer mediated communication) ultimately providing the user with the finest items of interest. It collects data implicitly as well as explicitly and takes into consideration user activity, preferences, and ratings to evaluate weights for calculation of trust, social intimacy, popularity and semantic scores. The accumulation of these scores generates the final recommendation score and based on it a recommendation list is generated for each user .Several important theories in this regard have proven to be viable and some not so feasible. Thus comparative study of some recommendation systems can throw light on the problems faced and suggest solutions in this regard.

Keywords
Social Networking, Participatory Media, Personalized Recommender system, Semantic, Trust.

1. INTRODUCTION
In the cyber era social networks are exploited and users are inspired to share their own posts or personal information with the other users. There exists a large amount of information in the web containing the texts based blog articles, profiles, pictures, multimedia resources etc .A basic problem that is faced is Information Overload Problem. The redundant data may allude the needy from the desired information. The problem revolves around, how do they deal with information overload problems and how do they effectively retrieve information they consider important?

Hence the above mentioned problem inspires us to develop a personalized recommender system approach and design an information filtering [13][14] mechanism.

However, the design of such systems presents many challenges due to the varied interests of users; personalized recommendations should be provided to them.

2. RELATED WORK
In this section, we explore approaches used to find out semantic nature of recommender systems in various research articles, including Design of a Social Network Based Recommender System for Participatory Media Content, A Synthetical approach for blog recommendation: Combining trust, social relation, and semantic analysis.

Rapid growth of participatory media content such as blogs, videos, and podcasts, requires to build personalized [4][17] recommender systems to recommend only useful content to users. The collaborative filtering model used in A. Seth[1], main goal is to create automated R.S.S Feed for information and recommend appropriate content to appropriate user

The paper tries to focus on news related content and tries to propose the design of a social network based recommender system. It shows the use of sociological theory to simplify the system design, improve scalability, and understand the behavior of system. The paper Outline the system design for Recommender System It also tries to discuss some open problems (like the design of efficient data structures to store large social network graphs, such that clustering and computation operations on these graphs can be executed efficiently) in social network theory.

3. PROPOSED METHOD
The formation of social networks is on the basis of content or topic based networks, means different is the topic different is the network formed having the nodes denoting people interested in the same topic. So the algorithm used for the building of the above mentioned network is clustering algorithm. The model proposed by [1] is based on the context and the topics, it is based on the principle that same words in same context tend to have similar meanings

For retrieving and indexing we have used Latent Semantic Indexing (LSI) [3][5], that uses Singular value decomposition (SVD) to identify patterns present within terms and contents. LSI has a key feature that is, its ability to extract the conceptual content of a body of text by establishing associations between those terms that occur in similar contexts. LSI correlates semantically related terms that are latent in a collection of text. In paper[2][7], Author says that it's very important in the context of blog recommendation that how we introduce interesting, personalized and socially related weblogs of this peer-produced information to bloggers[15][16] through recommendation mechanism. Trust model, social relation and semantic similarity play an important role in trust recommender system, social networking analysis and information retrieval/textual comparison [6], respectively and they are three crucial factors to help prepare the ground for the development of personalized and trustworthy recommendation mechanism. So, for the calculation of trust score rely goes to tidal trust model (algorithm).To retrieve the meaning of users demand we parse text (need) thorough CKIP Chinese word parser algorithm, which plays a vital role in the calculation of semantic score. After the calculation of trust score, semantic score and social intimacy we feed these three scores as the input to the back propagation neural network, to calculate the final recommendation score.

We have proposed a modified system architecture over the system architecture provided in A. Seth[1].The modified system architecture, consider user preference to create user preference list shown in fig 3.a-e

4. SYSTEM ARCHITECTURE

Primarily inspired by Cobra, the blog and RSS-feed aggregation system proposed in [14], the fig:1 shows the proposed architecture of the system.

4.1. Data sources: News or blogging websites

4.2. I/O servers: These are responsible to download new content from the data sources.

4.3. Topic categorizers: Downloaded content analyzed to categorize it into various topics. Document classification schemes such as Latent Semantic Indexing or fast matching of tags and keywords with pre-defined topics may prove useful here.

4.4. Distributed data structure: Data get stored in large clusters of servers in data centers.

4.5. Reflectors: Whenever a new message enters the system, the message and its associated metadata has to be pushed out to various client applications.

