2013 International Conference on Advances in ICT for Emerging Regions (ICTer): 125 - 131

# Semi-Supervised Algorithm for Concept Ontology Based Word Set Expansion

N. H. N. D. de Silva[#1], A. S. Perera[#2], M. K. D. T. Maldeniya[*3]

[#]*Department of Computer Science and Engineering, University of Moratuwa*
*Moratuwa, Sri Lanka.*
[1]NisansaDdS@cse.mrt.ac.lk
[2]Shehan@cse.mrt.ac.lk
[*]Codegen International *(Pvt) Ltd*
*29,Braybrooke Street*
*Colombo 2, Sri Lanka*
[3]danajamkdt@gmail.com

**Abstract— Word lists that contain closely related sets of words is a critical requirement in machine understanding and processing of natural languages. Creating and maintaining such closely related word lists is a critical and complex process that requires human input and carried out manually in the absence of tools. We describe a supervised learning mechanism which employs a word ontology to expand word lists containing closely related sets of words. The approach described in this paper uses two novel supervised learning techniques that complement each other for the purpose of expanding existing lists of related words. Expanding concept variable lists of RelEx2Frame component of OpenCog Artificial General Intelligence Framework using WordNet is used as a proof of concept. Intervention of this project would enable OpenCog applications to attempt to understand words that they were not able to understand before, due to the limited size of existing lists of related words.**

***Keywords*— word lists, supervised learning, ontology**

## I. Introduction

Natural Language Processing (NLP) is a field of Artificial Intelligence (AI) on which a lot of research had been and also currently being carried out. It is a critically important hurdle in the development of a complete AI that simulates human behaviour.

Creating and maintaining word lists is an integral part of many NLP researches and applications thereof. Said wordlists usually contain words that can be deemed to have closely related meanings in the level of abstraction involved in the application. Therefore two words $W_1$ and $W_2$ which belong to a single word list in a certain application might belong to two wordlists in another application. This vagueness of definition and usage makes creation and maintenance of these wordlists a complex process.

RelEx, which is a component, developed for the OpenCog [1] framework is an English-language semantic dependency relationship extractor, built on the Carnegie-Mellon Link grammar parser [2]. Subject, object, indirect object and many other syntactic dependency relationships between words in a sentence can be identified by RelEx. In this system, words in English language are categorized into considerable number of concepts (e.g.: Pronoun, Storing, Time etc.). Words that belong to each concept are defined as concept variables.[3] There are manually categorized concept variables, present already. Said concept variable lists were expanded using the algorithm explained in this paper as a proof of concept. The results and performance section contain the outcome thereof.

The expected outcome of the project has a variety of applications in real life IT Solutions. Mainly AI related applications which require English language processing would benefit from the project. In addition the project deliverable can be used in chat applications, text critiquing, information retrieval from the web, question answering, summarization, gaming, and translation as it is intended for general use rather than focusing on specific areas of English language.

## II. Background

The exact requirement of the paper is to introduce a methodology to learn new words that fall into a certain concept based on the words that are already in the list relevant to the said concept using an ontology. This is a very specific implementation, thus implementations or algorithms that are closely catered to the likes of this situation are nonexistent.

### A. Ontology

An ontology is defined as "formal, explicit specification of a shared conceptualisation" [4] in information science. The term is borrowed from philosophy, where an Ontology is a systematic account of Existence. An ontology is consisted of a set of concepts from a selected domain and relationships between the said pairs of concepts. Conversely, the primary task of an ontology is to provide with the vocabulary to model the domain that it represents by providing the types of concepts that exist in the domain along with their properties and interrelations.[5] This set of objects (concepts), and the describable relationships among them, are reflected in the representational vocabulary with which a knowledge-based program represents knowledge.

Ontologies are used to organize information in many areas as a form of knowledge representation. These areas include; artificial intelligence, the Semantic Web, biomedical informatics, library science, enterprise bookmarking, and information architecture. In each of these use cases the ontology may either model the world or a part of it as seen by the said area's viewpoint.

