MARKUP AND VALIDATION AGENTS IN VIJJANA
A Pragmatic Model for Collaborative, Self-organizing, Domain Centric Knowledge Networks[1,2]

S. Devalapalli, R. Reddy, L. Wang and S. Reddy
SIPLab, Department of Computer Science & Electrical Engineering, West Virginia University
Morgantown, WV 26506, U.S.A.

Keywords: Vijjana, Knowledge Network, Semantic Web, Markup, Firefox Browser Extensions.

Abstract: In this paper we describe the Markup and Validation agents in Vijjana, a model for transforming a collection of URLs (Uniform Resource Locators) into a useful knowledge network which reveals the semantic connections between these disparate knowledge units. The markup process is similar to, but much more involved than the traditional book-marking. All the relevant metadata corresponding to a particular Uniform Resource locator is generated and passed on to the organizing agent, which adds this URL to the database. Validation agent checks and ensures the database is consistent and has valid entries.

1 INTRODUCTION

The World Wide Web is the most comprehensive source of information available to almost anyone who has a computer. More often than not, information seekers find what they are looking for on the web. The most popular search engines available today give to the users not just the information requested but additional links and resources to related topics. But the base assumption of all these technologies is "*the user knows what he wants.*" There is a difference between knowing what you want precisely and having only a vague idea of what you are interested in. A computer science student might start out broadly with operating systems and have the following traversal storage systems → memory allocations → multitasking → multitasking in embedded systems → operating systems in embedding systems and end up reading about VxWorks or similar real time operating systems.

To achieve the precise awareness process, Vijjana System provides a series of agents to fulfill the tasks. Very often, a seemingly unguided but a well organized browsing strategy would immensely benefit anybody One of the objectives of Vijjana is to never have an unproductive browsing session. Though this might seem ambiguous, it's easy to see that it can be achieved to an extent.

With the idea of a system like Vijjana mentioned above, conceiving such an application is a two step process – build the library i.e., build a database with data on domains that the system is being built for and browse the system i.e., view/walk through the system in the method described above. In this paper, we will introduce one rudimental but essential part, Vijjana Markup and Validation agent, which simply in a way fits into both categories by applying several recently popular techniques like KEA algorithm to ensure information correctness and also cooperate with other Vijjana agents to achieve high accuracy.

Since building the database is a continuous process, the support for the same is provided as a feature that is referred to as the Markup agent. The purpose of the markup agent is to allow a user to add anything of interest to the database. The markup agent adds this resource to the database and invokes the Organizing Agent which ensures that the resource added will sit at a place that is most appropriate. The markup agent identifies the domain of the document/resource (JAN) being added by generating the key phrases and other meta-data for the contents of the JAN. The purpose of the

[1] This work is supported in part by an endowment from the Bell Atlantic Foundation.
[2] The authors thank all the members of SIPlab for their valuable contributions. The authors are also grateful to Dr. Asesh Das, Dr. Srini Kankanahalli, Sentil Seliah and Ravi Raman for their valuable feedback on earlier drafts of this paper.

Devalapalli S., Reddy R., Wang L. and Reddy S.
MARKUP AND VALIDATION AGENTS IN VIJJANA - A Pragmatic Model for Collaborative, Self-organizing, Domain Centric Knowledge Networks.
DOI: 10.5220/0001825802630269
In *Proceedings of the Fifth International Conference on Web Information Systems and Technologies (WEBIST 2009)*, page
ISBN: 978-989-8111-81-4

validation agent is to remove any dead links i.e., JANs that don't exist anymore and ensure that the database is in a consistent and complete state.

This paper is organized into six parts. A background introduction is given in Section 2 to illustrate our motivation for developing the Vijjana system. Section 3 briefly introduces the framework of our system. From it we can tell the essence of Markup and Validation agent which crosses two agents and play an important role in maintaining information consistence. Detailed agent work process and principle can be found in Section 4. The first part in this section explains a popular key phrases algorithm and its application in Vijjana. And followed is the whole work process. Section 5 gives out A concise conclusion and some future works.

