Research Paper

Volume : 3 | Issue : 5 | May 2014 • ISSN No 2277 - 8179

Building an Efficient Reverse Dictionary

Engineering

KEYWORDS :

Sneha P.S — PG Scholar, Department of Computer Science & Engineering, Vedavyasa Institute of Technology, University of Calicut , Kerala, India

Mrs.S.Kavitha Murugesan — Associate Professor, Department of Computer Science & Engineering, Vedavyasa Institute of Technology, University of Calicut , Kerala, India

ABSTRACT

In a traditional forward dictionary, words are mapped to their definitions. In the case of a reverse dictionary, a phrase describing the desired concept is taken as the user input, and returns a set of candidate words that satisfy the input phrase. This research work describes the implementation of an efficient reverse dictionary where the user is also able to find out the actual dictionary definition of those words returned as output and thereby help the user to check the accuracy of the results obtained from the reverse dictionary output. Effectively, a Reverse Dictionary addresses the "word is on the tip of my tongue, but I can't quite remember it" problem. This work has significant application not only for the general public, particularly those who work closely with words, but also in the general field of conceptual search. This research work proposes a novel approach called "wordster" to create and process the reverse dictionary and also evaluate the retrieval accuracy and runtime response time performance. The wordster approach can provide significant improvements in performance scale without sacrificing the quality of the result.

I. INTRODUCTION

This paper report work on creation of an efficient online reverse dictionary .An online dictionary [1] is a dictionary that is accessible via the Internet through a web browser. A forward dictionary is one which maps from words to their definitions. What if we already had the meaning or the idea but aren't too sure of the appropriate word, then REVERSE DICTIONARY is the right one for use. For example, to find the word "doctor" knowing only that he is a "person who cures disease". The RD addresses the "word is on the tip of my tongue, but I can't quite remember it" problem. This problem is mainly faced by writers, including students, professional writers, scientists, marketing and advertisement professionals, teachers, the list goes on.

While creating a reverse dictionary, it is important to consider the following two constraints. First, the user input phrase may not exactly match the forward dictionary definition of a word. For example, user may enter the phrase "talks a lot" when looking for a concept such as "garrulous" whose dictionary definition might be "full of trivial conversation". Both phrases do not contain same words but are conceptually similar. Second, the response to a request must be quick since it needs to be usable online. According to a recent study, online users become impatient if the response to a request takes longer than 4-5 seconds. In order to build a reverse dictionary, first a forward dictionary is needed. WordNet is the forward dictionary used in this work. WordNet [2] is a lexical database which is available online and provides a large repository of English lexical items. There is a multilingual WordNet for European languages which are structured in the same way as the English language WordNet. WordNet was designed to establish the connections between four types of Parts of Speech (POS) - noun, verb, adjective, and adverb. The smallest unit in a WordNet is synset, which represents a specific meaning of a word. It includes the word, its explanation, and its synonyms. The specific meaning of one word under one type of POS is called a sense. Each sense of a word is in a different synset. Synsets are equivalent to senses= structures containing sets of terms with synonymous meanings. Each synset has a gloss that defines the concept it represents. For example, the words night, night-time and dark constitute a single synset that has the following gloss: the time after sunset and before sunrise while it is dark outside. Synsets are connected to one another through the explicit semantic relations. Some of these relations (hypernym, hyponym for nouns and hypernym and troponym for verbs) constitute is-a-kind-of (holonymy) and is-a-part-of (meronymy for nouns) hierarchies.For example, tree is a kind of plant, tree is a hyponym of plant and plant is a hypernym of tree. Analogously, trunk is a part of a tree and takes trunk as a meronym of tree and tree is a holonym of trunk. For one word and one type of POS, if there is more than one sense, WordNet organizes them in the order of the most frequently used to the least frequently used (Semcor).

Work in text analyzer, surveyed in detail in [3] , performs exact word searching. However, there is no text stemming. The exact text matching is case sensitive. That means, the word "JAVA" is different from the word "Java". Synonyms such as "sick" and "ill" are words that are used in particular phrases where they denote unique meaning but are not identified. The work discussed in [4] need to tackle other parts of speech in addition to nouns and must consider multi-word expressions and multiple senses of polysemous words and their translations. The pairwise similarity of words in two phrases is well described in and the major drawback identified in the work is sequence of words is not considered important.For example, Consider two phrases: "the dog who bit the man" and "the man who bit the dog." They contain the same words and so the method in [2] would consider them virtually equivalent, without considering that the two phrases convey very different meanings.

