International Journal of Informatics and Communication Technology (IJ-ICT)
Vol.7, No.1, April 2018, pp. 8~12
ISSN: 2252-8776, DOI: 10.11591/ijict.v7i1.pp8-12

An Optimum Approach for Preprocessing of Web User Query

Sunny Sharma, Sunita, Arjun Kumar, Vijay Rana*
Department of Computer Science & Engineering, Arni University Kathgarh, Indora, India

Article Info

Article history:

Received Jan 18, 2018
Revised Feb 27 , 2018
Accepted Mar 7, 2018

Keyword:

NLP
Stemming
Stopwords
Spelling correction

ABSTRACT

The emergence of the Web technology generated a massive amount of raw data by enabling Internet users to post their opinions, comments, and reviews on the web. To extract useful information from this raw data can be a very challenging task. Search engines play a critical role in these circumstances. User queries are becoming main issues for the search engines. Therefore a preprocessing operation is essential. In this paper, we present a framework for natural language preprocessing for efficient data retrieval and some of the required processing for effective retrieval such as elongated word handling, stop word removal, stemming, etc. This manuscript starts by building a manually annotated dataset and then takes the reader through the detailed steps of process. Experiments are conducted for special stages of this process to examine the accuracy of the system.

Corresponding Author:

Vijay Rana,
Department of Civil Engineering,
Arni University Kathgarh,
Indora, India.
Email: vijay.rana93@gmail.com

1. INTRODUCTION

With the large amount of information available on the Web, it is much tiresome task to locate and retrieve the desired information. However the user needs to invest the time in hours to make himself satisfy for his needs. In such cases, Web Search engines play a great role in retrieving meaningful results for the user query. A *search engine* is a software system that is designed to search for information on the World Wide Web correspond to keywords or characters specified by the user [1-3]. These search engines built on various Information Retrieval techniques have thoroughly changed the ways that people search and acquire information. Nevertheless, in many situations search engines have difficulties in retrieving relevant and quality information due to noisy nature of user's query [4]. This problem occurs in many ways, for instance when a user searches the same query in different ways or enters ambiguous queries. Moreover, the user query is often much ambiguous to be easily understood by the system. So in due course, the results retrieved by the search engine are deficient. Therefore, it is important to preprocess the query by exploiting imperative algorithms before passing it to the search engine [5]. In query processing phase a set of keywords is given as an input for preprocessing, which describes the user information needs. We have created an optimum system for preprocessing of Web user query.

A set of keywords is given as an input which describes the user information needs. In contrast to usual search engines that retrieve only the results on the basis of the likely search while ignoring the semantics of the user requirements, our scheme uniquely contributes a special and novel algorithm that focus on finding the relevant meaning that describes the user's desires. The algorithm tries to find the user behavior and prevents from the repetition of accessed data. Thus it is an intelligent module as it improves the probability of success by finding the appropriate results. It performs several tasks to achieve the preprocessing phase: Stop words removal, Stemming, Spelling Correction, Tokenization and part of speech. Tokenization is the process of splitting up a query string into a set of tokens or words. It usually splits words by blank, punctuation and quotation marks at both sides of a sentence. The tokens not only considered as words but also numbers, punctuation marks, parentheses and quotation marks. Parsing is the process of analyzing a string of symbols, either in natural language or in computer languages, conforming to the rules of a formal grammar.

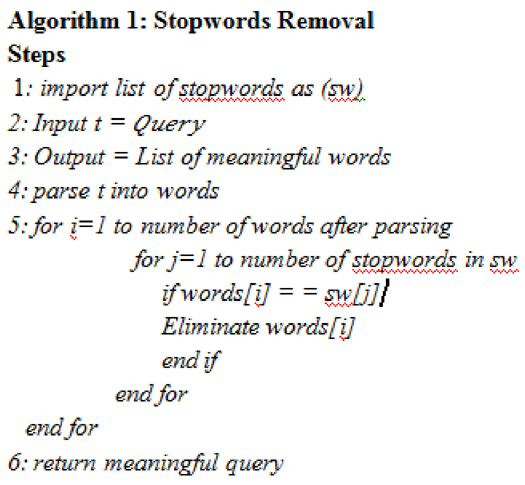
Our mean to focus on finding the relevant meaning that describes the user's behavior and prevents from the repetition of accessed data [6]. It improves the probability of success by finding the appropriate and concerned results. It performs several tasks to achieve the preprocessing phase such as stop words removal, stemming, spelling correction, tokenization. Stopwords are words which are removed after processing of natural language. The process of reducing words to their stem known as Stemming and whereas tokenization is the method of splitting the sentence into terms.