4.6. Client application: A client application runs on user devices such as laptops and mobile phones. The application periodically downloads RSS feeds from the reflector corresponding to the cluster of the user, and monitors the user's browsing history to learn the usefulness and credibility.

4.7. User Preference list generator: This covers following:

4.7.1. Preference List generation

4.7.2. Document List Generation

4.7.2.1. Input Representation

4.7.3. Recommendation List generation

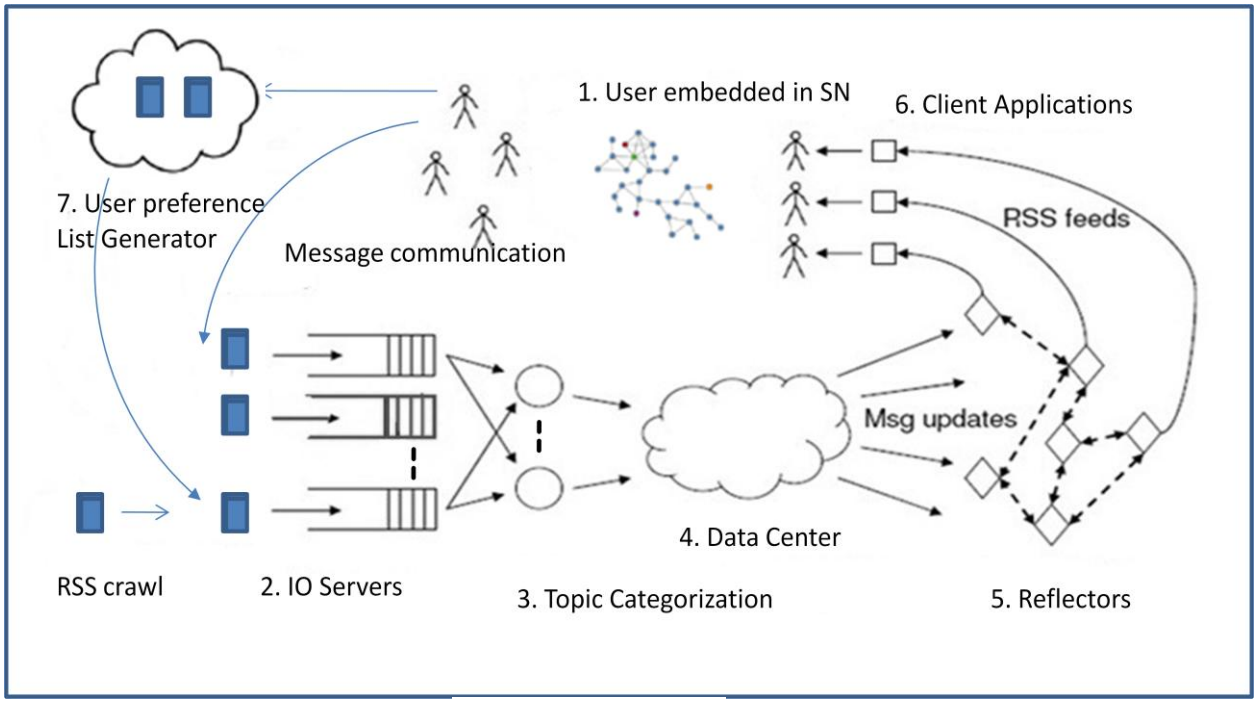

Fig. 1 Proposed Model

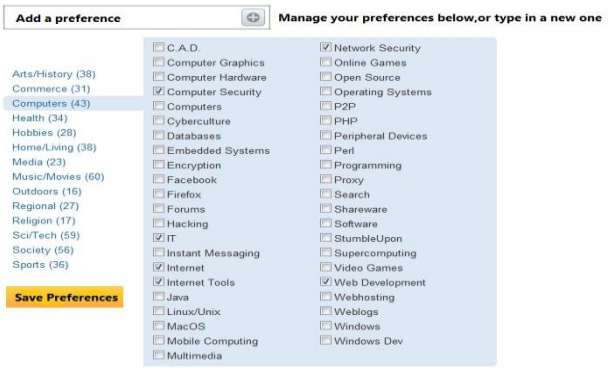

Fig.2 RSS feed Method

RSS feed fig. 2 can also help in collecting user preference. RSS is a way of providing content to the user's browser or desktop in an efficient way. By using RSS feeds, the user can stay updated on the news from news sources and different blogs with little extra effort. User is required to subscribe to particular RSS feed they want. RSS feeds allow users to subscribe to news channel [10] so users are kept up to date with new articles without having to continually check particular news web site. Once user subscribes with the RSS reader of their choice, new articles will automatically appear allowing user to click on those that interest user to read the full story[11][12]. This method works well mainly in the news and blog network. Since this method is used extensively now, and we believe most of the user is aware of this method so we do not elaborate on this method more.