2 MOTIVATION AND RELATED WORK

The main difficulty of using the Web as a knowledge source lies in the fact that the Web is nothing more than a list of hyper-linked pages where the links have no associated semantics. Research on semantic webs is aimed at mitigating this difficulty. Tiwana et al. (2001) and Knoblock et al. (1997) discuss a uniform way to represent web resources and suggest models for automatic integration. Work at IBM on the SHER project (Dolby et al., 2007; Fokoue et al., 2007) focuses on simplifying ontologies and scalable semantic retrieval through summarization and refinement. There has been also considerable research in the Artificial Intelligence community on formalizing knowledge representation (Sowa, 2000; Minsky, 1968; Sowa and Majumdar, 2003) which is being adopted by the researchers in the semantic web community (http://www.semanticweb.org). All of these efforts rely in one form or other on the ability to discover semantic links automatically by analyzing the contents of web pages, which poses considerable difficulties due to the ad hoc nature of web pages. While automatically converting the current web into a fully linked semantic web may be a solution, such an outcome is unlikely in the near future.

Meanwhile a number of organizations such as del.icio.us (http://del.icio.us), Webbrain.com (http://webbrain.com), digg.com (http://digg.com) are busy creating what are called social networking sites where a person searching the web may come across an interesting link that is then "marked" with a set of tags (keywords) which are stored in the site owner's server. A recent start-up company – RadialNetworks has developed a system called Twain (http://twain.com), which claims to create a semantic network automatically. While this may be an advantage for casual social networking it will be unsuitable for enterprise-wide knowledge networks as there are well-established relationships between document types specific to that organization or domain which cannot be derived automatically.

The information created via these sites may be kept private, or it may be combined with similar lists created by other people – thus the name social network. In due course these lists may grow enormously needing the employment of a search engine bringing us back to the original problem - how to cope with a large number of links that cannot be visualized in their semantic context. Current social bookmarking sites do not have any semantic linking of web pages. For a knowledge network to be useful for a large community of users working in a well-defined domain (e.g. Computer Science Teaching), the semantic web should be buildable co-operatively using a predefined taxonomy and link semantics.

With this motivation, we propose a model we call Vijjana (a Sanskrit word that represents collective knowledge created through classification and analysis) which can help in organizing individually discovered web pages drawn from a narrowly bounded domain into a knowledge network. This can be visualized as a hyper tree (http://InXight.com), or a radial graph (http://iv.slis.indiana.edu/sw/), thus making the semantic relations visible. The visibility of semantic relationships is the key to comprehending what is actually inside the knowledge network. It can be perused and also searched by anybody who wants to "discover" knowledge in that domain. Let us consider a simple example where two professors Smith and Bradley among others can contribute useful links to web pages (we call them *Jans* which has roughly the same meaning as the word knol popularized by Google to represent units of knowledge) such as syllabi, homework problems, etc. to an evolving Computer Science specific knowledge network, say Vijjana-CS. These Jans are then classified and interlinked using a pre-defined taxonomy and relational semantics. This Vijjana-CS will grow organically as contributions continue. We can also define a number of agents associated with this model, which can keep the knowledge network complete and consistent by removing missing Jans and associated links. In addition, we can create a

mechanism to maintain usefulness ratings of individual Jans as determined by the users or by using a heuristic based on the number of semantic links and "hits" on the web page.

Now, let us re-visit, Professor Smith and his problem of defining a new course. If Vijjana-CS described above were available, he will simply visit the associated website where he can fully visualize all the textbooks liked to associated syllabi which are further linked to associated lecture notes and homework problems, etc. Furthermore, Professor Smith can quickly notice the brightly glowing nodes (representing most useful Jans as designated by users or by a heuristic based rating agent) and synthesize his course quickly and effectively. Later he may return to Vijjana-CS to contribute his ratings or other Jans he may have created. Thus grows the Vijjana-CS's content and its user/contributor community - collaboratively and by self-organizing. We should also hasten to add that the semantic network created using the Vijjana model could tolerate incomplete links, as we do not apply any formal methods for retrieval of information, but depend on visualization and perusal to guide our discovery of information we need.

3 THE VIJJANA MODEL

We define the Vijjana model as:

Vijjana-X = { J, T, R, dA, oA, cA, vA, sA, rA}

where
X = the domain name
J= the collection of Jans in the Vijjana-X
T = the Taxonomy used for classification of Jans
R= the domain specific relations
dA = the discovery agent which find relevant Jans
oA = the organizing agent which interlinks the Jans based on R
cA = the consistency/completeness agent
vA = the visualization agent
sA = the search agent
rA = the rating agent

The markup agent is a sub-agent of the discovery agent. Similarly, the validation agent is a sub-agent of the consistency/completeness agent. We now examine the underlying concepts followed by the markup process and the validation process.