2. QUERY PREPROCESSING TECHNIQUES

2.1. Stopwords Removal:

Stopwords [7] are the superfluous terms which are detached after processing of natural language. The motivation is automating the process of identifying and removing the stop words and produces the list of meaningful words. Stop words are like (the, of, and, or, etc.). These types of words don't carry any weight, hence need to be removed.

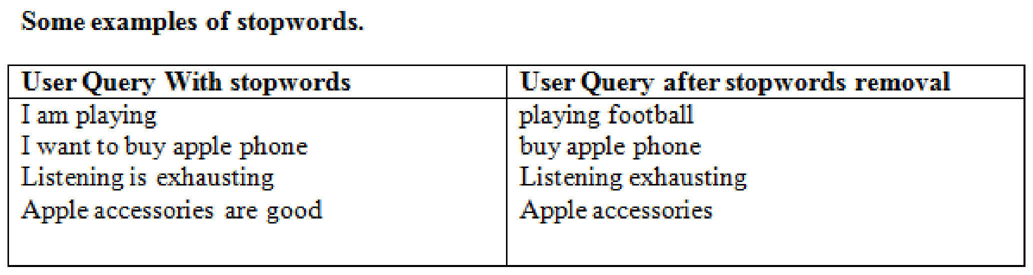
If the texts are based on a template, it might be useful to remove the words that make up the template to reduce these words' impact on the similarity measure [8].



Algorithm 1: Stopwords Removal

Steps

```
1: import list of stopwords as (sw)
2: Input t = Query
3: Output = List of meaningful words
4: parse t into words
5: for i=1 to number of words after parsing
             for j=1 to number of stopwords in sw
                if words[i] = = sw[j]
                Eliminate words[i]
                end if
            end for
   end for
6: return meaningful query
```



Some examples of stopwords.

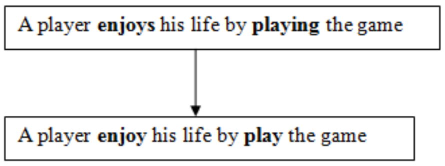
User Query With stopwords	User Query after stopwords removal
I am playing	playing football
I want to buy apple phone	buy apple phone
Listening is exhausting	Listening exhausting
Apple accessories are good	Apple accessories

2.2. Stemming:

It is the process of reducing the terms to their stem. Its aim at identify the basic forms of word. During this phase affixes and other lexical components are removed from each token, and only the stem remains [8]. For example, played and playing are both stemmed into play.

```
Algorithm 2: Stemming
Steps
1: import stemmer
2: Input t = List of meaningful words
3: Output = Sentence with stemmed words
4: arr[ ] = parse(t)
5: for i=1 to number of words in arr[ ]
            if arr[i] = = compound word
            stemmer.stem(arr[i])
            end if
      end for
6: return query with stemmed words
```

Example of Stemming


A player **enjoys** his life by **playing** the game

A player **enjoy** his life by **play** the game

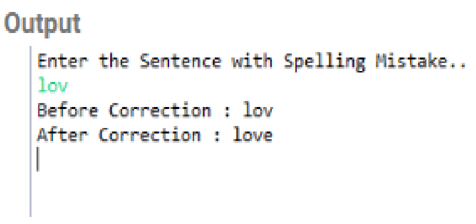
2.3. Spelling correction:

The proposal is to use the correct spelling [9] of the word that is most regular among queries typed in by other web users. It is more probable that the user who typed lov intended to type the query love. There is list of corrected words in the dictionary. The system finds the distance between misspelled words and corrected word and returns a corrected word, where there is smallest distance between misspelled word and corrected word.

