International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-3 Issue-5, June 2014

Enhancing Scalable Database-Driven Reverse Dictionary

Priyanka Nanaware, Rahul Ambekar

Abstract— Contrasting a traditional Forward Dictionary which convert word to their meaning ,we express design and implementation of reverse dictionary, a reverse dictionary return a set of candidate words that assure the input phrase, that input phrase concerning the desired idea. in current paper , we here a set of algorithms and the results of a set of experiment showing the retrieval accurateness of our methods and the runtime response time performance of our completion. This effort has Major useful for general public those who work closely with word also in general field of conceptual search. Our conduct experiment judge against the quality of the result to currently available implementations of reverse dictionary also to provide major improvements in performance level.

Index Terms— Keywords— Dictionaries, Search process, Web-based services, Reverse mapping

I. INTRODUCTION

In this paper, we report work on creating an online Reverse Dictionary and also enhancing features such as efficiency, response time. Reverse Dictionary performs the opposite mapping i.e. input is given as phrase which describing the concept, it gives words whose definitions equivalent to the entered phrase, as opposed to traditional(forward) dictionary which maps word to their meaning. Reverse dictionary provide the user an opportunity to enter the phrase "check out natural caves" as input, and anticipate to acquire the word "spelunking" (also provide other words with similar meaning) as output. Effectively, the Reverse dictionary resolve the problem is "sometime words is on my tongue But can't relatively memorize it". A particular category of people such as writer with student, professionals, teachers, scientists and list goes on, heavily cause this problem. Using Reverse dictionary we reduce the concept similarity problem (CSP), the check with the forward Dictionary as removal and select those words whose definitions are related to this meaning to form output to this RD search.

II. RELATED WORK

Some work in the literature does deal with multiword matches. One area of such work [7] addresses the problem of result the similarity of multiword phrases crosswise a set of documents in Wikipedia. However, there is a additional problem (our second problem) in relate these technique directly, because the methods proposed in these works assume that the documents contain sufficient related information (at least 40-90 words) for matches.

Manuscript published on 30 June 2014.
***** Correspondence Author (s)
Priyanka Nanaware, Research Scholar, Department of Computer Engineering, Sighgad Institute of Technology, Lonavala, Maharashtra, India.
Rahul Ambekar, Asst. Prof., Department of Computer Engineering, Sighgad Institute of Technology, Lonavala, Maharashtra, India.

Identical, which fits within the conventional notion of "short documents" in IR research [2].In another area of related work, i.e., passage retrieval [2], [7], [3], the concept of concern is copy as a multiword input query. Work in this area effort to identify relevant passages in large-text documents. Present semantic likeness capacity schemes are highly computationally intensive, making online scaling difficult [4]. The two most general methods to complete this, hidden semantic indexing (LSI) [10] and principal section analysis (PCA) [9], both analyze the keywords of documents in a amount to identify the central concepts in the document. Accordingly these foremost concepts are representing as vectors in the keyword space and are used as the basis of similarity association for classification. Such Vectorization is a highly calculating exhaustive operation, as is the evaluation of the distance between two vectors (using techniques like cosine distances [5]). In this paper, we report the creation of the WordStar Reverse Dictionary (WRD), a database-driven RD system that challenges to address the core problem identified above. The WRD not only fulfils new functional objectives delineate above, it does so at an order of amount performance and scale improvement over the best concept similarity quantity schemes available without impacting solution quality. We also express that the WRD is far better in solution quality than the two commercial RDs [1], available. Using RD we improving the efficiency and response time of system using multiple techniques. The remainder of this paper is structured as follows: In Section 2, we describe our proposed solution and in Section 3, we describe the architecture of the WRD system

III. PROPOSED SOLUTION

Reverse dictionary system is based on the concepts that a phrase that conceptually explain the word should de like the word's with real meaning, if not matching the exact words then provide at least conceptually similar for example, consider a phrase:"talks a lot, but without much substance " a reverse dictionary return a word such as "talkative", "garrulous", "chatty". The dictionary Definition of garrulous "be full of slight discussion" which is clearly closes in concepts, but not includes exact matching words. In our RD, a user might input a phrase describing an unknown term of interest. Since an input phrase might potentially satisfy the definition of multiple words, a RD should return a set of possible matches from which a user may select his/her choice of terms. This is complex, however, because the user is unlikely to enter a definition that exactly matches one found in a dictionary. However, the meaning of the phrase the user entered should be conceptually similar enough to an actual dictionary definition to generate a set of possible matches.

