Concurrent Context-Free Framework for Conceptual Similarity Problem using Reverse Dictionary

V. Nageena
Computer science Engineering
UCEK-JNTUK
Andhra Pradesh, India
nageena29@gmail.com

Ch. Ratna kumari
Computer science Engineering
UCEK-JNTUK
Andhra Pradesh, India
ratnamala3784@gmail.com

***Abstract*--**Semantic search is one of the most prominent options to search the required and relevant content from the web. But most of them are doing key word and phrase wise similarity search. It may or may not find the relevant information because they directly search with that phrase. But, in most of the cases documents may conceptually equal instead of term wise. Reverse dictionary can solve such type of problems. This will take meaning of the word and it will return related keywords with respective ranks. But main problem here is building such dictionaries is time and memory consuming. Cost effective solutions are required to reduce search time and in-memory requirements. This paper focuses on such aspects by utilizing concurrent programming and efficient index structures and builds a framework to Conceptual similarity problem using reverse dictionary. Simulation results shows that proposed approach can take less time when compared to existing approaches.

Keywords: *semantic search, reverse dictionary, concurrent programming, stemming, bloom filters, nlp*

*****

I. INTRODUCTION

Search and Communication both are dependent words. Whenever we search for specific topic the system or other human has to retrieve the relevant information. Semantic Search and natural language processing is one of the wing of Artificial Intelligence. Sometimes user may give input like phrase or event like sentences rather than simple keywords. It may be wrong due to spelling mistake or context problem. For example 'week' is different from 'weak'. According to his expected context given spelling may be wrong, But there should be a system in such a way that the results should be both conceptually and sound wise similar.

People who work more with natural languages are affected heavily by this problem. Most of the people are having a problem to express their feelings in a single word instead of collection of sentences. In that case Reverse dictionary is useful. If any user is able to give a sentence as input then RD gives most prominent and related words with ranks which reflect the full or partial meaning of the given input. But building such dictionaries require more language specific knowledge such as grammar, vocabulary etc.

The CSP is a well-known hard problem [1] which has been addressed in a number of ways with a limited degree of success. The real-time, online concept similarity identification problem we need to tackle is different from what extent CSP work addresses. In effect, one of the core contributions of this work is the development and implementation of a practical and scalable (i.e., capable of supporting online interactive applications) concept similarity measurement system. Specifically, the two problems have key differences that make direct use of existing results infeasible.

II. PRELIMINARIES

A. OpenNLP Parser:

Parser is a software component that takes input data and builds a data structure often some kind of parse tree giving a structural representation of the input, checking for correct syntax in the process. OpenNLP is a machine learning tool kit for the processing of natural language text. It supports tokenization, sentence-segmentation, part-of-speech tagging, namely entity extraction, chunking, parsing and conference resolution.

B. Information Retrieval:

Information Retrieval [6] is a process which begins when an input query is given by the user. In this process, the query simply does not identify a single object in the Database. Instead, several objects may match the query, perhaps with different degrees of relevancy.

Most Information Retrieval systems compute a numeric score on how well each object in the Database matches the query, and rank the objects according to this value. The top ranking objects are then shown to the user. The process may be iterated if the user wishes to refine the query.

Index is a mechanism for locating a given term in a text. It is possible to find information without resorting to a page -by -page search. There are many ways to do indexing like signature files, bitmaps, inverted file indexing etc. Inverted

files offer better performance than signature files and bitmaps in most applications. An Inverted file indexing also requires a lexicon. A lexicon is a list of all terms that appear in the database. Lexicon supports a mapping from terms to their corresponding inverted lists.

C. Context-Free Parsing Algorithm:

Context-Free grammars are extensively used for describing the syntax of both programming languages and natural languages. Parsing algorithms for Context-Free grammars play a major role in the implementation of programs which understand or translate natural languages. This is an efficient Context-Free parsing algorithm [7] which is capable of handling large class of grammars in linear time. It is similar to both Knuth's LR (K) algorithm and the familiar top-down algorithm. Its time bound is proportional to n^3, where n is the length of the text being parsed; it has n^2 bound for unambiguous grammars

.