4 MARKUP AND VALIDATION AGENT

4.1 KEA Algorithm

Key phrases summarize documents. The gist of large documents can be conveniently and coherently described using a set of key phrases that reflect the document's contents. They save a lot of reader's time also. The importance of assigning keywords to documents becomes even more pronounced in a digital library or information retrieval systems. Most of the documents available in electronic form today contain a list of keywords that describe them. However, there exist many that don't. Manually going through them and assigning keywords to each document can be quite a task. It would need several hundred man-hours and people with some knowledge of the subject matter which, considering the use of a computer program to do the task instead is a futile exercise.

The KEA (Witten et al., 1999) algorithm addresses this very need of extracting/assigning keywords to documents. Of course, even at its best KEA might not be able to extract the very same keywords that the author of the document might assign. It still offers a satisfactory solution to the problem of text summarization.

Extraction of key phrases using KEA involves two stages:

1. Training (building a key phrase extraction model): before KEA can extract any key phrases, it must be trained on a set of documents. These documents need to have key phrases assigned to them. KEA builds a model based on these documents.
2. Extracting the keywords: Once a model is built, KEA uses the model to extract key phrases for any new document. Because key phrase extraction is based on this model, it is best to have domain-centric models i.e., training documents need to be related (belong to the same domain/subject) and the documents on which KEA is used with this model must also belong to the same domain for best results.

4.1.1 The Process

Witten et al. (1999) describe the following steps involved in extracting key phrases using KEA starting with cleaning the input document:

1. Cleaning: The document input must be

cleaned first. All punctuation marks, stop words, hyphenations must be removed.

2. Identify the candidate phrases: There is a maximum length set for candidate phrases (5 in Vijjana). All possible subsequences are examined to get a list of candidate phrases.
3. Case folding and stemming: Stemming refers to the process of removing suffixes like *–ed, -es, -ation,* etc. This eliminates redundant considerations of variations of the same phrase.
4. Feature calculation: KEA uses two feature values to compute the probability of a candidate phrase being a key phrase for the document. These are TF×IDF score and the first occurrence.
   TF×IDF score is a measure of a phrase's frequency in the document compared to its frequency in general use. General use here refers to the use of that phrase in the training corpus.
   The TF×IDF score for a phrase P is computed as:

$$\text{TF×IDF} = \frac{freq(P,D)}{size(D)} \times -\log \frac{df(P)}{N} \quad (1)$$

   Where freq(P,D) is the number of times P occurs in D. size(D) is the number of words in D df(P) is the number of documents containing P in the training set
   N is the size of the training set

First occurrence is the fraction of the document that has to be read before the phrase's first appearance. It is the ratio of the number of words that precede the phrase's first occurrence to the total number of words in the document.

With the TF×IDF (t) score and the the first occurrence (d) values, KEA computes the probability of a candidate being a key phrase as follows:

$$P[yes] = \frac{Y}{Y+N} P_{TF\times IDF}[t \mid yes] P_{distance}[d \mid yes] \quad (2)$$

$$P[no] = \frac{N}{Y+N} P_{TF\times IDF}[t \mid no] P_{distance}[d \mid no] \quad (3)$$

$$p = \frac{P[yes]}{P[yes]+P[no]} \quad (4)$$

where Y is the number of positive instances in the training files and N is the number of negative instances i.e., candidate phrases that are not key phrases.

Once the probabilities for all candidate phrases are computed using this method, KEA sorts them and returns the requested number of key phrases.

4.1.2 KEA in Vijjana

An implementation of KEA can be found at (http://www.nzdl.org/Kea/). This implementation has been modified to fit into Vijjana's requirements. The initial version of Vijjana is targeted at people interested in Computer Science. The Vijjana database would contain the JANs that are specific to Computer Science. Keeping this in mind, the training documents were chosen from various areas of Computer Science specifically from areas such as Algorithms, Database Theory, Operating Systems and Pervasive Computing technologies.

A set of 24 documents with author assigned key phrases, covering all the basics of the above mentioned areas was input to KEA.

Since KEA does not always generate key phrases that a reader might expect to associate with the document, Vijjana Markup Agent provides a way to add/remove key phrases.

4.2 Firefox Browser Extensions

Firefox allows application developers to write their own extensions. Extensions, as the name implies extend the functionality of the browser. The extension used in Vijjana is a Toolbar that allows a user to perform operations like Mark-up, Validate JANs, navigate to the Vijjana homepage etc.

