Information Retrieval
Final Report

12/06/2015

Team 6

Computer science 2013210085 Sangho Suh
Industrial engineering 2010170835 Moonshik Kang
Computer science 2010210012 Hoonhee Cho

Contents

I. Introduction

In order to make data visualization not only accessible and relatable but also useful to general users, we plan to make a tool that can be entertaining and useful for general users' daily tasks. Thus, we propose a data visualization tool [1] whose primary function is providing keywords from the Internet news or articles user is currently looking at. It will help simplify the process of rummaging through information presented in the form of text in the Internet. With this tool, users can instantly get a summary of the corresponding Internet news or article by identifying their significant keywords.

The most common data format used in information retrieval is term-document matrix. A document-term matrix or term-document matrix is a mathematical matrix that describes the frequency of terms that occur in a collection of documents. [2] But in our project, we intend to extract keywords from a single web page, not topics of keywords from a corpus. As a result, we had to start with a constraint where we have to extract keywords within sparse text and without using term-document matrix. Our goal was to develop Chrome extension that provides topic or keywords to users for convenience and quick understanding of the news or article. Also, we evaluated the quality of information retrieval through various experiments and evaluation measures.

II. Proposal Recap

So which information can we use for discovering keywords? The most basic is word-count data. By calculating the frequency of appearance in the web page, we can estimate its significance. [3]

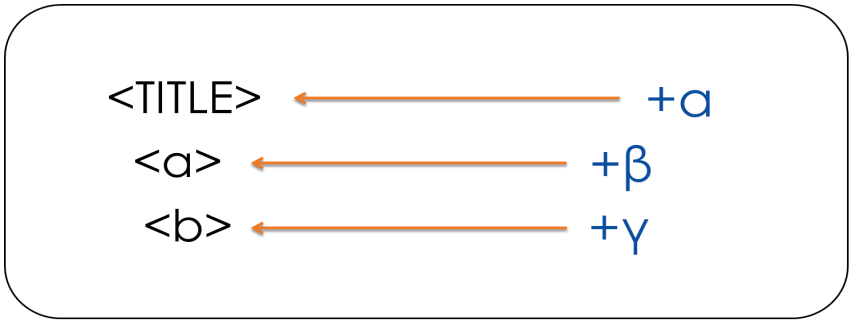

Fig 2-1. Weighting based on HTML tag

The second is Meta tag information. Web page is made out of html tags, and they have relation to the words surrounded by particular tags. For instance, <TITLE> tag usually contains topic of the web page, and it is reasonable to assume that words within it may have keywords representative of the web page. Also, <b> tag imposes 'bold' style on the words within the tag, and since 'bold' is used to imply 'significance', we can use this as well. Based on these, we can apply weighting on words within specific tags and experiment with different weights to find optimal weight, albeit under certain conditions. The sketch of this idea can be seen in Fig. 2-1. The values in Fig. 2-1, i.e., alpha, beta, gamma, are weights that can be experimented under different situations with users to construct application that best extracts keywords.

The third is distance information from the title. Through the tag analysis we conducted, we were able to locate the title of the text. And although other websites may vary to a great extent, we found that Wikipedia provides definition of the queried text in the first sentence of the first paragraph. As a result, we concluded that it is highly likely that words close to title, i.e., words appearing in the first sentence of the first paragraph, are probable candidates of keywords. Building on this concept, we may give greater weight to words close to title or in abstract and less weight on words distant from title.

Aside from these, we also thought about a looping approach, where we use the method above to select 3 high ranked keywords and use extracted keywords gathered by searching with these 3 keywords in Google News as 'feedback' information. Additionally, expanding on the distance weighting idea above, we also thought about utilizing Ontology based similarity to give greater weight to words that are found to have greater similarity in terms of ontology with words within the <TITLE> tag. [5]

Using the methods mentioned above, we need to present the keywords in visually impressive manner, because our goal is to allow users to get quick understanding of the content in the page.

Fig 2-2. Chrome based visualizer example

The image of our application we initially imagined is shown above in Fig. 2-2. While maintaining the order of contents, the size and color of each word is shown differently according to given weights. The words considered ‘significant’ based on its high weight would have big font size and highly vivid color, while words with less ‘significance’ would have small font size and less vivid color. Based on this composition, users will be able to discover important keywords very quickly.