SVM approach discussed in [5] deal with the pre-creation of a context vector for each word in WordNet during the learning phase.Since the SVM approach has access to all the word relationship data available and compares the input phrase to the context vector for each word in the WordNet corpus, we can use the SVM quality as a rough benchmark for high quality output. In LSI[6], pre-create the model for each word in WordNet(Similar as in SVM).Once the model has been created and given input to the system, it can be directly searched by a user input U. The main disadvantage with the SVM approach and LSI is that the response time is high.

In LDA [7], input all the dictionary words and their corresponding context vectors to GibbsLDA++, and receive a set of classes as output. Then provide the user input U to identify the class to which U belongs, based on GibbsLDA++ inference. The dictionary words corresponding to that class are identified as the Reverse Dictionary output. LDA can be readily embedded in a more complex model—a property that is not possessed by LSI. The work described in need to compute the user input concept vector at runtime and then subsequently compute the distance between this and the dictionary concept vectors. Vector computation is known to be quite computationally intensive even though much effort has been expended in building efficient schemes.

The Cp/cv method described in deals with single word focus and addresses multiword similarity problem- where semantic similarity must be computed between multiword phrases. The two existing online reverse dictionaries [6],[7] available do not rank the output and thereby reduces the efficiency.

This paper introduces a new approach called Wordster that overcome all the problems discussed above and thereby create an efficient reverse dictionary.

II. PROPOSED APPROACH

The proposed RD system is based on the fact that the user input phrase should resemble the word's actual definition, and, if not exactly matching, then at least it must be conceptually similar. The proposed approach mainly consists of two phases. The first phase is the *find candidate word* .In this phase, when a user input phrase is received, first find candidate words from a forward dictionary, whose definition have some similarity to the user input phrase. This phase consists of two key substeps: 1)build the RMS ; and 2) query the RMS. The second phase involves the ranking of those candidate words in the order of quality of match. Before discussing the above steps in detail, there is a need to define several concepts used in this work.

Synonym set, $W_{syn}(t)$: A set of conceptually related terms for t. For example, W_{syn}(talk) might consist of {speak, utter, mouth}.

Antonym set, $W_{ant}(t)$: A set of conceptually opposite or negated terms for t. For example, W_{ant}(pleasant) might consist of {"unpleasant," "unhappy"}.

Hypernym set, $W_{hyr}(t)$: A set of conceptually more general terms describing t. For example, W_{hyr}(red) might consist of {"color"}.

Hyponym Set, $W_{hyo}(t)$: A set of conceptually more specific terms describing t. For example, W_{hyo}(red) might consist of {"maroon", "crimson"}.

Having discussed certain concepts, now we move on to describe various processing steps in the creation of a reverse dictionary.

A. Building the Reverse Mapping Sets

Build RMS of a term t, R(t), in order to find out the candidate words in whose definition the term 't' appears. For example, in a forward dictionary, the definition of word 'jovial' is "showing high-spirited merriment" and for word 'languid' is "lacking spirit or liveliness". Here RMS build for the term 'spirit'(i.e., R(spirit)) will include both "languid" and "jovial". Likewise we need to build RMS for each and every relevant term that appears in the definition of a word in the forward dictionary. This is a one-time, offline event and so it has no effect on runtime performance

B. Querying the Reverse Mapping Sets

This section, describes the use of R indexes described in the above section, to respond to user input phrases. When a user input phrase U is received, first extract the core terms from U. The next step is to apply stemming. Stemming is done in order to convert a term to its base form. For example, if the input phrase given is "two winged insect" and when stemming is applied, 'winged' will get converted to its base form 'wing'. Stemming is done through a standard stemming algorithm, e.g, the Porter Stemmer.

After stemming, consult the appropriate R indexes (ie. RMS) of these terms extracted from the user input phrase to find out the candidate words. Given an input phrase "a tall building" first extract the core terms : "tall" and "building" (the term "a" is a stop word and hence it is ignored). Then consult the appropriate R indexes, R (tall) and R (building) and will return words in whose definition "tall" and "building" occurs simultaneously. Each word becomes a candidate word.

