Database Driven Reverse Dictionary

[1]Prof.Pravin Adivarekar, [2]Nitin Kumar Singh, [3]Shraddha Shanbhag, [4]Renuka Jagtap

[1]Asst. Professor, [2,3]BE Student, [1,2,3]Comp. Engg. Dept, SSJCET, Asangaon, India.

[1]engineerpravin2008@gmail.com, [2]nitinsingh7860@rocketmail.com, [3]shraddha95@yahoo.com, [4]rjrenuka01@gmail.com

Abstract - The Forward Dictionary is used to get the definition from the user entered words whereas in the Reverse Dictionary the user enters the phrase or sentence and gets the words similar to the concept of the input phrase. The Reverse Dictionary is a non-alphabetical dictionary. At times people go on describing the situation in long sentences. Reverse dictionary gives a precise and appropriate word to one's thoughts. In this paper, our system provides higher quality improvements in the performance. In the proposed system, user can add his own words along with its descriptions which he can use for his own work in near future.

Keywords —Forward Dictionary (FD), Reverse Dictionary (RD), Reverse Mapping, Forward Mapping, Stemming, Candidate words

I. INTRODUCTION

Dictionaries are one of the most popular types of books available today. From the first years of primary school to the last years of life, any user of a language will consult a dictionary as part of day to day life[3]. When using a dictionary, the user will look up a single word, known as the headword (which may consist of more than one word separated by spaces, known as a compound word) and will be given a definition, which can be defined as a set of descriptor words[4].

A. Reverse Dictionary

Forward dictionary is the one which maps from word to their definitions[3]. For example: 'PEDANT'-One who makes display of his learning. the function of a dictionary can be described as:

lookup(h) =d

A reverse dictionary is a collection of concepts or phrases, the user enters the concept as the meaning receiving a set of words as output Reverse dictionary perform reverse mapping i.e. the user enters phrase and the words are output equivalent to the entered phrase[10]. For example, in forward dictionary the user gets the meaning of the word "inaudible" as "incapable of hearing". In reverse dictionary, the user gets an opportunity to enter "cannot hear" as input, and gets output as "inaudible" and possibly other words with similar meaning as output. A reverse dictionary functions by returning a headword given a set of descriptor words:

lookup(d)=h

In this approach, the reverse dictionary is used to predict the similarity between the concept and phrases. The Reverse mapping set is created. The basic steps to create the reverse dictionary are the same for all languages, only the techniques to extract the meaning differs[10]. Such a reverse dictionary would be helpful for students, professional writers, teacher, etc.

B. Aim and objective

The aim is to create a working reverse dictionary. The aim of this report is to describe the design, implementation and results of the reverse dictionary as well as explaining the issues encountered during its completion and how they were addressed.

Objective

- More accurate the data more accurate the result.
- This approach can provide significant improvements in performance scale without sacrificing solution quality
- Objective in proposed is to provide efficient reverse dictionary to users like students, teachers, writer, etc.
- A reverse dictionary takes a user input phrase describing the desired concept, it reduce the well-known conceptual similarity problem.
- Objective here is not only to have correct reverse dictionary but to also have a dictionary in users own language.
- Adding new feature like word searching enhances the dictionary

II. LITERATURE SURVEY

Let's take analysis of different proposed methodologies and our proposed method. Various popular methods are:-

TABLE 1 :LITERATURE TABLE

TECHNIQUE	REMARK	EXAMPLES
.Pos Tagging[10][2]	Part-Of-Speech Tagging is used to assign parts of speech to each word. Pos tagger performs pos tagging. The natural languages contain lot of ambiguity, which can make distinguishing words difficult so pos tagger is used.	The two types of pos tagger are Brill's tagger and Stanford POS tagger.
.Stemming [0][2]	Stemming algorithm is a computational procedure which reduces all words with same stem to a common form by stripping each word of its derivational suffixes. Comparison of stemmed words becomes easier and in addition storing becomes easier.	The words with common stem: Connect Connecting Connection Connected
3.Stop Words[2].	These are the words which do not tell much about the content but helps humans to understand the contents .So removing these words will increase the accuracy of the search results and reduce the search time.	List of stop words: about above across after again against all
4.Candidate Words[7].	To find candidate words phase consists of two key sub-steps: 1) build RMS 2) query RMS. Then rank these words in the order of quality of match.	Input : Present in large Quantity Candidate words : Ample, bigger, larger, wide, bulky, tremendous

III. PROPOSED SYSTEM

A reverse dictionary is a dictionary which is kept in reverse order (usually referring to being in a so-called "reverse" order) that provides the user with that would be difficult to obtain from a traditionally alphabetized dictionary[5]. Reverse dictionary can be of two types reverse word dictionary and reverse conceptual dictionary.