As the objective is to provide a mechanism to identify and sort different web documents to generate a personalized recommendation list so that appropriate document should be recommended to appropriate user, so we need to identify web documents and sort it according to user preference.

We crawl the web to get documents which is shown as format shown in fig. 3.c .We are required to give this document or file and user preference list to SVD as input , which will be

generating recommendation list for a particular user as output. But the format in which we get the document by crawling the web is not suitable for SVD input, so here in section 7.2.1 we present our input representation. If this particular representation is followed then we can parse the file to get another document as shown in fig. 3.e which can serve the purpose of input to SVD.

```
Continent1_contry1_place1_preference1.html: Document1111
Continent1_contry1_place1_preference2.html: Document1112
Continent1_contry1_place1_preference3.html: Document1113
Continent1_contry1_place1_preference4.html: Document1114
Continent1_contry1_place1_preference5.html: Document1115
Continent1_contry1_place2_preference1.html: Document1121
Continent1_contry1_place2_preference2.html: Document1122
Continent1_contry1_place2_preference3.html: Document1123
Continent1_contry1_place2_preference4.html: Document1124
Continent1_contry1_place2_preference5.html: Document1125
Continent1_contry2_place1_preference1.html: Document1211
Continent1_contry2_place1_preference2.html: Document1212
Continent1_contry2_place1_preference3.html: Document1213
Continent1_contry2_place1_preference4.html: Document1214
Continent1_contry2_place1_preference5.html: Document1215
Continent2_contry1_place1_preference1.html: Document2111
Continent2_contry1_place1_preference2.html: Document2112
Continent2_contry1_place1_preference3.html: Document2113
Continent2_contry1_place1_preference4.html: Document2114
Continent2_contry1_place1_preference5.html: Document2115
Continent2_contry2_place2_preference1.html: Document2221
Continent2_contry2_place2_preference2.html: Document2222
Continent2_contry2_place2_preference3.html: Document2223
Continent2_contry2_place2_preference4.html: Document2224
Continent2_contry2_place2_preference5.html: Document2225
Contry3_place2_preference1.html: Document321
Contry3_place2_preference2.html: Document322
Contry3_place2_preference3.html: Document323
Contry3_place2_preference4.html: Document324
Contry3_place2_preference5.html: Document325
                :    :
ContryN_placeM_preference1.html: DocumentNM1
ContryN_placeM_preference2.html: DocumentNM2
ContryN_placeM_preference3.html: DocumentNM3
ContryN_placeM_preference4.html: DocumentNM4
ContryN_placeM_preference5.html: DocumentNM5
```

Fig 3.c Crawled document from web

4.7.2.1. Input Representation

Crawled Document from web cannot serve the purpose of input to SVD, but it can serve the purpose of intermediate input which after getting parsed can serve purpose of input to SVD.

As there can be variation in the representation of web documents, For example in the above fig.3.c we can see at least two variations which is as follows:

```
Continent1_contry1_place1_preference1.html: Document1111
```

```
Contry3_place2_preference5.html: Document325
```