A tuneable input parameter α is defined which represents the minimum number of candidate words needed to stop processing and return output. If the first step discussed above does not generate a sufficient number of candidate words (W) according to α, then expand the query Q to include synonyms, hypernyms and hyponyms of the terms in Q. When threshold number of candidate words have been found out, then sort the results based on similarity to U and return top β Ws where β is an input parameter representing the maximum number of words to be returned as output.

C. Ranking candidate words

Here semantic similarity of definition of candidate words "S" found is compared with the user input phrase U. On the basis of that, sorts a set of output words in the order of decreasing similarity to U as compared to S. There is a need to assign a similarity measure for each (S,U) pair, where U is the user input phrase and S is the definition of candidate words found out. Here it is necessary to compute both term similarity and term importance.

D. Term Similarity

Compute term similarity between two terms based on their location in the WordNet hierarchy. The WordNet hierarchy organizes words in English language from general at root to most specific at leaf nodes. Consider the LCA (Least Common Ancestor) of the two terms and if the LCA of two terms in this hierarchy is root, then those two terms will have little similarity. If the LCA is more deeper, two terms will have greater similarity.

Define a similarity function $\rho(a,b)$ to compute the similarity between two terms 'a' and 'b'

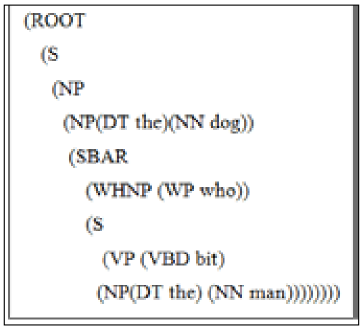
$$\rho(a,b) = \frac{2\times E(A(a,b))}{(E(a)+E(b))} \qquad (1)$$

Where "b" is the term in the user input phrase U and "a" is the term in the sense phrase S.A(a,b) return the LCA shared by both a and b in the WordNet hierarchy. E[A(a,b)] is the depth of LCA. E(a) and E(b) return the depth of terms "a" and "b" respectively. Value of ρ(a,b) will be larger for more similar terms.

E. Term Importance

Besides considering the term similarity, it is essential to consider the importance of each term in the phrase. For Example, Consider two phrases " the dog who bit the man" and "the man who bit the dog". These two phrases contain similar words but convey different meanings. So it is important to consider the sequence of words in a phrase. To generate the importance of each term, a parser can be used. OpenNLP[12] parser is used in this work. The parser returns the grammatical structure of a sentence. A parser return a parse tree for a given input phrase. In a parse tree, the terms in the phrase that add most to its meaning appears higher than those words that add less to its meaning.

TABLE I
EXAMPLE PARSER OUTPUT FOR P1



```
(ROOT
  (S
   (NP
    (NP(DT the)(NN dog))
     (SBAR
       (WHNP (WP who))
       (S
         (VP (VBD bit)
         (NP(DT the) (NN man)))))))
```

TABLE II
Example Parser Output for P2

```
(ROOT
 (S
  (NP
    (NP(DT the)(NN man))
    (SBAR
       (WHNP (WP who))
       (S
         (VP (VBD bit)
            (NP (DT the) (NN dog)))))))
```

Volume : 3 | Issue : 5 | May 2014 • ISSN No 2277 - 8179

Table 1 shows the parse tree output Z (P1) for the phrase P1= “the dog who bit the man” and table 2 shows the parse tree output Z (P2) for the phrase P2= “ the man who bit the dog”. It

is clear from the parser output for P1 that the word “dog” is more important than the word “man” (subject is more important than the object in a phrase) Since “dog” appear higher in the parse tree than “man” in Z(P1) and vice versa in Z(P2).

Define a term importance function λ(t, P) where “t” is a term in the phrase P. To compute the importance of term “a” in phrase S

$$\lambda(a,S) = \frac{\left(d\left(Z(S)\right) - d_a\right)}{d\left(Z(S)\right)} . \tag{2}$$

and to compute the importance of term “b” in a phrase U

$$\lambda(b,U) = \frac{\left(d\left(Z(U)\right) - d_b\right)}{d\left(Z(U)\right)} \tag{3}$$

Z(S) and Z(U) represents the parse tree for sense phrase S and user input phrase U respectively. d[Z(S)] and d[Z(U)] indicate the depth of Z(S) and Z(U) respectively. d_a is the depth of term “a” in Z(S) and d_b is the depth of term “b” in Z(U).