III. Implementation

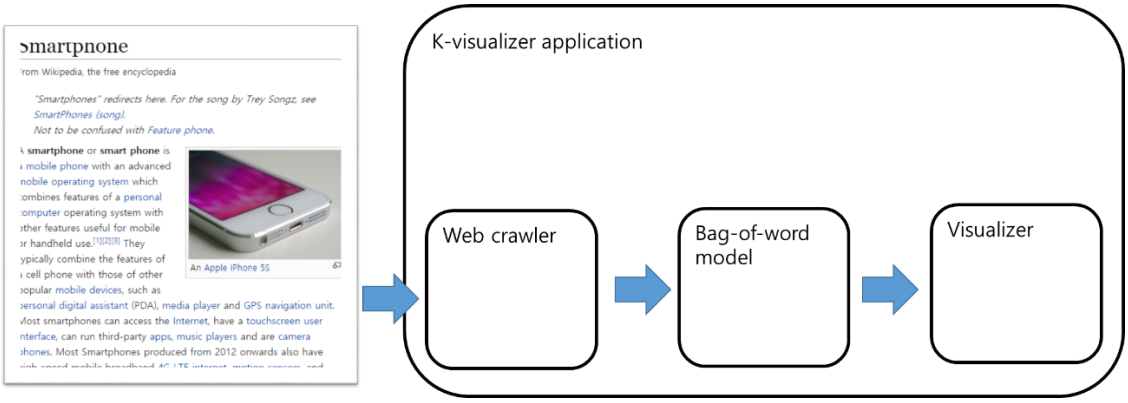

Fig 3-1. Overall architecture

Our k-visualizer application is a chrome extension app that can be easily accessed while users are surfing the web. The main mechanism consists of three parts, Web crawler, bag-of-word generator, and Visualizer. They are implemented by JavaScript, PHP, and HTML, and the detailed architecture is shown in the below diagram.

a. Web page crawler

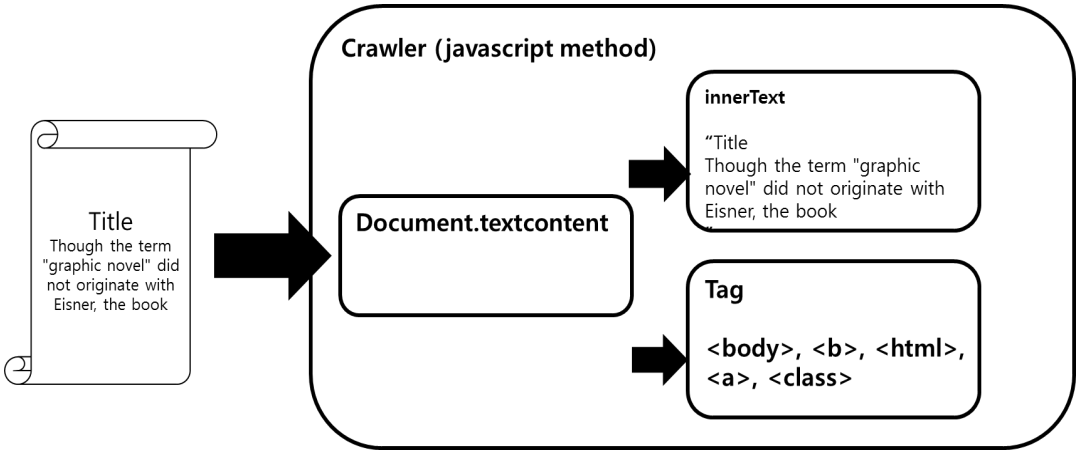

Fig 3-2. Crawling architecture

First, Chrome plugin uses Javascript to extract text as well as HTML tags in the web page. After the plugin crawls all the content within the page, we store HTML tag information and text in separate array.

