International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 11 (2015) pp. 28555-28560

http://www.ripublication.com

Content Ranking Using Semantic Word Comparison And Structural String Matching

R. R. Sathiya
Assistant Professor, Department of Computer Science and Engineering, Amrita School of Engineering, Amrita Vishwa Vidyapeetham (University), Coimbatore-641 112. rr_sathiya@cb.amrita.edu

Abstract

Search algorithms today focuses only on ranking based on user history, preferences and traffic to the pages but not necessarily on the content itself. Sometimes the site/page may not be famous as it would not appear in the first set of results which makes users go through many results in order to get what they were looking for since the Content on the site/page is more important than it's ranking. So, to make content appear as first class citizens on the results of a search instead of sites/pages that might contain the query, we're proposing a solution to rank the content based on the query instead of rank only sites/pages to improve the relevance of the results. This would also improve the site/page visibility and increase productivity for users.

Keywords: content; search; ranking; algorithm; content-ranking, semantic-search

Introduction

There are several word_based schematic search methods used such as plagiarism software, content summarization, etc. But there is no such method that measures its words either in a context of a sentence or word flow, word prediction, etc. Measuring a word doesn't mean only in number of letters used, differences in capitalization, compound or complex words but also having to contrast a simple word to a simple context i.e, usage of the word will also be measured.

So lets begin with a question. How to measure a word? (Either within a context or independent)? There are several ways (such as)

- Length
- Similar words
- Synonyms
- Antonyms
- Derivatives

The work has been divided into Query Processor, Search Service, Results Aggregator and Presenter.

Related Works

The problem persists in classifying category based documents. There are many approaches have been proposed to achieve this. Using text analysis method, the terrorism related articles on the web can be detected. For this analysis Keselj's context weight is used [1]. Another approach uses keywords for retrieving relevant documents. There are approaches where WUP uses the WordNet knowledge base to get the semantic relationship between the words [2]. Some other approaches make use of clustering of the retrieved search result records and query rewriting techniques for categorization [3, 4].

Proposed Model

In this paper, we propose a method for comparing letters, words, sentences and calculating their measures and using these measures to rank the results more relevant to the user query.

Query Processor

In this section, a list of words including the query is generated which is to be analyzed. The inputted query is first contextually mapped i.e, meanings and similar words of the query are added to the list. Then alternatives to the query i.e, suggestions or similar query's are added to the list. At last the list is finalized by adding grammatical information, contextual information and is ready to be processed.

Search Service

This section retrieves information from various domains like sports, entertainment, news, history, etc and performs the letter comparison, word comparison and sentence comparison between the generated query list and retrieved results.

Crawler

A list of URLs from each domain which are to be crawled are specified. Sub URLs inside a URL are extracted by identifying "href " tag and are added to the list of URLs to be crawled. Each crawled URL with its content is indexed in a database. Here we are using Raven Database which is a document database to handle large amount of documents. And we are using Reddis Cache to have consistent connection to the database. As the crawling goes on continuously and since there is a chance that the same URL can be parsed again, an additional feature "Last visited " is used to allow an URL to be crawled after a specific time say 6 hrs.

Letter Comparison

The present std code page for computer symbols is Unicode (UTF-8, 16, 32) to display and edit various human interaction language or script on the computer. So, we

are using Unicode (Pr UTF-8/ANSI) and UTF-16, 32 for more advanced comparison of letters. These are already present in .NET framework under System.Globalization, System.Text, System.String, System.char. There are three types of letter comparisons

- Standard
- Ordinal
- Case Sensitive/Insensitive

Standard is the default comparison for letters where it uses both ordinal and case sensitive comparisons. Ordinal uses expanded unicode set where any compressed latin or any other script can be expanded to simpler ones. Example: = ae, ê = ei

Case comparison uses case sensitive rules to compare.

Example : A,a | Â,α

In the case insensitive languages the service yields null or zero value since the language doesn't have casing rules.

For now, the measure of a letter just yields the Unicode code-point value in the UTF table for each bit type (8, 16, 32). Comparison of two chars yields the difference between their measures. Advanced comparison options include phonetic comparison, script comparison and origin language comparison.

Example :

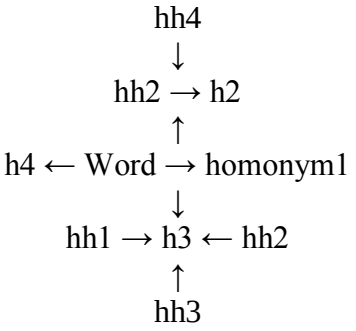
Same language, Same script, Same phonetic Different language, different script

Word Comparison

Measure between words like (Apple, apple), (Runtime, Run Time, Run-Time) have been included.

