WordNet Based Online Reverse Dictionary with Improved Accuracy and Parts-of-Speech Sorting

Asha Chandran T [1], Sajina K [2]

[1] *Lecturer In Computer Engineering, Department of Computer Engineering, Govt. Womens' Polytechnic College Kayamkulam, Kerala, India*
[2] *Lecturer In Computer Engineering, Department of Computer Engineering, Govt. Polytechnic College Neyyattinkara, Kerala, India*

---***---

Abstract - *Reverse dictionaries are the ones which map a phrase or short sentence to one or more words whose meaning matches these phrases. The input to a reverse dictionary problem is a regular forward dictionary itself. But simply mapping each meaning phrase in a forward dictionary does not serve the complete requirement for a reverse dictionary. This is because the user input phrase may not be exactly similar to that one in a forward dictionary. The input phrase may contain synonyms of the words in the meaning phrase. So the mapping problem aims to find a set of words in the forward dictionary which contains words or synonyms of the words in the meaning phrase. The reverse dictionary is particularly useful for writers, journalists, data mining experts and even to the general public to find a suitable single word for a phrase in their ideas. The paper aims to refine the RD output to include most similar words and eliminate unrelated words.*

Key Words: Synset, Ranking , Forward Mapping Sets (FMS), Reverse Mapping Set (RMS), Parts-of-Speech, WordNet , Query

1.INTRODUCTION

Effective writing is an important concern to many categories of people like professional writers, students, scientists, teachers, marketing and advertisement professionals etc. Clarity and brevity are two important factors of effective writing. Brevity means being precise on the topic and clarity means using the apt words instead of beating the bushes.

Usually people find it difficult to choose the suitable words in particular contexts. In fact, for most people with a certain level of education, the problem is often not lacking knowledge of the meaning of a word, but, rather, being unable to recall the appropriate word on demand. Here comes the role of a reverse dictionary (RD). Effectively, the RD addresses the "word is on the tip of my tongue, but I can't quite remember it" problem.

A forward dictionary contains a set of words and each word has a set of definition phrases. To identify a single word equivalent to a phrase entered by the user, we need to compare the phrase with each definition in the dictionary, i.e., we need to find the semantic similarity between the user input phrase and dictionary definitions. Semantic similarity has gained great importance over long time in many fields such as artificial intelligence, natural language processing, data mining etc. By addressing the RD creation problem, we develop better methods for semantic similarity identification.

These methods can be used for a wide range of applications such as question-answer systems, plagiarism detection, meta data mining, document compression, document classification etc.

A regular (forward) dictionary maps words to their definitions, whereas a RD performs the converse mapping, i.e., given a phrase describing the desired concept, it provides words whose definitions match the entered definition phrase. For example, suppose a forward dictionary informs the user that the meaning of the word "spelunking" is "exploring caves." A reverse dictionary, on the other hand, offers the user an opportunity to enter the phrase "check out natural caves" as input, and expect to receive the word "spelunking" (and possibly other words with similar meanings) as output.

To construct a forward dictionary, we need a forward dictionary at our disposal. Upon receiving a search concept, the RD consults the forward dictionary and selects those words whose definitions are similar to this concept. These words then form the output of this RD lookup. The problem reduces to a concept similarity problem (CSP) .The CSP is a well-known hard problem which has been addressed in a number of ways with a limited degree of success. The RD problem can be thought of as a real-time online concept similarity problem.

But there are many key differences between RD problem and CSP which make direct use of existing results infeasible.

1.1 Related Works

Most of the related works fall into the concept similarity identification attempts. In concept similarity identification, the domain may or may not be predefined. In the case of domain specific applications, systems may have a higher rate of accuracy since the list of words/concepts in specific domains will be more or less fixed. The concepts, in such cases, will be easily distinguishable and hence less effort may be needed for sense disambiguation. For the same reasons, domain specific ontologies may converge faster to output. But in RD creation, the input comes from real world instances and similarity checking tends to be generic in nature and hence no specific restrictions are placed in domains.