b. Bag-of-word generator

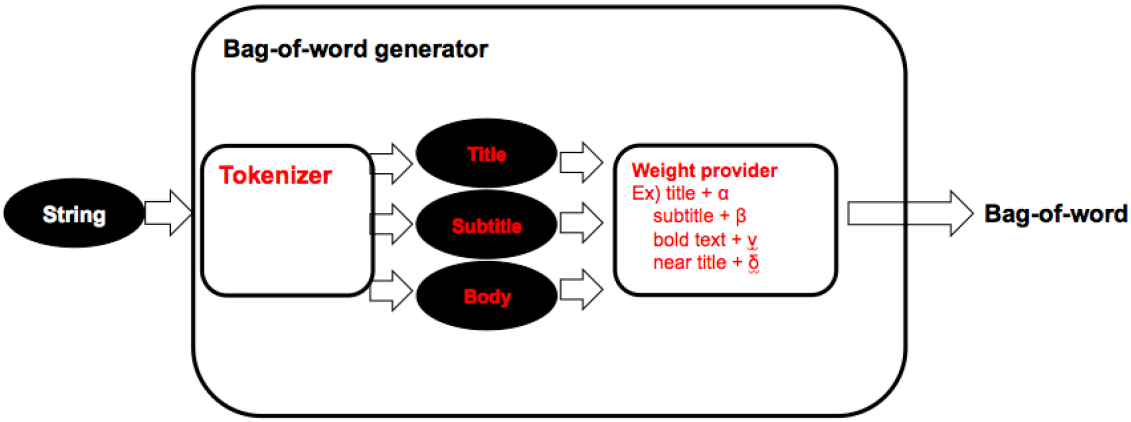

Fig. 3-3. Bag-of-word generator architecture

Bag-of-word generator makes a word-frequency matrix that has tokenized words of the string (crawled string) and its word count. In addition to this, it can give each token customized weight. For example, as you can see in Fig. 2.1, the words that consist of title, subtitle, and bold text get higher weight (50, 30, 20) than normal text (weight is 1 for normal text).

In detail, to explain each component of Bag-of-word generator, we first look at Tokenizer. Its input is crawled texts in string format. Tokenizer splits them into word chunks and outputs them in array, i.e., “wordcount[word]”.

The second component is Weight provider. Among many weighting methods, we use Meta tag weighting, because it takes advantage of the fact that we are crawling from the web. [4] The detailed mechanism we implemented is as follows. Tokenized word chunk in array is decomposed into Title, Subtitle, and Body. After that, each word is given weight of one for one count but varying weight depending on which tag it is associated with.

Also, weight calculation based on distance from title is possible. That is, we assign more weight to words close to title than words further away from title. This may not be true for all web sites, but as we confine the usage of our app in Wikipedia, which always provides definition right after its title, it was considered reasonable for us to impose this weighting method. This method is expressed mathematically as follows:

Weight on words within ε of <title>,

$$\left| x - < title > \right| < \varepsilon$$

where $\left| x - < title > \right|$ is distance from <title> tag

The top 3~5 keywords extracted using this method were shown to be highly correlated with the given page. To further increase usability, we implemented additional feature called ‘Related Google News’. It sends 3 top keywords as query to Google News and then shows the list of returned news with related keywords in a new tab. Original idea we wanted to execute was to conduct additional crawling of the web sites given by Google News and use that data as a feedback to make our Meta tag weighting algorithm more sophisticated and responsive. However, we discovered that Google blocks crawling of its own site. As a result, we resorted to simply providing related Google News, which fortunately turned out to be very useful.

c. Visualizer

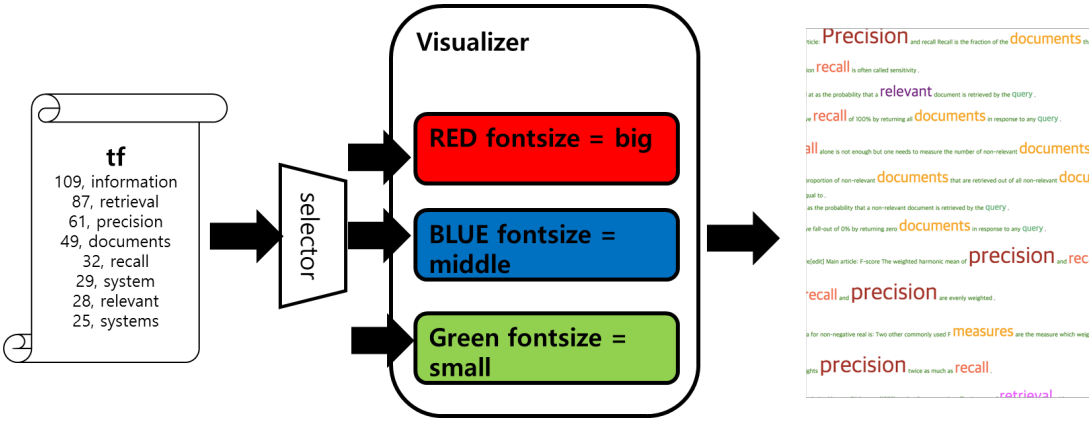

Fig. 3-4. Visualizer architecture

All we have left to do is to show to users in a visual manner. As can be seen in Fig. 4.1, words with different weights are classified by selector into classes of different font size and color. Although diagram shows only 3 classes, we have created 10 different classes for the application. Classified words are then given proper tags and then imprinted on the web page. The result of using our visualizer is as follows.