D. Word Net:

Word Net [3] is conceived as a machine readable dictionary. It is a popular lexical database used in NLP (Natural Language Processing). It constructs lexical information in terms of word meanings. Word Net maps word forms in word senses using the syntactic category as a parameter. Words of the same syntactic category that can be used to express the same meaning are grouped into a single synonym set called synset.

For example, the noun "*computer*" has a synset: {computer, data processor, electronic computer, information processing system}.

E. Query processing:

The simplest type of query is the Boolean query processing in which terms are combined with the connectives AND, OR and NOT. Every input query needs to be processed. It is straight forward to process such a query using inverted file index. The Lexicon is searched for each term; each inverted list is retrieved and decoded. Finally the documents so indexed are retrieved and displayed to the user as the list of answers.

If the user is not satisfied with the retrieved number of results, then in order to improve our search process, we make use of these connectives. With the help of these connectives we can improve our search process and retrieve required number of related results. A query can have one or many results [10] based on the context of our search. We can choose between them.

III. PROPOSED APPROACH

Motivation

There are some existing approaches to solve the problem of concept similarity as well as reverse dictionary. Those approaches build the reverse dictionary from the forward dictionary as one time task. Later on whenever user gives the input it searches the conceptually similar words based on synonyms, hypernyms, hyponyms and antonyms. Actual problem arises here. For every input phrase one has to search the related words in different databases which will increase the search time. Multi phrase or sentence input cannot be validated in the previous approaches. To reduce this complexity an efficient framework is required. Following sections describe the solution of above mentioned problems

Proposed approach is an extension of work proposed in [2]. But there are two new features added to that work. First, rather than searching all the databases which contains synonyms, antonyms, hypernyms and hyponyms sequentially, apply concurrent searching. Second, not only concept similarity but also sound wise similarity is also considered. First part of the proposed approach is further divided into two parts. Multi-phrase processing with concurrent execution and context-selection then search. The second part of first phase also uses the first part of first phase. Here is the description of each phase.

A. Multi-phrase processing

In this type, whenever user gives input Word net [3] sentence detect is used to identify and divide the sentences. Now these sentences are pipelined to concurrent process. At this stage one fork join task is created to each phrase and submitted to multi-core system. Every task has following responsibilities

- Tokenization of phrase
- Identify and removal of stop words
- Identification of negation words and search the antonyms of those words
- Fetching of synonyms, hypernyms, hyponyms and stem words
- Building of Bloom Filter for each term of the given phrase
- Initialization of parse tree threads to each term to find the relevance of the terms to given phrase
- Executing the query and generate the results with rank.

1. Tokenization of phrase:

Tokenization is a process of dividing the sentence into smallest individual units called tokens. Each phrase is dividing into multiple terms with space as a delimiter. These terms becomes input for further processing.

2. Stop word removal:

In the real time most of the words are occurred frequently. These words are less significant in searching. So removal of these words will increase the accuracy of the search results and reduce the search time.

List of stop words	a, be, person, some, someone, too, very, who, the, in, of, and, to, that, for, with, this, from, which, when, what, into, these, where, those, how, during, without, upon, toward, among, although, whether, else, anyone, beside, whom, onto, anybody, whenever, whereas.

TABLE 1: Stop Words

3. Replacement of negation words:

Some of the words such as not, none, etc are treated as negation words. They give different or opposite meaning when appended with other words next to them. In this case find the antonyms of the words next to these words and replace the negation word + next word (neg word) with antonym changes the sentence into more appropriate phrase for further processing.

4. Fetching of synonyms, hypernyms, hyponyms and stem words

Fetching the related words will increase the scope of the search and concept relevance probability. It will increase the word density. There is a problem here. Every word has different senses and different parts of speech. So we have two possibilities here. One is to fetch the words with all the senses or take confirmation from the user before query execution i.e., at the time of submission of input. Next part is stemming of each term i.e., conversion of the given word into its general form. Porter Stemming algorithm [4] is used in the proposed system. This will broaden the search space.