Combining these two ideas, Vijjana implements the Markup agent and Validation agents as firefox extensions. On installing the Vijjana extension, a set of toolbar buttons, a menu and a context menu will be installed into the firefox browser. A view of the browser after installing Vijjana extension is shown in Figure 1.

Figure 1: Firefox extension.

4.3 Vijjana Markup Agent

Markup is the process where the user marks up a page/resource of interest to be a new JAN containing feathers like key phrases generated by KEA, and moves it into his/her own collection which is informally called “user space”. When a JAN is moved to the user’s space, it must be placed in its proper corresponding classification of its taxonomy. This is taken care by theVijjana Organizing Agent(VOA) which applies some pattern recognition techniques on key phrases to classify it correctly.

The markup process flow is as shown in Figure 2:

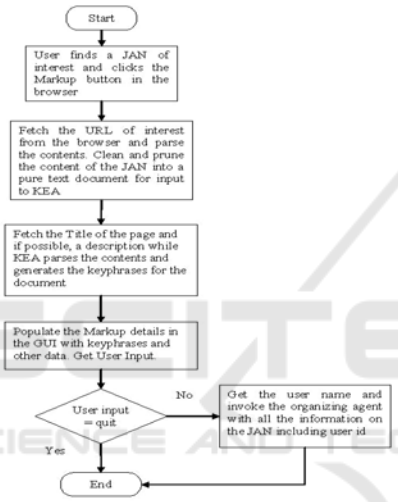

Figure 2: Markup Process Flow.

The Markup process is illustrated in the following series of screenshots. A JAN/URL corresponding to an article on “Ubiquitous Computing” at www.wikipedia.org is illustrated. The user first needs to open the URL in the browser as shown in figure 3.

The next step is to markup this page. This is accomplished by clicking the Markup button in the Vijjana Toolbar, which would trick the JAN creation process. The webpage content will flow into KEA algorithm to generate Key phrases which can be further altered by user. As shown in the Figure 4, the Markup GUI provides the user more options to enrich this JAN, such as adding some semantic features, such as title, descriptions. A default rating of 1 out of 5 is given to the JAN, which also can be altered before adding the JAN. Appropriate predefined categories for the JAN are provided to user to select. As mentioned above, VOA is in charge of providing or preserving these categories.

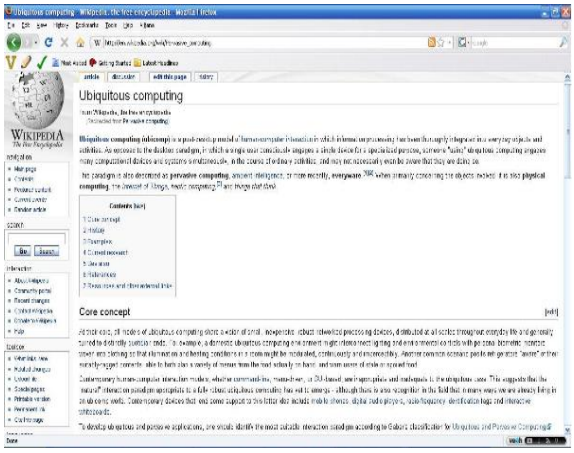

Figure 3: Marking-up a Jan.

The following screenshot (figure 4) illustrates this:

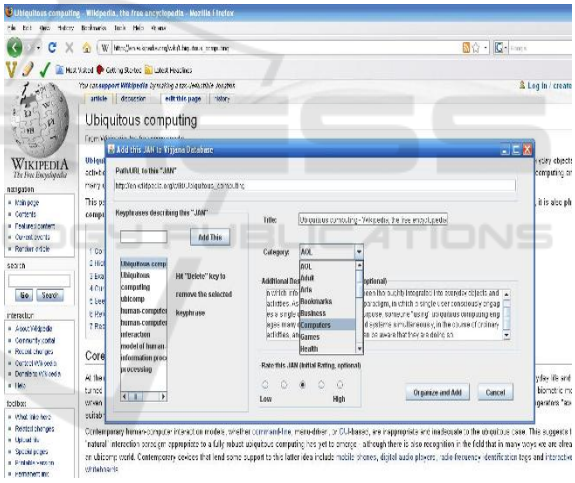

Figure 4: Adding/Deleting Keywords.