Concept vector creation and similarity calculations based on cosine of vectors[3] is a domain specific approach to measure similarity between concepts. Here words are considered as concepts. An m-node hierarchy in a corpus can be mapped to an m-node concept hierarchy with each node in the hierarchy having an m-dimensional concept vector. The method, while being suitable for domain specific applications, still bears the load of constructing high dimensional vectors. Also high dimensionality of vectors cause the sentences represented as sparse. Latent Semantic Indexing [6] is a mathematical technique used to reduce the dimensionality of concept vectors without sacrificing their quality. LSI is thus a dimensionality reduction technique.

In the case of multiword similarity, the works are usually concentrated on paragraphs matching. The paragraphs are considered as short documents itself. In such cases, the system has sufficient contextual information to compare [8]. Hence these methods cannot be directly applied in RD creation.

Similarity identification between phrases or short sentences can be done using bipartite matching [2]. This work is more relevant in RD construction since inputs to RD also lie in the category of 'short sentences'. The system assumes words in each sentence to be matched as nodes in a bipartite graph. The edges between nodes are weighted using any of the word similarity computation methods. Overall similarity between sentences is measured by determining maximum bipartite matching between the two sets.

Another approach identifies similarity between sentences by constructing a word order vector and similarity vector[5]. A word order vector is constructed from the joint word set of the sentences to be compared. A raw semantic vector is constructed from the lexical database and the joint word set. A semantic vector is constructed from the raw semantic vector and the corpus. Semantic similarity is calculated from the semantic vectors of the sentences. Similarly, order similarity is calculated from the order vectors of the sentences. Overall similarity is calculated from the semantic similarity and the word order similarity.

Many other works try to identify similarity between sentences or phrase by calculating similarity between their constituent words. Similarity identification methods are broadly classified into corpus based and knowledge based. A number of such methods are described in [4].

1.2 Existing Reverse Dictionary techniques

Wordster Reverse Dictionary based on semantic relationships

The reverse dictionary system under our consideration is Wordster Reverse Dictionary (WRD) [1]. The system takes a phrase entered by the user as input and constructs a query based on it. The query is a Boolean expression containing words other than stop words connected using the logical operand AND.

A. Steps in RD Execution

a. RD Creation.
b. Input query generation from user input phrase.
c. RD querying (Query Execution)
d. Probable Query Expansion (if sufficient results are not obtained).
e. Ranking of candidate output words
f. Sorting of candidate words based on ranking

RD Creation

To start with, we need to create the reverse mapping of a forward dictionary. We construct reverse mapping set for the forward dictionary as follows. For every word w in the forward dictionary, the algorithm takes each sense phrase (meaning phrase) of the word. Each word in the sense phrase is stemmed to its root form. The popular Porter Stemming algorithm [9] is used for this. The word w is then included in the reverse mapping set (RMS) of each of the stemmed word.

RD Querying

Next, we need to query the RMS to find the possible output words (candidate words) for an input phrase. For this, the user input phrase U has to be modified to the form of a query. The GenerateQuery algorithm does this as follows. From the user input phrase U, the level 1 stop words are removed. The remaining terms are connected using AND to form a Boolean expression Q. Q is again modified by expanding antonyms present in it.

The Expand Antonyms algorithm takes a query as input and if any negated terms such as not, nor, neither etc. are present in the query, antonyms of the succeeding word is retrieved. The negated term ti is replaced with its antonyms connected using OR. The new query is termed as Q'. The GenerateQuery algorithm takes the output Q' and combine it to Q using OR. Each term ti in Q is stemmed to get ti^ and in Q, it gets replaced as (ti OR ti^). The terms in Q is reordered such that all nouns appear before verbs and all verbs appear before adjectives and adverbs. By this step, the user input phrase is converted to the RMS query Q.

The query can now be executed. The ExecuteQuery algorithm simply takes set intersection of the RMS of the constituent terms. Since the RMS now contains only integer values corresponding to the word id s, this is a fast arithmetic operation. This algorithm returns a set of word id s of the candidate output words.