likewise there may be lots of variation in document, which makes it impossible for a computer program (parser) to split the crawled document to get the desired document in the form shown in fig. 3.e which can be given to SVD as input, so we need an input representation that should be followed by developer or web ontology creator while creation of documents. Our input representation is shown in fig. 3.d where we # symbol as separator.

```
Continent1_contry1#_place1#_preference1.html: Document1111
Continent1_contry1#_place1#_preference2.html: Document1112
Continent1_contry1#_place1#_preference3.html: Document1113
Continent1_contry1#_place1#_preference4.html: Document1114
Continent1_contry1#_place1#_preference5.html: Document1115
Continent1_contry1#_place2#_preference1.html: Document1121
Continent1_contry1#_place2#_preference2.html: Document1122
Continent1_contry1#_place2#_preference3.html: Document1123
Continent1_contry1#_place2#_preference4.html: Document1124
Continent1_contry1#_place2#_preference5.html: Document1125
Continent1_contry2#_place1#_preference1.html: Document1211
Continent1_contry2#_place1#_preference2.html: Document1212
Continent1_contry2#_place1#_preference3.html: Document1213
Continent1_contry2#_place1#_preference4.html: Document1214
Continent1_contry2#_place1#_preference5.html: Document1215
Continent2_contry1#_place1#_preference1.html: Document2111
Continent2_contry1#_place1#_preference2.html: Document2112
Continent2_contry1#_place1#_preference3.html: Document2113
Continent2_contry1#_place1#_preference4.html: Document2114
Continent2_contry1#_place1#_preference5.html: Document2115
Continent2_contry2#_place2#_preference1.html: Document2221
Continent2_contry2#_place2#_preference2.html: Document2222
Continent2_contry2#_place2#_preference3.html: Document2223
Continent2_contry2#_place2#_preference4.html: Document2224
Continent2_contry2#_place2#_preference5.html: Document2225
Contry3#_place2#_preference1.html: Document321
Contry3#_place2#_preference2.html: Document322
Contry3#_place2#_preference3.html: Document323
Contry3#_place2#_preference4.html: Document324
Contry3#_place2#_preference5.html: Document325
                :    :
ContryN#_placeM#_preference1.html: DocumentNM1
ContryN#_placeM#_preference2.html: DocumentNM2
ContryN#_placeM#_preference3.html: DocumentNM3
ContryN#_placeM#_preference4.html: DocumentNM4
ContryN#_placeM#_preference5.html: DocumentNM5
```

Fig 3.d Document after input representation

We have written program for parsing and tested it on our representation. It works well and gives output in the desired form.

```
Continent1_contry1    place1    preference1      Document1111
Continent1_contry1    place1    preference2      Document1112
Continent1_contry1    place1    preference3      Document1113
Continent1_contry1    place1    preference4      Document1114
Continent1_contry1    place1    preference5      Document1115
Continent1_contry1    place2    preference1      Document1121
Continent1_contry1    place2    preference2      Document1122
Continent1_contry1    place2    preference3      Document1123
Continent1_contry1    place2    preference4      Document1124
Continent1_contry1    place2    preference5      Document1125
Continent1_contry2    place1    preference1      Document1211
Continent1_contry2    place1    preference2      Document1212
Continent1_contry2    place1    preference3      Document1213
Continent1_contry2    place1    preference4      Document1214
Continent1_contry2    place1    preference5      Document1215
Continent2_contry1    place1    preference1      Document2111
Continent2_contry1    place1    preference2      Document2112
Continent2_contry1    place1    preference3      Document2113
Continent2_contry1    place1    preference4      Document2114
Continent2_contry1    place1    preference5      Document2115
Continent2_contry2    place2    preference1      Document2221
Continent2_contry2    place2    preference2      Document2222
Continent2_contry2    place2    preference3      Document2223
Continent2_contry2    place2    preference4      Document2224
Continent2_contry2    place2    preference5      Document2225
Contry3    place2    preference1      Document321
Contry3    place2    preference2      Document322
Contry3    place2    preference3      Document323
Contry3    place2    preference4      Document324
Contry3    place2    preference5      Document325
                :    :
ContryN    placeM    preference1      DocumentNM1
ContryN    placeM    preference2      DocumentNM2
ContryN    placeM    preference3      DocumentNM3
ContryN    placeM    preference4      DocumentNM4
ContryN    placeM    preference5      DocumentNM5
```

Fig 3.e Parsed document (Input To SVD)

4.7.3 Recommendation List generation

To generate the preference list we used SVD technique. The singular value decomposition was originally developed by differential geometers, who wished to determine whether a real bilinear form could be made equal to another by independent orthogonal transformations of the two spaces it acts on. In linear algebra, the singular value decomposition (a.k.a SVD) is a factorization of a real or complex matrix.A matrix is basically a collection of vectors. Vectors are sequence of numbers corresponding to measurements along various dimensions.

Mathematically, if M is an m×n matrix whose entries come from the field K, which is either the field of real numbers or the field of complex numbers. Then there exists a factorization of the form

$$M=U\sum V^*$$

where U is an m×m unitary matrix over K, the matrix Σ is an m×n diagonal matrix with non-negative real numbers on the diagonal, and V*, an n×n unitary matrix over K, denotes the conjugate transpose of V. Such a factorization is called the singular value decomposition of M.