To add this JAN to the database, the user clicks on “Organize and Add” button. This invokes VOA to work. The Organizing Agent will then depending on the key phrases add this JAN to its appropriate location in the database. That is all is needed on behalf of the user to add a JAN in his database.

4.4 Vijjana Validation Agent

Validation refers to the process of ensuring that the database is in a consistent state. The next part of this report deals with validation of a user’s JANs. The JANs might be relocated or dead over a period of

time. At any time the database must reflect the most recent status of the JANs. The next button in the Vijjana toolbar is the Validate JANs button The purpose of this button is to ensure that there is no dead links (JANs) in the Vijjana database. In case of such links being present, they need to be removed from the database. It is essentially a cleansing process where the database is cleaned of any inconsistencies.

The most natural way of implementing this feature is to fetch the URLs pertaining to a particular user and validate them one after the other. A major hurdle in doing this is that a user might have too many JANs in his database which would cause the system to run out of memory or take a considerable amount of time to perform the validations. While running out of memory can be corrected by fetching subsets of rows and performing validations on those rows. The size of subset in this implementation has been set to 10000 rows which works fine. However, the problem of time consumption cannot be avoided. The time taken to validate all the JANs is proportional to the number of JANs in a user's space. So a user must be at least informed of this before the validation can begin. Only when a user confirms, the validation will begin. The following flowchart (Figure 5) illustrates the validation process:

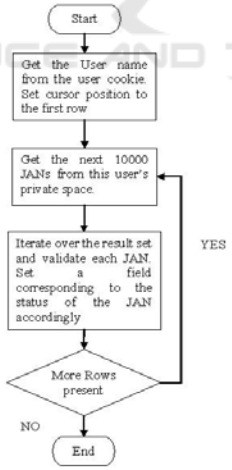

Figure 5: Validation of a Jan.

The following screenshots illustrate the Validation agent in action. If validate has not been performed at all, the valid/invalid state of a JAN is not defined. The column in the database corresponding to this state is termed "InvalidCount". So all those JANs that have not been validated will have the invalidcount column set to null as shown below:

identifierid	entry	invalidcount
1	http://www.liquidgeneration.com/	NULL
2	http://www.onlineanime.org/	NULL
3	http://www.ceres.dti.ne.jp/~nekoi/SENNO/SENfirst.html	NULL
4	http://www.galeon.com/kmh/	NULL
5	http://www.fanworkrecs.com/	NULL
6	http://www.animehouse.com/	NULL
7	http://www2.117.ne.jp/~mb1996ax/enadc.html	NULL
8	http://archive.rhps.org/fritters/yui/index.html	NULL
9	http://www.freecartoonsex.com/	NULL
10	http://www.cutepet.org/	NULL
11	http://www.io.com/~cwagner/jason/	NULL
12	http://www.taremeparadise.com/	NULL
13	http://www.internetdump.com/users/pornographite/index1...	NULL
14	http://darkkaminari.net	NULL
15	http://www.iei.net/~bkos1/velneko.htm	NULL
16	http://www9.kinghost.com/fetish/hentaibee/	NULL
17	http://www.geocities.com/kaseychan17/index.html	NULL
18	http://www.angelfire.com/journal/coldlemonade/index.html	NULL
19	http://www.angelfire.com/anime2/kiyakotari/warning.html	NULL
20	http://www.storyanime.com/	NULL
21	http://e.webring.com/hub?sid=&ring=hentff98&id=&list	NULL
22	http://www.nemurokinenkan.net	NULL
23	http://j-heaven.tripod.com/library.htm	NULL
24	http://trinitycross.net/word/	NULL

Figure 6: Before Validation.

To validate the JANs in a user's space, the user clicks the validate button on the toolbar. A confirmation is required before the validation can begin. The validation process begins and the user is notified when the validation is complete. The next screenshot shows the validation complete confirmation. During the process of validation, it might seem that the browser isn't responding. So validation is best performed overnight as the last task in the day.