In case if the ExecuteQuery algorithm does not return sufficient number of candidate words(α) as specified in the application, the query is expanded using other conceptually similar terms such as synonyms, hyponyms, hypernyms etc. The ExpandQuery algorithm takes each term ti in the input query Q and replace ti with (ti OR syn(ti)) or (ti OR hypo(ti)) or (ti OR hyper(ti)) according to the argument type being synonym, hyponym, hypernyms respectively. The expansion process is done sequentially starting from synonyms set. After expanding using synonyms, the algorithm checks whether the required no. of results are obtained. If yes, they are sorted. Otherwise, expansion is done using the second type, i.e, hyponyms, and the process is repeated. Finally expansion using hypernyms are done.

If none of these expansions help in retrieving required number of terms, then the GenerateQuery algorithm computes the cardinality of RMS of each term ti . The terms are then sorted in the decreasing order of the cardinality and terms with highest cardinality are removed from the query. Again, the algorithm checks whether the required no α is reached. If not, the above process is repeated until the number of terms is reduced to two. Once α output words are obtained, they are sorted.

The reason for removing terms with higher cardinality is that they reduce the probability to get enough word id s while taking intersections. The removal is done until there only two terms, because it needs at least two sets to perform an intersection or union.

Ranking and Sorting the Candidate words

The output words are to be sorted on the decreasing order of their similarity with the input phrase. The SortResults algorithm achieves this by considering two factors of each output words- term similarity and term importance. Term similarity ρ is computed between every pair of terms (a,b) where a ε S and b ε U. Term importance λ(a,S) indicate how critical the term a is in the context of the phrase S. Also, λ(b,U) is calculated. Similarity measure between S and U, μ(a,S,b,U), is calculated as the product of these three terms ρ(a,b), λ(a,S) and λ(b,U).

Architecture for Wordster Reverse Dictionary

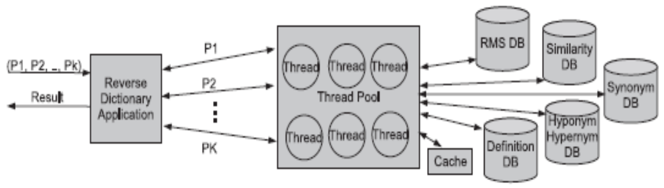

Fig-1 Dictionary Architecture

The input query is passed to the RDA module which take each valid term in the query and access the database for the term's RMS, synonyms, hyponyms and hypernyms. The database is constructed from WordNet 3.1 version. A pool of threads accomplish the simultaneous access of RMS and relationship sets. This makes a faster and parallel execution of the query possible. A cache is used to store these databases. According to the size of the database, cache size is designed to accommodate these sets.

OneLook.com and Dictionary.com

OneLook.com [11] and Dictionary.com [7] are two reverse dictionary systems which are already available online. These systems provide up to a maximum of 100 output words matching to the user input phrase. But the quality of the solution of the Wordster approach is better than these from a set of experimental results.

2. PROPOSED REVERSE DICTIONARY SYSTEM

Consider an input phrase, 'study of sound'. The expected result is 'acoustics' .When evaluating the direct expression, we get the candidate words 'acoustics, 'echogram' and 'echography'. These results are perfectly acceptable since they all lie in the category of study about sound.

Since we got only three candidate words and the user requires at least ten words, we go to the indirect expression. Indirect expressions are formed using synonyms, hyponyms and hypernyms of the input words. Now let us see the candidate words from the result of indirect expression evaluation.

The indirect expressions are constructed as follows:

(Study OR synonym(study)) AND ((sound OR synonym(sound))
If sufficient results are not obtained, the hyponyms are used for query.
(Study OR hyponym(study)) AND ((sound OR hyponym(sound))
If still we don't obtain sufficient results, hypernyms are tried.
((Study OR hypernym(study)) AND ((sound OR hypernym(sound))

Accordingly we get many indirect expressions since a word can have many synonyms, hyponyms and hypernyms.

Surprisingly, the only one result from one of the indirect expressions is the word 'Aquinas' which is the name of an ancient Italian theologian!!!