SVD is very useful in Information Retrieval (IR) to deal with linguistic ambiguity issues. IR works by producing the documents most associated with a set of keywords in a query. A singular value decomposition provides a convenient way for breaking a matrix, which perhaps contains some data we are interested in, into simpler, meaningful pieces. SVD somehow reduces the dimensionality of the dataset and captures the "keywords" that can be used to compare users interest. The first step is to represent the data set as a matrix where the users are rows, documents are columns, and the individual entries are search words. The matrix below is a word * document matrix which shows the number of times a particular word occurs in some made-up documents.

	Doc1	Doc2	Doc3
Abbey	2	3	5
Spinning	1	0	1
Soil	3	4	1
Stunned	2	1	3
Wrath	1	1	4

Table II: Word*document matrix for some made-up documents.

SVD is based on a theorem from linear algebra which says that a rectangular matrix A can be broken down into the product of three matrices - an orthogonal matrix U, a diagonal matrix S, and the transpose of an orthogonal matrix V . The theorem is presented as:

$$A_{mn} = U_{mm}S_{mn}V_{nn}^T$$

where U^TU = I; V^TV = I; the columns of U are orthonormal eigenvectors of AA^T , the columns of V are orthonormal eigenvectors of A^TA, and S is a diagonal matrix containing the square roots of eigenvalues from U or V in descending order. An eigenvector is a nonzero vector that satisfies the equation

$$A\vec{v} = \lambda\vec{v}$$

where A is a square matrix, ¸ λ is a scalar, and v is the eigenvector. ¸ λ is called an eigenvalue. Vectors of unit length that are orthogonal to each other are said to be orthonormal. Two vectors are orthogonal to each other if their inner product equals zero.

The inner product of two vectors (also called the dot product or scalar product) defines multiplication of vectors. It is found by multiplying each component in vector v1 by the component in vector v2 in the same position and adding them all together to yield a scalar value.

Steps

1. Start with finding the transpose of the given matrix.

2.Calculate U= AA^T.

3. Now find the eigenvalues and corresponding eigenvectors of AA^T.

4. Use vector v and lambda and get set of equations.solve them by setting the determinant of the coefficient matrix to zero.

5. Thus generate eigen vector for λ eigen values.These eigenvectors become column vectors in a matrix ordered by the sizeof the eigenvalue. In other words, the eigenvector of the largest eigenvalue is column one, the eigenvector of the next largest eigenvalue is column two, and so forth and so on until we have the eigenvector of the smallest eigenvalue as the last column of our matrix.

6. Finally, we have to convert this matrix into an orthogonal matrix which we do by applying

the Gram-Schmidt orthonormalization process to the column vectors.

Begin by normalizing vector v1.

$$\underset{u_1}{\rightarrow} = \frac{\vec{v_1}}{\overrightarrow{|v_1|}}$$

Compute $\underset{w_2}{\rightarrow} = \underset{v_2}{\rightarrow} * \underset{u_1}{\rightarrow} * \underset{v_2}{\rightarrow} * \underset{u_1}{\rightarrow}$

Normalize $\underset{u_2}{\rightarrow} = \frac{\vec{w_2}}{\overrightarrow{|w_2|}}$

To get the matrix U.

Similarly V is calculated from A^TA using the similar procedure using eigen values and eigen vectors and then by normalization.

Compute V^T also.

7. For S we take the square roots of the non-zero eigenvalues and populate the diagonal with them, putting the largest in s11, the next largest in s22 and so on until the smallest value ends up in smm. The non-zero eigenvalues of U and V are always the same, so that's why it doesn't matter which one we take them from.

Reduced singular value decomposition is the mathematical technique underlying a type of document retrieval and word similarity method variously called Latent Semantic Indexing or Latent Semantic Analysis. The insight underlying the use of SVD for these tasks is that it takes the original data, usually consisting of some variant of a word*document matrix, and breaks it down into linearly independent components. These components are in some sense an abstraction away from the noisy correlations found in the original data to sets of values that best approximate the underlying structure of the dataset along each dimension independently. Because the majority of those components are very small, they can be ignored, resulting in an approximation of the data that contains substantially fewer dimensions than the original. SVD has the added benefit that in the process of dimensionality reduction, the representation of items that share substructure become more similar to each other, and items that were dissimilar to begin with may become more dissimilar as well. In practical terms, this means that documents about a particular topic become more similar even if the exact same words don't appear in all of them.