The reverse word dictionary is a dictionary in which the words are not in alphabetical order as traditional dictionary. Reverse concept dictionaries are those where the concept is taken as input and set of words defining the input are given as output[4]. Reverse dictionaries have been published for most major alphabetical languages. In Reverse dictionary user enters any logical phrase and gets number of words as output[10]. for example: if user enters "A strong smelling colorless gas" and get "Ammonia", "nitrogen" "anonymous" as output.

Reverse dictionary is based on the concept that a phrase entered by user if not exactly matching the word then they should be at least conceptually similar[5]. Building a ranking based dictionary, helps user to choose from a set of words which are closely related to each other[4]. Hence it becomes easy and time saving for user to choose words from such set. Example: RMS for "BELIEVING IN GOD, RELIGION" will be faith, trust, reliance, loyalty, honesty these words are ranked based on their concept similarity. The stop words like "about, above, across" etc is ignored as they don't tell much about the content.

Forward mapping (standard dictionary): A forward mapping applies all the senses for a particular word phrase. This can be expressed in the form of a forward map set (FMS) [6]. For example, suppose that the term "jovial" is associated with various meanings, including "showing high-spirited merriment" and "pertaining" to the god Jove, or Jupiter."

Reverse mapping (reverse dictionary): Reverse mapping is applied to terms and can be expressed as a reverse map set (RMS)[6]. To build an RMS means to find a set of words in whose definitions any word "W" is found. Building an RMS means to find a set of words which are conceptually closer to the user input phrase. When the user input do not get enough output. The RMS of words also form a part of the output words.

In existing reverse dictionary user used to get over a 100 search results as output for any input. Reverse Dictionary also uses the concept of Stemming. It is a process of

obtaining the root form of the word [2]. For example: the root form of the word cleared is "clear" also words like clearing, clears should be converted to their root form clear.

Thus, the reverse dictionary which uses the concept of building reverse mapping set is built. In existing dictionary user enters the phrase and get number of words as output but sometimes it becomes time consuming for user to choose one word from it. Hence building a ranking based dictionary helps user to get the output words easily without consuming lot of time [4].

IV. SYSTEM ARCHITECTURE

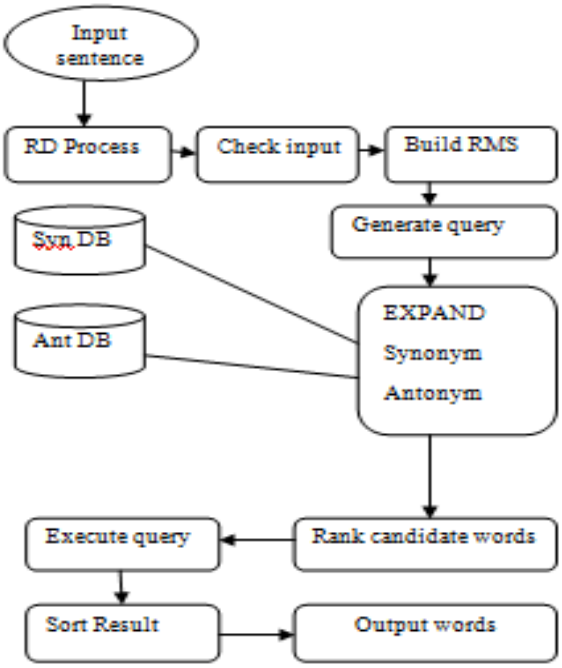

FIG 1: SYSTEM ARCHITECTURE

The implementation architecture of reverse dictionary application is as [2] [4]:

- Take Input Phrase from user.
- Split the user input phrase and perform stemming technique so as to obtain root form of each word.
- All the stop words are removed to get the better performance. Stop words are those whose removal does not hamper the performance of the process.
- Reverse Mapping Set is built according to the input phrase.
- Query is generated to get respective set of synonym words, set of Antonym words & set of Hypernym words. Each of these set of words is connected to their respective databases. All these databases are parsed to get the appropriate output.
- Syn DB **(**Synonym Database): It contains the set of synonym words
- Ant DB (Antonym Database): It contains the set of Antonym words.
- Identify and rank candidate words based on user input.
- Execute the query.
- Return output words to user

V. ALGORITHMS IMPLEMENTATION

At a high level, our approach consists of two sequential steps [1]. Upon receipt of a user input phrase, first find candidate words from a forward dictionary data source, where the definitions of these candidate words have some similarity to the user input. Then rank the candidate words in order of quality of match. The find candidate words phase consists of two key sub steps: 1) build the RMS; and 2) query the RMS.

1. BuildRMS

1: Input: a dictionary D.
2: for all mapping Wi → Sj belongs to D do
3: Extract the set of terms {t1 . . . tx}, tk belongs to SPj
4: for all tk do
5: Apply stemming to convert tk to convert it to its general form tk*
6: Add Wi to R(tk*) and increment N(tk*) by 1.