2.1 Reason for Unrelated Output words

We can see that even after several similarity checking and sorting based on relevance, the output contain certain erroneous terms. This may arise because we are trying to obtain set intersections of the RMS s of the terms in the user query. If in case a more appropriate word appears in the intersection of two of the synset id sets out of three sets, for example, but not in the third set, that candidate word is eliminated in the existing algorithm.

When a query is executed, if sufficient numbers of candidate words are not obtained, the query is expanded to include synonyms, hyponyms and hypernyms of the constituent terms respectively. While including hypernyms of a term, we are traversing up in the WordNet hierarchy. Going higher, the terms tend to be more general in nature. This may result in a higher probability for obtaining results in set intersection, since these terms, being general in nature, will appear in the dictionary definitions of many words.

This cause a more appropriate word to be discarded from the list and more general or unrelated words to be added in the list. Also, when the user enters a phrase, if we can determine which Parts-of-Speech is used in the input phrase, we can sort the candidate words according to this Parts-of-Speech. This will help in reducing user frustration when he/she expects a noun and the returned output is an adjective or verb.

Our efforts are pointed towards eliminating these problems.

2.2 Methods to Eliminate Unrelated words from Output

Elimination of an appropriate word can be avoided by slightly changing the ranking process so as to accommodate issue inherent in the set intersection problem. The ranking has to be modified such that a candidate word that appears in m out of n RMS sets of the words in the input query will be assigned an appropriate rank.

2.3 New Ranking Algorithm

Inputs: User input phrase with stop words removed.

Output: A set of candidate words whose dictionary definition matches the user input phrase.

Database: words, RMS, synsets, synonyms, hyponyms and hypernyms from WordNet 3.1

Let the user input phrase U consists of n words after removing stop words. Let the word IDs corresponding to the n words be $w_1, w_2, \ldots w_n$. RMS s of each of these terms are named as $S_1, S_2, \ldots S_n$. Each of these sets will be a set of word IDs $t_1, t_2, \ldots t_m$. Let the user requires α candidate words.

Algorithm:

1. Execute the direct expression, i.e., take set intersection of RMS of each term in the input query.
2. Check if the required numbers of candidate words are obtained. If not goto step 3
3. For i=1 to n, do steps 4 and 5.
4. For each $t_i \varepsilon S_i$, do the following step.
5. If t_i is present in S_j, assign $rank(t_i)=rank(t_i)+1/n$. [$rank(t_i)$ is initialized to zero.]
6. Select all $t_i \varepsilon S_i$ with $rank(t_i)>$ preset threshold. Let the new set be C with cardinality m .
7. For i=1 to m, do the following.
8. For j=1 to n, do the following.
9. Check if $t_i \varepsilon S_j$. If not, take the word id w_j corresponding to the set S_j.
10. Replace S_j with synonym set of w_j and Check if $t_i \varepsilon S_j$.
11. If yes, set $rank(t_i)=rank(t_i)+(1/2)n$ Else goto next step.
12. Replace S_j with hyponym set and Check if $t_i \varepsilon S_j$.
13. If yes, set $rank(t_i)=rank(t_i)+(1/3)n$
14. Sort the terms based on descending order of their rank.
15. Return the candidate words of the terms.

2.4 Pseudocode

```
1.  Construct the query Q from the user input phrase U .
    $Q=t_1$ AND $t_2$ AND …AND $t_n$
2.  Execute the direct expression, i.e., take set intersection of
    RMS of each term in the input query.
    $O=RMS(t_1) \wedge RMS(t_2) \wedge \ldots \wedge RMS(t_n)$
3.  If $|O|>= \alpha$ goto step 14. Else goto next step.

4.  For i=1 to n
        For j=1 to m
            If $t_j \varepsilon S_i$
                Set $rank(t_j)=rank(t_j)+1/n$.
            [$rank(t_j)$ is initialized to zero.]

5.  For each Si in the set [ i:1…n]
        For j=1 to m
            If $rank(t_j)>$ preset threshold and $t_j$ not in O
                Add $t_j$ to output word set O.
            Else add $t_j$ to candidate word list C.
6.  If $|O|>= \alpha$ goto step 14 . Else goto next step.
8.  For i=1 to |C|
        For j=1 to n
            If $t_i$ not in $S_j$
            take the word id $w_j$ corresponding
             to the set $S_j$.
                    Replace $S_j$ with synonym set of $w_j$.
                    If $t_i$ ε Syn($w_j$)
                     {
                        rank($t_i$)=rank($t_i$)+(1/2)n
                              goto step 11
                     }
                    Else
                     {
                     Replace $S_j$ with hyponym set of $w_j$.
                     If $t_i$ ε Hypo($w_j$)
                       {
                        rank($t_i$)=rank($t_i$)+(1/3)n
                              goto step 11
                       }
                     }
                    Else
                    {
                    Replace $S_j$ with hypernym set of $w_j$.
                    If $t_i$ ε Hyper($w_j$)
                        Rank($t_i$)=rank($t_i$)+(1/4)n
                    }
9.    For i=1 to |C|
        If rank($t_i$)> preset threshold
            Add ti to O.
10.   Sort the terms in O based on descending order of their
      rank.
11.   Return the sorted list of candidate words .
```