One approach might be to evaluate the user input phrase to every definition in a dictionary, appear for definitions containing the same words as the user input phrase. Such an approach has two major problems: 1) it requires the user's input phrase to contain words that exactly match a dictionary definition; and 2) it does not scale well—for a dictionary containing more than 100,000 defined words, where each word may have multiple definitions, it would require potentially hundreds of thousands of queries to return a result. This is a form of inverted index [1], [2,7], where we incorporate stemming to ensure that we use the most generic form of a word possible This removes the possibility of concepts that are actually similar, but are not found to be similar based on differing grammatical usage. This optimization, though useful, only takes us so far. The user's input phrase must still exactly match words in a dictionary definition, which is very unlikely.

We further define a phrase P as a sequence of one or more terms, i.e.

P = <t1; t2….ti….tn >.

A dictionary D is a set of mappings P -> P.

For clarity, We will distinguish between the two classes of phrases, Word phrases (W) and sense phrases (S), Where W refers to a word or sequence of words indexed for lookup in a Dictionary, and S refers to a definition of a W in a dictionary. Under this classification, a dictionary D can be redefined as a set of mappings W -> S. In particular, the mapping Wi ->Sj expresses the following relationship: Sj denotes a sense (or Meaning) of the (word) phrase Wi. From this meaning to word mapping create the Forward mapping (standard dictionary), Reverse mapping (reverse dictionary)also create Synonym set Antonym set,Hypernym set, Hyponym Set.to create this set from WordNet database.

User phrase: A user phrase U is a phrase defined by a sequence of terms <t1; . . . ; tk; . . . ; tx > input by a user. This string serves as input to our method. Given an input U, we note that some common words are not useful for the purposes of indexing. Such words are called stop words, Stop word sets: We define two sets of stop words, Level 1 stop words (L1), which are always removed during index building and querying, and Level 2 stop words (L2), which may or may not be useful in indexing. An example of an area where a word may or may not be useful is gender, because the gender is important in some cases but not others Negation word set: Negation is also a concern. Query: A query phrase Q is a Boolean expression based on an input U. Output: The output O of our method consists of a set of Ws, such that a definition of each W in the output satisfies Q. In the proposed work implement our own Algorithm which will increase efficiency and decrease the response time of our Project The algorithm will be as Follows

Algorithm:

1. U = *GetPhraseFromTextField*()
2. If(*isPhrasePresent* (U)){
3. //show it on Auto complete box
4. //which reduce further typing of user and may he want to find same thing
5. Result = *getResultfromDatabase*(U)
6. Collect the obtained result
7. *showResultOnPage*(Result); }
8. Else{
9. //implement our base algorithm to find result for phrase U
10. *StorePhraseAndResultinDB* (Result,U);}

For e.g. If a user types un* then all the words starting with 'un' would be presented to user. Such as unaware, unity, unarmed, unethical…Let U is the set of abstract phrase provided by the user. E.g. abs*.

U = {u1*, u2*, u3*, u4*…}

Algorithm:

1. U=GetPhraseFromTextFieldOfWordsFinding()
2. Process the obtained substring.
3. I.e Placeholder will be present already, remove it.
4. **if** (words not found) {
6. Give message "no matching words found";}
8. **else** {
10. Query the dictionary database: Select word from db where word starts with 'UN'.
11. Where un* = all words starting with 'UN'.
12.*er = all words ending with 'er' **for** e.g. Defender, Founder, Cleaner etc..}
14. ResultSet set;
15. while (rs.next())
16. list = setWords ;

If suppose the user wants to search a specific word in the category of Parts Of Speech. The next feature of ours will help him/her to find out that. For e.g. user type's animal: noun all noun results related to animal as a noun would be shown to user. Let U be the set of phrase provided by the user. E.g. animal: noun.