2. GenerateQuery(U, α, β)

1: U=U s.t.∀ti ∈ U, ti ∉ L1
2: Form a boolean expression Q by adding all ti∈ U to Q, separated by AND
3: Q = Q OR ExpandAntonyms(Q)
4: for all ti ∈ U do
5: Apply stemming to ti to obtain ti*
6: Replace ti in Q with (ti OR ti*)
7: Reorder terms in Q s.t. all nouns appear before verbs,and verbs before adjectives and adverbs
8: O = ExecuteQuery(Q)
9: if |O| >α then
10: Return SortResults(O)
11: for all ti ∈ Q s.t. ti ∈ L2 do
12: Remove ti from Q
13: O = O U ExecuteQuery(Q)
14: if |O| > α then
15: Return SortResults(O)
16: O = O [ExpandQuery(Q, "synonyms")

17: if |O| > α then
18: Return SortResults(O)
19: O =O [ExpandQuery(Q, "hyponyms")
20: if |O| > α then
21: Return SortResults(O)
22: O = O ∪ ExpandQuery(Q, "hypernyms")
23: if |O| > α then
24: Return SortResults(O)
25: Create a list l of all terms ti ∈ Q
26: Sort l in descending order based on N(ti)
27: while |Q > 2| do
28: Delete ti from Q, where ti has the highest value of N(ti) in l
29: O = O ∪ ExecuteQuery(Q)
30: if |O| > α then
31: Return SortResults(O)
32: else
33: Delete ti from l

3. ExecuteQuery(Q)

1: Given: a query Q of the form Q = t1 ⊕1 t2 ⊕2 t3 . . .tk-1 ⊕k-1 tk , where ⊕i ∈ {AND,OR}
2: Oe = R(t1) ⊗1 R(t2) ⊗2 R(t3) . . . R(tk-1) ⊗k 1 R(tk),where if ⊕i = AND, ⊗i = ∩ and if ⊕i = OR, ⊕i = ∪
3: return Oe

4. ExpandQuery(Q, SetType)

1: Given: a query Q of the form Q = t1⊕1 t2⊕2 t3 . . .tk-1 ⊕k-1 tk, where ⊕i ∈ {AND, OR}
2: for all ti ∈ Q do
3: if SetType is "synonyms" then
4: F =Wsyn(ti)
5: if SetType is "hyponyms" then
6: F =Whyo(ti)
7: if SetType is "hypernyms" then
8: F =Whyr(ti)
9: Create a new subquery q = (ti)
10: for all tj ∈ F do
11: Add tj to q, connected with OR
12: Replace ti in Q with q
13: Return ExecuteQuery(Q)

5. SortResults(O, U)

1: Create an empty list K
2: for all Wj ∈ O do
3: for all Sk ∈ W do
4: if ∃ti ∈ Wj s.t. ti ∈ L1 then
5: Remove ti from Wj
6: Compute Z(S) and Z(U)
7: for all pairs of terms (a, b), where a ∈ Z(S) and b ∈ Z(U) do
8: if a and b are the same part of speech in Z(S) and Z(U), respectively then
9: Compute ρ(a,b)= 2∗E(A(a,b))/(E(a)+E(b))
10: Compute λ(a, S) = (d(Z(S))-da)/d(Z(S))
11: Compute λ(b,U) = (d(Z(U))-db)/d(Z(U))
12: Compute μ(a,S,b,U) = (a, S) ∗λ(b,U) ∗ρ(a,b)
13: Use μ(a, S, b, U) values to measure the phrase similarity M(S,U) of S and U following the algorithm described in
14: Insert the tuple (S,U,M) in to the list K
15: Sort the tuples in K in descending orderbased on the value of M
16: For the top _ word senses in K, return the corresponding word.

TABLE 2: NOTATION

NOTATION	MEANING
T	any legitiment word in the english language
*t**	stemmed (generic)version of *t*
P	a sequence of one or more words
D	a set of mappings *P*→P
W	a word phrase
S	a sense phrase
F(W)	a forward dictionary ,i.e., a set of mappings *W*→S
R(t)	reverse mapping for *t*, i.e., all the w's that include *t* in their definitions
N(t)*	count of definitions in which a stemmed term *t** appears
Wsyn(t)	set of synonyms of *t*
Want(t)	set of antonyms of *t*
Whyr(t)	set of hypernyms of *t*
Whyo(t)	set of hyponyms of *t*
U	a user input phrase
G	set of negation words
Q	boolean expression query, based on *U*
L1	the set of level 1 stop words
L2	the set of level 2 stop words
O	set of output *W P*s satisfying *Q*
A	minimum threshold number of *W P* ∈ O required to stop processing
B	maximum number of *W P* to include in output
ρ(t1,t2)	term similarity of the terms *t*1 and *t*2
λ(t,P)	importance of *t* in *P*
Z(P)	parse tree of a phrase *P*
Dt	depth of t in a phrase *P*
d(Z(P))	overall depth of a parse tree *P*

VI. MATHEMATICAL MODEL

The mathematical model of reverse dictionary application is as [5].