To generate a weighted similarity for each term pair (a,b) where a∊ S and b∊U, use a weighted similarity factor μ(a,S,b,U)

$$\mu(a,S,b,U) = \lambda(a,S) \times \lambda(b,U) \times \rho(a,b) . \tag{4}$$

This is given as input to generic string similarity algorithm described in [5] to obtain a phrase similarity measure M(S,U). Sort the results in the decreasing similarity to U and return top β matches.

But at times words returned may have different meanings at different contexts. So after retrieving the results from the reverse dictionary, user may have to look up at forward dictionary to find the exact word which matches their context. This will be more time consuming. In order to improve efficiency, the proposed system combines the features of forward dictionary and reverse dictionary. In the proposed system, instead of just returning the words that matches in meaning with the phrase, it returns the word along with their meaning and, thereby, improves the efficiency. Thus user can easily identify the matching word.

III. SYSTEM ARCHITECTURE

In the reverse dictionary architecture,the RDA is a software module that takes user input phrase and return a set of candidate words as output.The information is stored in separate databases and RDA needs to access to this information to perform all the processing described in section II.

1. the RMS DB, contains dictionary definitions and computed parse trees for definitions;
2. the Synonym DB, which contains the synonym set for each term ;
3. the Hyponym/Hypernym DB,contain hyponym/hypernym set for each term;
4. the Antonym DB, which contains the antonym set for each term and
5. the dictionary definitions for each word in the forward dic tionary.

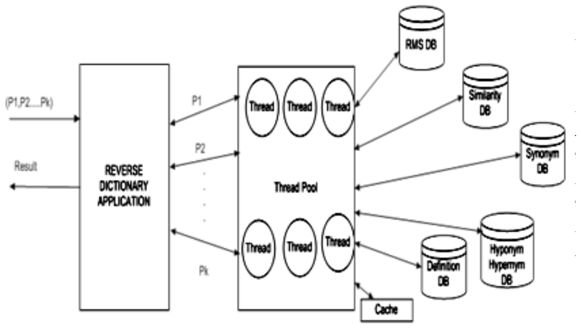


FIG 3.1: ARCHITECTURE FOR REVERSE

if so , the thread needs to contact the appropriate database to retrieve the needed data. The implementation of thread pool allows parallel retrieval of synonym, hyponym ,hypernym and RMS sets for terms.Each word in the WordNet dictionary is represented by a unique integer and all the mappings are stored as integer mappings. This integer mapping allows very fast processing. The use of cache, separate databases and thread pool ensure maximum scalability to the system.

IV. CONCLUSION

In this paper. a novel approach called Wordster approach is used to build an efficient reverse dictionary.. In order to improve efficiency, the proposed method combines the features of forward dictionary and reverse dictionary. In the proposed system, instead of just returning the words that matches in meaning with the phrase, it returns the word along with their meaning so that the user is able to check the accuracy of the results obtained from the reverse dictionary. The Wordster approach can provide significant improvements in performance scale without sacrificing the quality of the result.

ACKNOWLEDGMENT
The authors gratefully acknowledge the support and facilities provided by Department of CSE, Vedavyasa Institute of Technology.

REFERENCE

[1] Dr. John M. Jun, “IJCSNS International Journal of Computer Science and Network Security”, VOL.8 No.11, November 2008. | [2] Chu-Ren Huang and Dan Jurafsky,”In Proc. of the 23rd International Conference on Computational Linguistics(Coling 2010),ICLL | [3] Naoaki Okazaki and Jun’ichi Tsujii,”Simple and Efficient Algorithm for Approximate Dictionary Matching”,Coling 2010,ICLL,pp 851-859 | [4] Dietterich, ”Machine Learning Research” , vol. 3, pp. 993-1022, Mar.2003. | [5]M. Porter, “The Porter Stemming Algorithm,” http://tartarus.org/martin/PorterStemmer/, 2009. | SITE References | [6] http://dictionary.reference.com/reverse | [7] http://www.onelook.com/ |