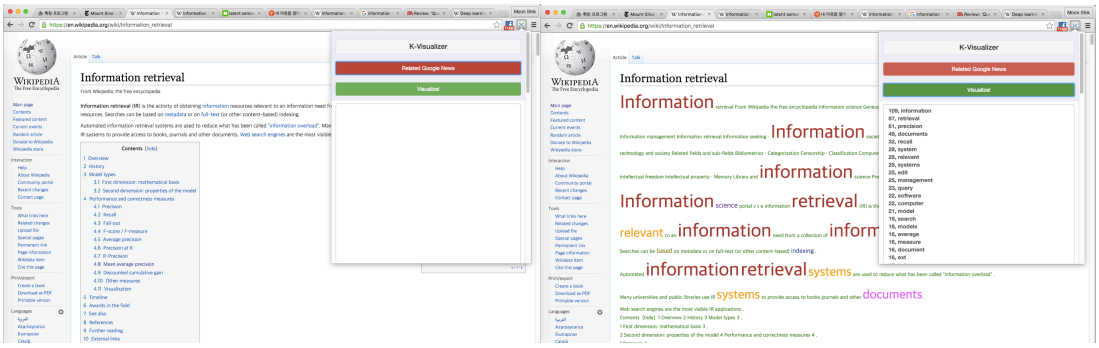

Fig. 3-5. User View

Now, we can set different options for weighting before visualizing. For instance, we may ignore all the Meta tag information and visualize with only word frequency information or add

weight based on distance from title. Although we show visualized image in the web page, we also show users the TF(term-frequency) count result in plugin for further convenience.

The question that becomes increasingly important is which weighting method should we use for certain sites to get optimal keyword result. We experimented with Wikipedia, New York Times, general sites by varying values in weights in order to see under which condition our application performs the best. The details of the experiment are as follows.

IV. Evaluation

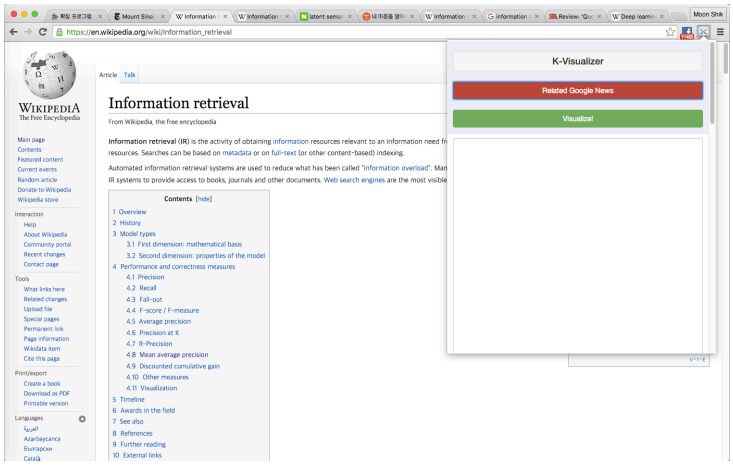

Fig. 4-1. Experiment with 'Information Retrieval' in Wikipedia

We tested with a page retrieved from Wikipedia by using 'information retrieval' as query. For this, we prepared term frequency of words within the page, and in order to increase performance, used weighting mechanism based on Meta tag information and distance from title. For Meta tag based weighting, we imposed additional weights on words associated with <title>, <subtitle>, <b> tags. As for distance based weighting, we gave more weights on words less than or equal to 30 words distant from title. The evaluation of weighting effect is presented below.

The initial experiment was done using just term frequency. And for Meta tag weighting experiment, we used different values, such as ({30,20,15}, {50,20,15}, {30, 12, 10}), on <title>, <subtitle>, and <b> tag. Also, we tested with different distance weighting values (3,5). Last but not least, we tested using combination of Meta tag weighting and distance weighting, e.g., (({30, 12, 10}, {5, 3}). Then, we compared the performance of these 7 cases.