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.

VII. RESULTS AND DISCUSSION

This paper concentrates on only training part. When the number of the resultant output is less then the accuracy is more. Below given graph shows the accuracy in the performance of the system. The graph is based upon the number of suggestions of output words and accuracy of the system in terms of percentage.

TABLE 3: *RESULT OF RANKING WORDS*

Number of suggestions	1	2	3	4	5	6
Accuracy percentage	100	99	95	94	90	90

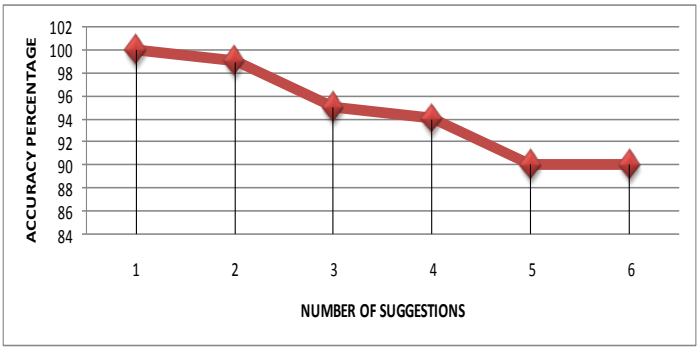

FIG 2: GRAPH OF THE SYSTEMS PERFORMACE

VIII. CONCLUSIONS

Depending upon the user input phrase candidate words are found and ranked according to the quality of match to give output. If the user doesn't find result of his phrase then, he can add the words with the appropriate meaning to the system

These added words can be further used by user in his work. Thus, adding new feature such as adding words with its meaning by the user itself for his work enhances the previously available reverse dictionary.

ACKNOWLEDGEMENT

This project consumed huge amount of work, research and dedication. It would not have been possible if we did not have a support of many individuals and organizations. Therefore we would like to extend our sincere gratitude to all of them.

I am using this opportunity to express my gratitude to everyone who supported me to publish this paper . I am thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the work. I am sincerely grateful to them for sharing their truthful and illuminating views for preparing this work.

REFERENCES

[1] Ryan Shaw, Member, IEEE, AnindyaDatta, Member, IEEE, Debra Vander Meer, Member, IEEE, and Kaushik Datta, Member, IEEE ,"*Building a Scalable Database-Driven Reverse Dictionary*", 2013.

[2] E.Kamalanathan1 and C. Sunitha Ram, "*Implementing a Higher Quality Input Phrase To Driven Reverse Dictionary*", International Journal of Advance Foundation and Research in Computer (IJAFRC)Volume 1, Issue 4, April 2014.

[3] Akanksha Tiwari1, Prof. Rekha P. Jadhav, International Journal of Modern Trends in Engineering and Research, "*Survey On Building A Database Driven Reverse Dictionary*".

[4] PriyankaD. Nanaware1 Rahul. K. Ambekar, International Journal of Advanced Research in Computer Science and Software Engineering **, "***Building and Improving Scalable Database Driven Reverse Dictionary*", Volume 4, Issue 7, July 2014.

[5] PriyankaD. Nanaware1 Rahul. K. Ambekar International Journal of Engineering and Advanced Technology, "*Enhancing Scalable Database-Driven Reverse ictionary*" Volume-3, Issue-5, June 2014.

[6] Nagesh Racha,Varsha Balghare,Sabir Khan,prof.Seema Bhardwaaj, "*A Proposed System for A Scalable Database Driven Reverse Dictionary"* Volume 3, Issue 2, February 2015

[7] E.Kondala Rao, M.S.R.Lakshmi Reddy"*Exposure Towards Accuracy Enhancement for Retrieval of Information*" Volume No.2, Issue No.5, August – September 2014, 1260 – 1263.

[8] Jincy A K, Sindhu L, "*A Survey of Building a Reverse Dictionary*", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (6), 2014.

[9] Kesavaram.P.H, T.Muthamilselvan, " *Building a ranking based reverse dictionary*" , International Journal of Innovative Research in Computer and Communication Engineering Vol. 2, Issue 4, April 2014.

[10] D.Gaya, "*A Study Of Building An Reverse Dictionary*", International Journal of Scientific and Research Publications, Volume 5, Issue 7, July 2015.

[11] R. Mihalcea, C. Corley, and C. Strapparava, "*Corpus-Based and Knowledge-Based Measures of Text Semantic Similarity*," Conf. Artificial Intelligence, 2006.

[12]. M. Porter, "*The Porter Stemming Algorithm*,"http://tartarus.org/martin/PorterStemmer/,2009.