The edit distance between two strings (or words) X and Y of lengths m and n respectively, can be defined [10] as follows.

$$
\begin{array}{rcll}
ed(X_{1\ldots i+1}, Y_{1\ldots j+1}) & = & ed(X_{1\ldots i}, Y_{1\ldots j}) & \text{if } x_{i+1} = y_{j+1} \\
 & & & \text{(last characters} \\
 & & & \text{are the same)} \\
 & = & 1 + min\{ed(X_{1\ldots i-1}, Y_{1\ldots j-1}, & \text{if both } x_i = y_{j+1} \\
 & & \qquad ed(X_{1\ldots i+1}, Y_{1\ldots j}), & \text{and } x_{i+1} = y_j \\
 & & \qquad ed(X_{1\ldots i}, Y_{1\ldots j+1})\} & \\
 & & & \text{(last two characters} \\
 & & & \text{are transposed)} \\
 & = & 1 + min\{ed[X_{1\ldots i}, Y_{1\ldots j}), & \text{otherwise} \\
 & & \qquad ed(X_{1\ldots i+1}, Y_{1\ldots j}), & \\
 & & \qquad ed(X_{1\ldots i}, Y_{1\ldots j+1})\} & \\
ed(X_{1\ldots 0}, Y_{1\ldots j}) & = & j & 0 \le j \le n \\
ed(X_{1\ldots i}, Y_{Idots0}) & = & i & 0 \le i \le m \\
ed(X_{1\ldots -1}, Y_{1\ldots j}) & = & ed(X_{1\ldots i}, Y_{1\ldots -1}) = max(m, n) & \text{(Boundary definitions)}
\end{array}
$$

```
Algorithm 3: Spelling Correction:
Steps
1: import dictionary as dic, distance
2: Input t[] = misspelled word
3: Output = correct word
4: for i=1 to n words in t
      for j= 1 to size of dic
         compute min distance( t, dic[i])
      end for
      print dic[i]
  end for
6: End
```


Output

```
Enter the Sentence with Spelling Mistake..
lov
Before Correction : lov
After Correction : love
```

2.4. Tokenization:

Tokenization is the scheme of splitting the sentence into terms. Different policies can be chosen regarding how to split into words, and the choice depends on the type of data to tokenize [8]. The main characteristic of tokenization is to remove noisy words, symbols and numbers that can affect the performance of the search query.

```
Algorithm 4: Tokenization
Steps
1: import split
2: input t = string
3: arr[ ] = t.split( )
4: for i = 1 to arr.length
          print arr[i]
       end for
5. End
```

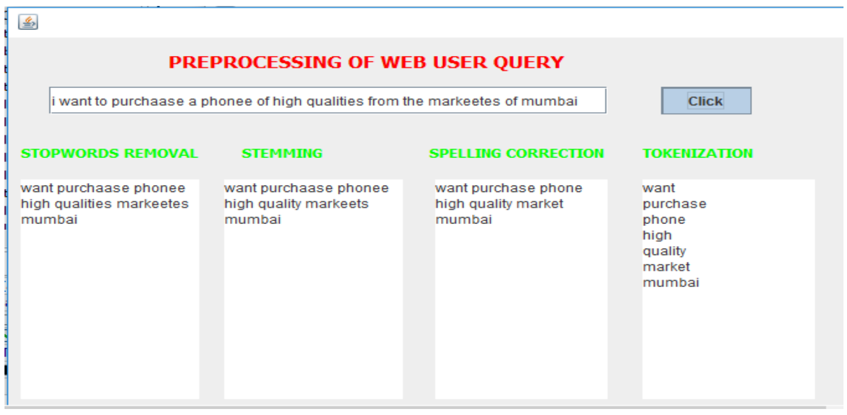
3. EFFECTIVE APPROACH FOR PREPROCESSING

In the supervised approach, which is also known as the corpus-based approach, machine learning classifiers such as Support Vector Machine (SVM), Naïve Bayes (NB), Decision Tree (D-Tree), K-Nearest Neighbor (KNN), etc., are applied to a manually annotated dataset [11]. After building the dataset, some pre-processing techniques are automatically applied to it.

```
Algorithm 4: Effective preprocessing for user query
Steps:
1. Import sw[], stemmer, dic[], k=0
2. Input t = query
3. arr[] = parse(t)
4. for i= 1 to size of arr[]
       for j= 1 to size of sw[]
         if words[i] == sw[j]
          Eliminate words[i]
         end if
       end for
  end for
5. arr2[] = meaningful query without stopwords
6. for i= 1 to size of arr2[]
       if arr[i] == compound word
       stemmer.stem(arr[i])
       end if
  end for
7. arr3[] = stemmed words
8. for i= 1 to size of arr3[]
       for j = 1 of dic[]
       compute min distance(arr3[], dic[j])
       end for
       k++
       arr4[k] = arr3[i]
  end for
9. for i= 1 to size of arr[]
       print arr[i]
   end for
10. End
```