5. Building of Bloom Filter for each term of the given phrase

Every word has so many synonyms, hypernyms, hyponyms and stem words. Handling these words and searching these words in later process requires so many probes. Without having proper index structure will again increase the search space, search time and degrade the response time. To overcome this problem proposed approach uses Forest of bloom filters. Those are like a hash table of string trees. Whenever we want to probe word, calculate its hash value and extract the synsets, hypernyms, and hyponyms. Internally this is array of trees based on linked lists. So it is fully dynamic in nature. Proposed approach make this data structure as In-memory as well as auxiliary to handle the memory overflow.

6. Initialization of parse tree threads to each term to find the relevance of the terms to given phrase

Every fork join task further creates new threads for each term to generate parse trees. For parsing purpose OpenNLP [5] is used. This parser is used to find the relativity of the two words in the parse tree [9]. The depth of those words will give the similarity of those words.

7. Executing the query and generate the results with rank

Execution of the query is three fold. First, expand query with synonyms, antonyms, hypernyms, hyponyms and stem words. All these words are separated or merged with logical operators && and ||.Second, execution of query to get the results. Finally, sort the results to give ranks based on the relevance with the concept.

B. Pronunciation based similarity search

This is the second part of the proposed system. In this soundex like functions are used to find related words based on the pronunciation i.e., when listening when pronouncing these words gives same sound.

Final part of the proposed system is to merge all the results i.e., results for all fork join tasks and sound wise similar results and give the output to the requested user.

C. Context Selection based Search

In this model first user select the domain of search then submit the input. This will greatly reduce the search space. If the user is familiar with the domain what he is searching then the system only search the documents of dictionaries which belongs to that domain. But for this case reverse dictionary has to build for most of the frequently used

domains or languages prior to the search. But this is not a considerable problem due to onetime work.

IV. RESULTS

When working with other existing Reverse dictionaries [8] we come across the problem of getting irrelevant words as result for the given input query. We can overcome this with our approach to a great extent. The following results show the performance our Reverse Dictionary.

Input Phrase	Query Result
Present in large Quantity	Ample, bigger, larger, blown-up, enlarged, puffy, broad, spacious, wide, bulky, tremendous, elephantic, giant, jumbo, extended, huge.
Move fastly	Quick, swift, fast-paced, express, high-speed, immediate, prompt, hurrying, scurrying, rapid, prompt, smart, windy.

TABLE 2: Sample Results

V. CONCLUSION AND FUTURE WORK

The proposed framework is used to assemble the various parts of the efficient programming models and data structures to solve the problem of concept similarity using reverse dictionary. According to proposal effective memory management and search management is possible in large scale. Multi-phrase searching with concurrent execution greatly reduces the time complexity and effective memory management is achieved through in-memory and auxiliary data structure utilization.

Future work could be focused on applying of various data mining techniques to find the more accurate conceptual similarities.

VI. REFERENCES

[1] D. Lin, "An Information-Theoretic Definition of Similarity", Proc.Int'l Conf. Machine Learning, 1998.

[2] Dictionary.com, LLC, "Reverse Dictionary", http://dictionary.reference.com/reverse, 2009.

[3] G. Miller, C. Fellbaum, R. Tengi, P. Wakefield, and H. Langone, "Word net Lexical Database", http://wordnet.princeton.edu/Word net / download /, 2009.

[4] M. Porter, "The Porter Stemming Algorithm", http://tartarus.org/martin/porterstemmer/, 2009.

[5] O.S. Project "Open nlp", http://opennlp.sourceforge.net/ , 2009.

[6] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval ACM Press, 2011.

[7] J. Earley, "An Efficient Context-Free Parsing Algorithm", Comm. ACM, vol. 13, no. 2, pp. 94-102, 1970.

[8] OneLook.com, "Onelook.com Reverse Dictionary", http://www.onelook.com/ , 2009.

[9] U. of Pennsylvania, "The Penn Treebank Project", http://www.cis.upenn.edu/ Treebank /, 2009.

[10] 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