1. Include/Ignore semantic capitalization
2. Increases the measure of the word by 1 unit (not necessarily equal to 1) 1 unit = 1*Capitalization/Ordinal weight
3. Include/Ignore compound words. Include = (Run)+(Time), (Run)+(-)+(Time). Ignore = Excludes the second word which leads to the same meaning as that of the first word.
4. Parts of speech options - noun, verb, prefixes, prepositions.

Other properties of that particular language - heteronym, homonym, Antonym. Each property of any particular language has weight-age. Weight-age depends on the number of properties associated with itself. For example, any word having 2-3 homonyms which in turn the homonyms have direct relationship with other homonyms.



So the measure of each property of a word in any language is equal to the weighted average of all the child property that relates to the same property itself.

$$P_1[W_{11},W_{12},W_{13}]$$

$$\uparrow$$

$$P_4[W_{41},W_{42},W_{43}] \leftarrow \text{Word} \rightarrow P_2[W_{21},W_{22},W_{23}]$$

$$\downarrow$$

$$P_3[W_{31},W_{32},W_{33}]$$

$W_{21} = P_2[W_0,W_{22},W_{23},W_{33},W_{32}]$
$W_{32} = P_3[W_{31},W_{33},W_{41},W_{42},W_{43}]$
$W_{41} = P_4[W_{42},W_{11},W_{12},W_{13},W_0]$

Measure for -
$P_1 = Wavg[P_1[W_{11},W_{12},W_{13}]]$
$P_2 = Wavg[P_2[W_{21},W_{22},W_{23}]]$
$P_3 = Wavg[P_3[W_{31},W_{32},W_{33}]]$
$P_4 = Wavg[P_4[W_{41},W_{42},W_{43}]]$

$W_n = P_n[W_{n-1},W_{n+1},W_{n+2}]$

For words with least or no dependency on properties, such as

$$P_3[W_{31}] \leftarrow [\text{Word W-1}] \rightarrow P_1[\text{No Words}]$$

$$\downarrow$$

$$P_2[W_{21},W_{22}]$$

$P_1 = Pw * Cv$

Pw = Property weightage

Cv = Constant value

Property weightage depends on the context. In searching, homonyms are more important than synonyms. Property weightage is calculated based on context or equivalency.

PW = CW/TP

CW : Context Weight

TP : Equalency

CW = CP * CR

CP : Total number of context properties

CR : Context Rank

1/CR = 1/R1 + 1/R2 + 1/R3 + ... + 1/Rn

R1,R2 are ranks of each property

Sentence Comparison

Semantic → Look for the meaning of the compared words with measure from their synonyms.

Contextual → Instead of using only synonyms, also use various properties in parts of speech (Configurable).

Grammatical → Same as contextual but usage information is also included replacing contextual information.

$C_i = ((M(W_1)_p - M(W_2)_p)/M(P))*100$

$C = \Sigma C_i W_i / \Sigma W_i$

Results Aggregator

After classifying the retrieved results using the above measures, we will rank the results so as to which one will be first displayed to the user, which one will be displayed next and so on.

Presenter

In this section, we will generate search meta data to uniquely identify the search and add or merge it to the generated list(mentioned in section 3.1) if possible. This is run during search service. So here the search meta data and device meta data is obtained, then UI specific transformation is applied and finally the data is presented to the user. UI includes cards(Google Now), legacy(Bing), domain specific(Wikipedia) and field specific(advanced).

Conclusion and Future Work

The proposed model for categorization of the documents as abusive or non-abusive, we can apply the same for any domain. Since the retrieved results are in the categorized form, it is highly useful to the user. The problem is with sub-categories for each domain. Another field where this module exactly can be implemented is in social networks, where all the abusive comments that is being given by the handler can be denied from being broadcasted. This would help restrict those abusive content from the view of the user.

References

[1] Choi, Dongjin, et al. "Text analysis for detecting terrorism-related articles on the web."*Journal of Network and Computer Applications* 38 (2014): 16-21.

[2] Babisaraswathi, R., N. Shanthi, and S. S. Kiruthika. "Categorizing Search Result Records Using Word Sense Disambiguation."

[3] Hemayati, Reza Taghizadeh, Weiyi Meng, and Clement Yu. "Categorizing Search Results Using WordNet and Wikipedia."*Web-Age Information Management*. Springer Berlin Heidelberg, 2012. 185-197.

[4] Hemayati, Reza, Weiyi Meng, and Clement Yu. "Semantic-based grouping of search engine results using WordNet."*Advances in Data and Web Management*. Springer Berlin Heidelberg, 2007. 678-686.