4. IMPLEMENTATION

The implementation has done on Eclipse IDE using Java language with jdk (java development kit). Eclipse is an fundamental tool for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration. Figure 1 illustrates the results obtained on the query “I want to purchase a phonee of high qualities from the markeetes of Mumbai”.



PREPROCESSING OF WEB USER QUERY

i want to purchaase a phonee of high qualities from the markeetes of mumbai

Click

STOPWORDS REMOVAL

want purchaase phonee high qualities markeetes mumbai

STEMMING

want purchaase phonee high quality markeets mumbai

SPELLING CORRECTION

want purchase phone high quality market mumbai

TOKENIZATION

want
purchase
phone
high
quality
market
mumbai

Figure 1. Preprocessing of User Query

Step-wise Illustration

Step 1.Input user query: I want to purchase a phonee of high qualities from the markeetes of mumbai
Step 2. Execute Stopwords removal module:
Output: want purchase phonee high qualities markeetes mumbai (query without any stopword)
Step 3. Execute Stemming:
Output: 3 want purchase phonee high quality markeete mumbai (query after performing stemming)
Step 4. Execute Spelling Correction module:
Output:4 want purchase phone high quality market mumbai (query with corrected word)
Step 5. *Execute* Tokenization:
Output:'want' 'purchase' 'phone' 'high' 'quality' 'market' 'mumbai' (Tokens)

5. CONCLUSION

This paper has given complete information about preprocessing techniques, i.e. stop words elimination and stemming algorithms. We hope this paper will help text mining researcher's community. We believe that, we have attempted to present an essential, dynamic, robust and novel tool for classification of Web user behavior and our results and findings support the strength of the system. The semantics analysis feature can be further enhanced by providing natural language processing techniques.

REFERENCES

[1] Sharma, Sunny, Vijay Rana. "Web Personalization through Semantic Annotation System." Advances in *Computational Sciences and Technology* 2017; 10(6): 1683-1690.
[2] Mahajan, Sunita, Sunny Sharma, Vijay Rana. "Design a Perception Based Semantics Model for Knowledge Extraction." *International Journal of Computational Intelligence Research* 2017; 13(6): 1547-1556.
[3] Brin, Sergey, Lawrence Page. "Reprint of: The anatomy of a large-scale hypertextual web search engine." *Computer networks* 2012; 56(18): 3825-3833.
[4] Song, Ruihua, et al. "*Identifying ambiguous queries in web search*." Proceedings of the 16th international conference on World Wide Web. ACM, 2007.
[5] Srivastava, Jaideep, et al. "Web usage mining: Discovery and applications of usage patterns from web data." *Acm Sigkdd Explorations Newsletter* 2000; 1(2): 12-23.
[6] Granka, Laura A., Thorsten Joachims, and Geri Gay. "*Eye-tracking analysis of user behavior in WWW search*." Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2004.
[7] Wilbur, W. John, Karl Sirotkin. "The automatic identification of stop words." *Journal of information science* 1992; 18(1): 45-55.
[8] Runeson, Per, Magnus Alexandersson, Oskar Nyholm. "*Detection of duplicate defect reports using natural language processin*.". Proceedings of the 29th international conference on Software Engineering. IEEE Computer Society, 2007.
[9] Peterson, James L. "Computer programs for detecting and correcting spelling errors". *Communications of the ACM* 1980; 23(12): 676-687.
[10] http://www.jandaciuk.pl/thesis/node39.html
[11] Abdulla, Nawaf A., et al. "*Arabic sentiment analysis: Lexicon-based and corpus-based*." Applied Electrical Engineering and Computing Technologies (AEECT), 2013 IEEE Jordan Conference on. IEEE, 2013.