For accurate evaluation of each case, we used Analytic Hierarchy Process (AHP). We put each result as a factor and created 7 x 7 matrix. We then observed the relative advantage of row factor to column factor. Diagonal area in the below diagram is indicative of that. The area below diagonal line is computed as inverse of the area above diagonal, where each element is matched in symmetry. Then, we computed Eigen value of each row from this matrix and L1-normalized it to get weight. It is important to note that in this case, we essentially evaluate all cases in pairwise manner and are able to make more precise evaluation than if we evaluate individually. The result

of this evaluation is shown below (Consistency index value was 0.0048 < 0.1 and thus shown to have consistency).

(1) 가중치 산정 결과

	Factor 01	Factor 02	Factor 03	Factor 04	Factor 05	Factor 06	Factor 07
Weight	**0.129**	**0.131**	**0.127**	**0.167**	**0.133**	**0.136**	**0.176**

(2) 비교 행렬

	Factor 01	Factor 02	Factor 03	Factor 04	Factor 05	Factor 06	Factor 07
Factor 01	1	0.802	1.213	0.711	0.899	1.139	0.702
Factor 02	1.246882793	1	0.918	0.759	0.942	1.049	0.669
Factor 03	0.824402308	1.089324619	1	0.847	0.91	0.857	0.787
Factor 04	1.406469761	1.317523057	1.180637544	1	1.299	1.176	0.961
Factor 05	1.112347052	1.061571125	1.098901099	0.769822941	1	0.882	0.752
Factor 06	0.877963126	0.953288847	1.166861144	0.850340136	1.133786848	1	0.806
Factor 07	1.424501425	1.494768311	1.27064803	1.040582726	1.329787234	1.240694789	1

Fig. 4-2. Analytic Hierarchy Process

Table (1) above shows that factor7 which took Meta tag and distance weighting into consideration has the highest weight. That is, factor7 has received the best evaluation. Also, we can see that factor4 performs well when using weighting combination, {30, 12, 10}. The intuition behind this is as follows. Since <subtitle> and <bold> tend to appear frequently, imposing large weight on <subtitle> and <bold> can result in too much weight. Thus, by imposing large weight on <title> associated words and relatively small weight on <subtitle> and <bold> associated words, factor4 received good feedback. The results of using factor1 (only TF) and factor7 (TF + Meta tag + distance weighting) visualization methods are as follows.

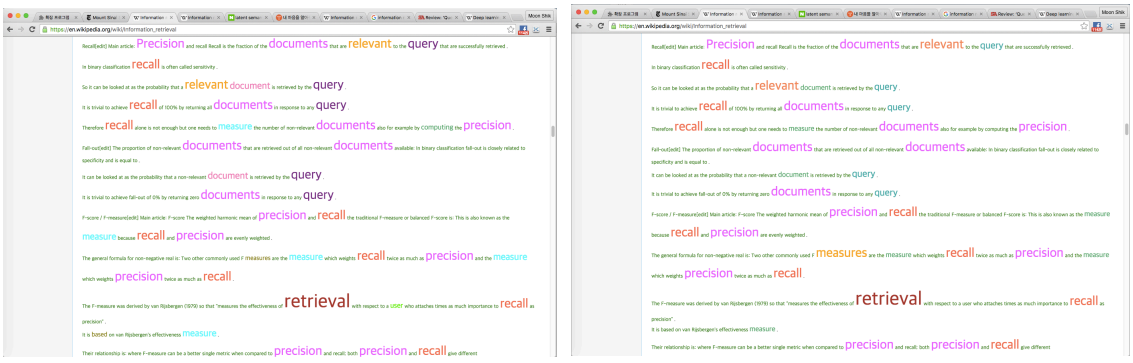

Fig. 4-3. Factor7 vs Factor1

Here, we see that differences become visible by observing words, such as ‘measure’, ‘document’, and ‘query’. Also, we see that indefinite word, such as ‘user’, is taken out.

We used the same experimental approach to test with Wikipedia, New York Times and Nature websites. In the graph below, we show the difference of using just TF, TF + Meta tag weighting, and TF + Meta tag + distance weighting.