When we select a particular user u we select his preferences from preference list .These preferences along with the document list is given to SVD, which tells in which document more matches are there, and which document should be added to user recommendation list.

We Give preference list and document list as doc1, doc2, doc3,…docN to SVD as input to calculate semantic score of term with respect to documents. SVD proposes better results than traditional collaborative filtering algorithms.

5. CONCLUSION & FUTURE WORK

We inferred from the comparative study that both the systems lack in providing solution to many open problems and have several challenges. We mainly searched solution for the problem of recommendation list generation i.e. there are so many documents available in the blogosphere and news network, how to search and sort the documents related to user's preference in order to produce final recommendation for the user. After comparative study we found that addition of preference list of the user, can make the recommendation more personalized. The keywords given by the user can act as bullets to fetch the documents, also the order or priority of user can help the system to sort the documents.

We suggested some modification in the system architecture proposed in [1], by the addition of preference list in it. Hence we conclude that adding preference list to the system is a simple and yet an efficient way to create personalized recommendation for a user. We can create user preferences to automate the recommendation process. Considering the semantics [8][9], we will be to establish content categorization which can help achieving a better recommendation process.

6. REFERENCES

[1] A. Seth, "Understanding Participatory Media Using Social Networks," Technical Report, CS-2007-47, U of Waterloo, 2007.

[2] A. Seth and J. Zhang, "A Social Network Based Approach to Personalized Recommendation of Participatory Media Content," ICWSM, 2008.

[3] Berendt, B., & Navigli, R. (2006). Finding your way through blogspace: Using semantics for cross-domain blog analysis. In AAAI symposium on computational approaches to analyzing weblogs. [5] J. Bryant and D. Zillman, "Media Effects: Advances in Theory and Research," Lawrence Erlbaum Associates, USA, 2002.

[4] A. Das, et al, "Google News Personalization: Scalable Online Collaborative Filtering," WWW, 2007.

[5] X. Zhu and S. Gauch, "Incorporating Quality Metrics in Centralized/Distributed Information Retrieval on the World Wide Web," SIGIR, 2000.

[6] J. Kleinberg, "The Small-World Phenomenon: An Algorithmic Perspective," STOC, 2000.

[7] Adar, E., Zhang, L., Adamic, L., & Lukose, R. (2004). Implicit structure and the dynamics of blogspace. In workshop on the weblogging ecosystem: aggregation, analysis and dynamics, WWW.

[8] ALi-Hasan, N., & Adamic, L. (2007). Expressing social relationships on the blog through links and comments. ICWSM.

[9] Golbeck, J. (2006). Trust and nuanced profile similarity in online social networks. Journal of Artificial Intelligence Research.

[10] Adar, E., Zhang, L., Adamic, L., & Lukose, R. (2004).Implicit structure and the dynamics of blogspace. In workshop on the weblogging ecosystem: aggregation, analysis and dynamics, WWW.

[11] ALi-Hasan, N., & Adamic, L. (2007). Expressing social relationships on the blog through links and comments. CWSM.

[12] Berendt, B., & Navigli, R. (2006). Finding your way through blogspace: Using semantics for cross-domain blog analysis. In AAAI symposium on computational approaches to analyzing weblogs.

[13] Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. In proceedings of seventh international World Wide Web conference. Fujimura, K., Inoue, T., & Sugisaki, M. (2005). The EigenRumer algorithm for ranking blogs. In second annual workshop on the weblogging ecosystem: aggregation, analysis and dynamics, WWW.

[14] Golbeck, J., & Hendler, J. (2006). FilmTrust: Movie recommendations using trust in web-based social networks, IEEE Consumer Communications and Networking Conference.

[15] Keinberg, J. M. (1999). Authoritative sources in hyperlinked environment..Proceedings of the ninth annual ACM- SIAM symposium on discrete algorithms, 46(5).

[16] Kolari, P., Finin, T., & Ly ons, K. (2007). On the structure, properties and utility of internal corporate blogs. ICWSM.

[17] Kritikopoulos, A., Sideri, M., & Varlamis, I. (2006). BlogRank: Ranking weblogs based on connectivity and similarity features. in Proceedings of the second international workshop on advanced architectures and algorithms for internet delivery and applications (p. 198).