The algorithm searches the semantic relationships for only those words whose RMS does not give any results for set intersection with the RMS sets of other words. Words in the input query whose RMS contain common candidate word ID s are not checked further for synonym, hyponym and hypernyms relations. Hence unnecessary reduction of accuracy is avoided.

2.5 Identifying Parts-of-Speech from User Input Phrase

WordNet contains nearly 1,55,300 words. Out of these nearly 1,17,800 words are nouns. Of the remaining, 11500 words are verbs , 21200 are adjectives and 4500 are adverbs approximately.

Since nouns contribute most of the vocabulary, there is a higher probability that the expected word to be a noun. We assume that if at least half of the valid words in the query are nouns, the expected parts-of-speech is a noun. In case an adjective is expected, the query will contain either adjective words or adjective indicators like ‘being’, ‘having ’,’like’ etc. Similarly, we can identify whether user expects a verb or adverb according to the parts-of-speech in which most of the words in the query fall in.

Once the expected parts-of-speech is identified, the list of candidate words obtained as the result of our new ranking algorithm can be further refined such that those words with the expected parts-of-speech come first in the list. This may help to provide the user with an output that matches his/her requirements as far as possible.

3. CONCLUSIONS

The paper describes how a reverse dictionary can be built, used and improved to match the user requirements. The paper proposes a new ranking algorithm that eliminates generic words and unrelated words from the candidate output word list. Also, it ensures that an eligible similar word will be added to the candidate words list in the appropriate position. Identification of parts-of-speech is done by reading between the lines. This also improves the quality of the output.

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 Sentences, ”http://opensvn.csie.org/WordNetDotNet/trunk/Projects/Thanh/Paper/WordNetDotNet_Semantic_Similarity.pdf Oct. 2009), 2009

[3] J. Kim and K. Candan, “Cp/cv: Concept Similarity Mining without Frequency Information from Domain Describing Taxonomies,”Proc. ACM Conf. Information and Knowledge Management,2006.

[4] R. Mihalcea, C. Corley, and C. Strapparava, “Corpus-Based and Knowledge-Based Measures of Text Semantic Similarity,” Proc.Nat’l Conf. Artificial Intelligence, 2006

[5] A Yuhua Li, David McLean, Zuhair A. Bandar, James D. O’Shea, and Keeley Crockett ” Sentence Similarity Based on Semantic Nets and Corpus Statistics”, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 18, NO. 8, AUGUST 2006

[6] Barbara Rosario,”Latent Semantic Indexing: An overview” Infosys 240 Spring 2000 Final Paper

[7] Dictionary.com,LLC,“Reverse Dictionary,”http://dictionary.reference.com/reverse, 2009.

[8] 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.

[9] MPorter, "The Porter Stemming Algorithm,"http://tartarus. org/martin/PorterStemmer/, 2009.

[10] O.S. Project "Opennlp," http://opennlp.sourceforge.net/, 2009.

[11] OneLook.com,"Onelook.comReverseDictionary,"

[12] U. of Pennsylvania, "The Penn Treebank Project," http://www. cis.upenn.edu/ treebank/, 2009.w.onelook.com/, 2009.