The state of the database after the validation process completes looks like

identifierid	entry	invalidcount
1	http://www.liquidgeneration.com/	0
2	http://www.onlineanime.org/	0
3	http://www.ceres.dti.ne.jp/~nekoi/SENNO/SENfirst.html	-1
4	http://www.galeon.com/kmh/	0
5	http://www.fanworkrecs.com/	0
5	http://www.animehouse.com/	0
7	http://www2.117.ne.jp/~mb1996ax/enadc.html	0
3	http://archive.rhps.org/fritters/yui/index.html	-1
3	http://www.freecartoonsex.com/	-1
10	http://www.cutepet.org/	0
11	http://www.io.com/~cwagner/jason/	0
12	http://www.taremeparadise.com/	-1
13	http://www.internetdump.com/users/pornographite/index1...	0
14	http://darkkaminari.net	0
15	http://www.iei.net/~bkos1/velneko.htm	-1
15	http://www9.kinghost.com/fetish/hentaibee/	-1
17	http://www.geocities.com/kaseychan17/index.html	0
13	http://www.angelfire.com/journal/coldlemonade/index.html	0
13	http://www.angelfire.com/anime2/kiyakotari/warning.html	-1
23	http://www.storyanime.com/	0
21	http://e.webring.com/hub?sid=&ring=hentff98&id=&list	0

Figure 7: After Validation.

An invalid count of 0 represents that the link is valid and a value of -1 represents otherwise. The reason we don't delete the JAN from the database if its invalid is that sometimes, the server may be down temporarily for maintenance in which case, validate would assume that the JAN is dead.

5 CONCLUSIONS & FUTURE WORKS

In this paper we have proposed a way to "Markup" URLs which involves extracting metadata of the

URL like the key phrases describing the contents of the URL, the title, a rating and a description of the URL. We have also implemented a validation agent that validates the database (of URLs) and marks the entries as valid/invalid. These agents are currently provided as firefox browser extensions which a user with privileges can install and use.

Although Vijjana Markup and Validation agents already work well in our current system, from above we can say that it still needs some improvements.. Accurate selection of key phrases is an important factor in determining the information correctness. Current KEA algorithm largely depends on domain-specific training data which might not be available before the markup. This calls for a new key algorithm which only marginally relies on domain information. Due to this requirement, we designed a new algorithm called VKE which is currently being evaluated. It combines the syntax heuristic and statistical method together to achieve high accuracy. Also the comparatively large amount of validating time consumed by the validation agent puts considerable demands on the server. To solve this, we are trying to apply some synchronization techniques which divide a large server computing work into many small client tasks, and distribute them along the network.

REFERENCES

Tiwana, Amrit and Ramesh, Balasubramanian (2001) Integrating Knowledge on the Web, IEEE Internet Computing, pp 32-39.

C. Knoblock, S. Minton, J. Ambite, N. Ashish, P. Modi, I. Muslea, A. Philpot, and S. Tejada (1997) Modeling Web Sources for Information Integration, Proceedings of the 1997 AAAI Conference.

J. Dolby, A. Fokoue, A. Kalyanpur, A. Kershenbaum, E. Schonberg , K. Srinivas and L. Mia (2007) Scalable semantic Retrieval Through Summarization and Refinement, Proceedings of AAAI.

Fokoue, A. Kershen baum, L. Mia, E. Schonberg, K. Srinivas (2007) Cutting Ontologies to Size, Lecture Notes in Computer Science, ISBN 978-3-540-49029-6, pp 343-356.

Sowa, John F. (2000) Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks/Cole Publishing Co., Pacific Grove, CA.

Minsky, Marvin, ed. (1968) Semantic Information Processing, MIT Press, Cambridge, MA.

Sowa, John F., & Arun K. Majumdar (2003) Analogical reasoning, in A. de Moor, W. Lex, & B. Ganter, eds., Conceptual Structures for Knowledge Creation and Communication, LNAI 2746, Springer-Verlag, Berlin, pp. 16-36.

http://www.semanticweb.org/

http://del.icio.us

http://webbrain.com

http://digg.com

http://twain.com

http://InXight.com

http://iv.slis.indiana.edu/sw/ - This website has a lot of information on displaying information as a radial graph.

http://www.rssprotocol.com/ - This website has a lot of information about the RSS protocol.

KEA: Practical Automatic Key phrase extraction - Witten I.H., Paynter G.W., Frank E., Gutwin C. and Nevill-Manning C.G. (1999): Proc. DL '99, pp. 254-256.

The KEA project - http://www.nzdl.org/Kea/

Mozilla Developer Center - Building an extension, http://developer.mozilla.org/.

Vijjana - A Pragmatic model for Collaborative, Self-Organizing, Domain Centric Knowledge Networks, Reddy, Dr. Ramana. Morgantown: IKE 2008.