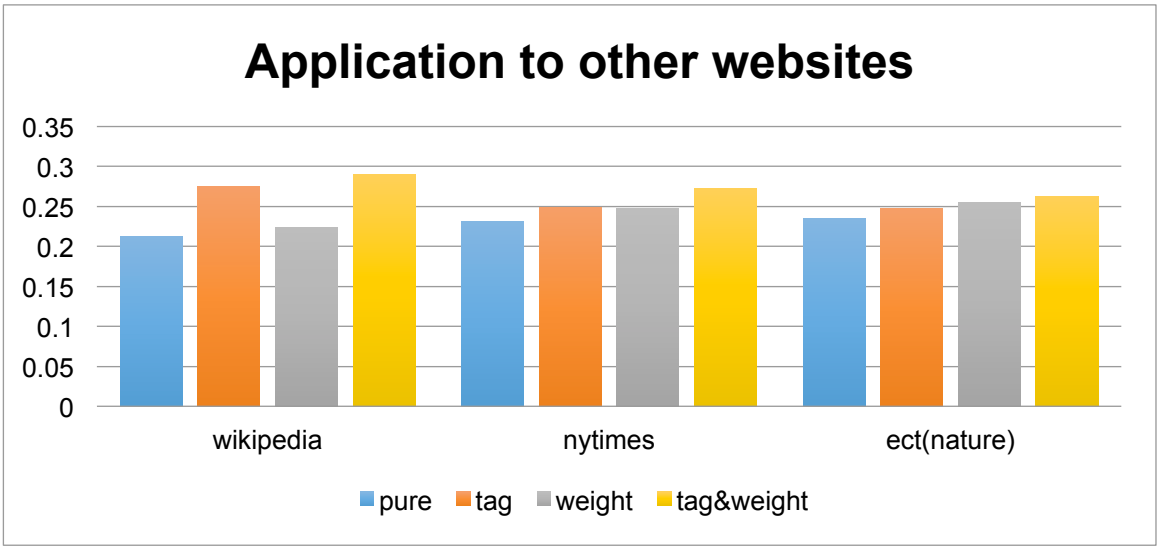

Fig. 4-4. Experiment on Wikipedia, NY Times, Nature

The above graph shows that performance tends to increase when both Meta tag and distance weighting are added to TF approach. However, the benefit of incorporating Meta tag weighting was not as large in non-Wikipedia sites. The reason seems to originate from the fact that other sites do not use 'title' attribute within <a> tag as much as Wikipedia does (cf. Fig 4.5). On the other hand, the benefit of distance weighting seems to be larger in non-Wikipedia sites, and this seems to stem from the fact that by its nature, New York Times and Nature provides important information in the beginning.

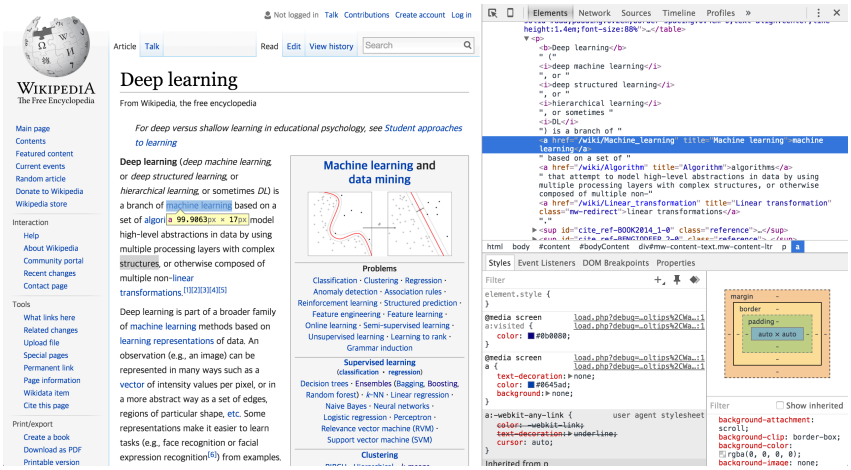

Fig. 4-5. Usage of 'title' attribute of <a> tag in Wikipedia

V. References

[1] Creative Bloq. "The 37 best tools for data visualization". http://www.creativebloq.com/design-tools/data-visualization-712402. Nov 11, 2014

[2] Wikipedia. “Document-term matrix”. https://en.wikipedia.org/wiki/Document-term_matrix. 14 November 2015

[3] Olena Medelyan, Ian H. Witten and David Milne. 2008. “Topic Indexing with Wikipedia”. University of Waikato. pp. 22

[4] H. V. Nguyen, P. Velamuru,D. Kolippakkam, H. Davulcu, H.Liu, and M. Ates. Mining “Hidden Phrase” Definitions from the Web

[5] U.K. Sridevi, N. Nagaveni. Ontology based Similarity Measure in Document Ranking. 2010 International Journal of Computer Applications.
http://disi.unitn.it/~p2p/RelatedWork/Matching/pxc387774.pdf