U= {u1: noun, u2: verb, u3: adjective, u4: adverb}

Where U = Main set of phrase of required result. And u (i) (identifier) = term for which requested parts of speech to be returned. Second, the implementation of a thread pool allows for parallel retrieval of synonym, hyponym, hypernym, and RMS sets for terms.

IV. SYSTEM ARCHITECTURE

We now describe our implementation architecture, with particular attention to design for scalability. Fig. 1 presents the architecture of the system. The Reverse Dictionary Application (RDA) is a software module that takes a user phrase (U) as input, and returns a set of conceptually related words as output.

RDA needs access to information stored in a set of databases:

- the RMS DB, which contains a table of mappings t → R(t), as well as dictionary definitions and computed parse trees for definitions;
- the Synonym DB, which contains the synonym set for each term t;
- the Hyponym/Hypernym DB, which contains the hyponym and hypernym relationship sets for each term t;
- the Antonym DB, which contains the antonym set for each term t; and
- the actual dictionary definitions for each word in the dictionary.

The mappings for the RMS, synonyms, hyponyms, hypernyms, and antonyms are stored as integer mappings, where each word in the Wordnet dictionary is represented by a unique integer. This both condenses the size of the mapping sets, and allows for very fast processing of similarity comparisons, as compared to string processing.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

This architecture has three characteristics designed to ensure maximum scalability of the system. First, a cache stores frequently accessed data, which allows a thread to access needed data without contacting a database. It Is well known that some term occures more frequently Than other, The synonym, hyponym, hypernym, and RMS sets of these popular terms will be stored in the cache and the query execution in the database will be avoided

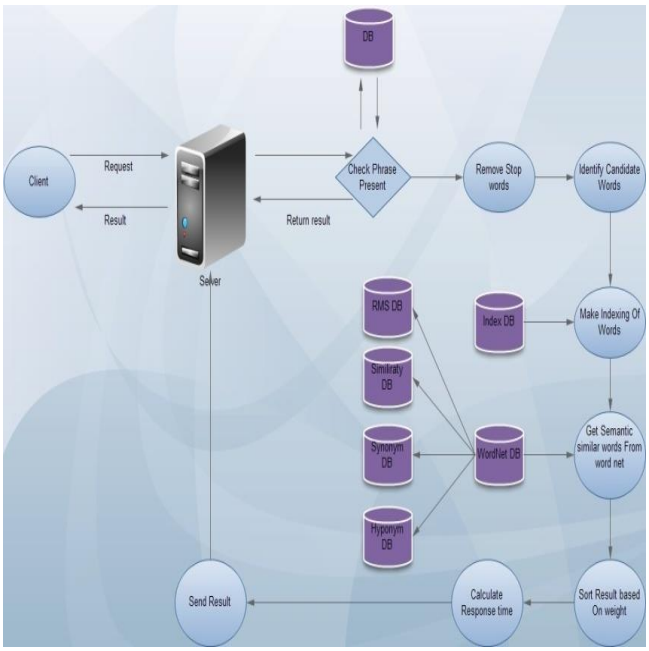

Fig. 1 Architecture of the System

V. MATHEMATICAL MODEL

Set Theory -

1. Identify Candidate Words

U is the phrase which user gives input.
U = {t1, t2, t3….tn},t1, t2...tn are terms used in Phrases i.e. Se ∈ Di.
Find out candidate words and remove stop words from phrase such that tn ∈ Se in Di.

2. Apply Stemming To each Candidate Word

Form a set of terms {t1…….tn} such that tn ∈ Se ,Obtain tn from each term from above Set.

3. Obtain Query

Form a set W. Add all candidate terms to it separated by AND such that tn ∈ U; tn ∉ S1. S1 = set of category1 stop words.

Remove negation: Let Q = t1*t2*t3… Where * ∈ {AND, OR} ∀ negative tn ∈ Q

4. Obtain Results

If |O| > a; where a = minimum no. of words to sort results.
Sort Results O. ∀ tn ∈ Q such that tn ∈ S2 where S2 = set of category2 stop words.
If |O| > a Sort Results O
O = O ∪ getSynonym(Q)
If |O| > a Sort Results O
O = O ∪ getHypernyms(Q)
If |O| > a Sort Results O
O = O ∪ getHyponyms(Q)

Let Q be the set of Nw = {not, never} remove such words from set with its Synonym word Like for

Execute Query On Set:

Given W = {w1, w2, w3….wn}
W = w1*w2*w3*w4 where * ∈ {AND, OR}.
If * = AND Then w1 ∩ w2
If * = OR Then w1 ∪ w2.
Where W= set of resultant words
{w1, w2...} is a set of similar meaning words with phrase U
{t1, t2, t3….} = {t1 AND t2 AND t3 AND}
Here terms are combined with AND
{t1, t2, t3, t4….} = {t1 AND t2 AND (t3 OR t4)}
Here similar terms are combined with OR

5. Sort Result

W = {w1, w2, w3…} the result should be in Sorted format with weightage. For weightage a Sense phrase Se and User phrase U is taken into account. ∀ paires of terms tn (a, b)
Where a ∈ Se and b ∈ U Computing similarity of term by the given formula

ρ(a,b) = 2*E(A(a,b))/(E(a)+E(b))

Where ρ = similarity returned by the equation.
A (a, b) = gives LCA (Least Comman Ancestor) by open NLP parser.
E (a) = gives the depth of the term in Wordnet
Computing importance of a term λ (a, S) is given by formula.

Λ (a, S) = (d (Z(S))-da)/ d (Z(S)

Where Λ = importance of a term in Sense phrase S,da = depth of term .d (Z(S)) = overall depth where a ∈ S
Computing importance of a term λ (b, U) is given by formula.

Λ (b, U) = (d (Z (U))-db)/ d (Z (U)

Where Λ = importance of a term in User phrase U,db = depth of term ,d (Z(U)) = overall depth where b ∈ U.
Using the information obtained from above we calculate weighted similarity factor μ given by

μ (a,S, b,U) = Λ(a,S)* Λ(b,U)* ρ(a,b)

Where a ∈ S and b ∈ U.

VI. CONCLUSION

In this paper we present our work in developing a meaning-to-word dictionary. Depending on the phrase input there may be variation of the results shown. With our feature of finding words from an abstract string given as an input empowers the concept of reverse dictionary Parts of Speech classification feature also add quality. Improving the efficiency by holding the results makes the access fast. Adding new features like words searching enhances our work from previously available.

REFERENCES

[1] Ryan Shaw, Debra Vander Meer and Kaushik Dutta ”Building a Scalable Database-Driven Reverse Dictionary”, vol. 25,no. 3,march 2013.
[2] T. Dao and T. Simpson, “Measuring Similarity between Sen-tences, ”http://opensvn.csie.org/WordNetDotNet/trunk/Projects/Thanh/Paper/WordNetDotNet_Semantic_Similarity.pdf(last accessed 16 Oct. 2009), 2009.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

[3] Dictionary.com, LLC, "Reverse Dictionary,"http://dictionary. reference.com/reverse, 2009.
[4] E. Gabrilovich and S. Markovitch, "Wikipedia-Based Semantic Interpretation for Natural Language Processing," J. Artificial Intelligence Research, vol. 34, no. 1, pp. 443-498, 2009.
[5] T. Hofmann, "Probabilistic Latent Semantic Indexing," Proc. Int'l Conf. Research and Development in Information Retrieval (SIGIR),pp. 50-57, 1999.
[6] OneLook.com, "Onelook.com Reverse Dictionary," http://www.onelook.com/, 2009.
[7] X. Phan and C. Nguyen, "A c/c++ Implementation of Latent Dirichlet Allocation (lda) Using Gibbs Sampling for Parameter Estimation and Inference," http://gibbslda.sourceforge.net/, 2010.
[8] D. Milne and I. Witten, "Learning to Link with Wikipedia," Proc.17th ACM Conf. Information and Knowledge Management, pp. 509-518, 2008.
[10] D. Widdows and K. Ferraro, "Semantic Vectors,"http://code.google.com/p/semanticvectors/, 2010

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)