At the ground level of an ontology are "Individuals" (instances). Depending on the nature of the Ontology in question, this may include concrete objects or abstract objects. Concrete objects may be people, animals, planets, etc. Abstract objects maybe numbers or words.

Individuals are grouped into structures called "classes". A class in an ontology can be referred as a concept, type, category,

978-1-4799-1276-6/13/$31.00 ©2013 IEEE

kind depending on the domain on which the ontology is based. Yet again depending on the convention used with a particular ontology, the definition of a class and the role thereof can be either analogous or distinct from that of a collection of individuals. The most important ability of classes in an ontology for this research is the ability to subsume or be subsumed by another class. A class that has been subsumed by another class is called a "subclass" while a class that has subsumed one or more classes is called a "superclass".

By this act of subsuming, a hierarchy of classes is formed. The critical importance of the subsumption relation is the phenomenon called "inheritance" whereby the properties of the subsuming class are inherited by the subsumed class. Traditionally the subsuming class is called the "parent" and the subsumed classes are called "children". Thus two or more classes that are sharing a parent would refer to each-other as "sibling". This naming convention is used throughout this paper.

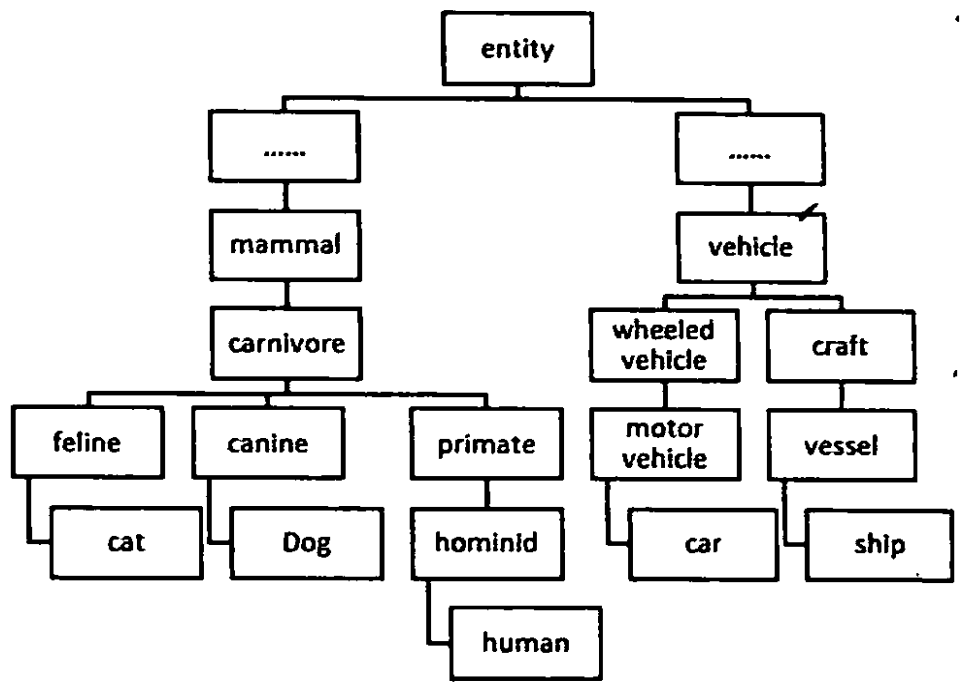


Fig. 1 Hyponym – Hypernym graph

### B. WordNet®

WordNet [6] is well known large lexical ontological database developed by the Cognitive Science Laboratory of the Princeton University, United States by Miller et al. [7]-[11]. It is a representation of the semantic relationships between words which are grouped together into sets of synonyms called synsets. The database contains more than 150,000 different words. In addition to this each word is coupled with a short description so that it can be used as a Dictionary as well as a thesaurus. The Database and the accompanying software tools are released under a BSD type license. For the purpose of this research, two of the semantic mappings of WordNet were examined.

1. Hyponym - hypernym mapping
2. Synonym mapping

A simplified version of the WordNet representation of three common semantic relations is shown in Fig. 3 WordNet Network representation of three semantic relations; hyponymy, antonymy and meronymy.

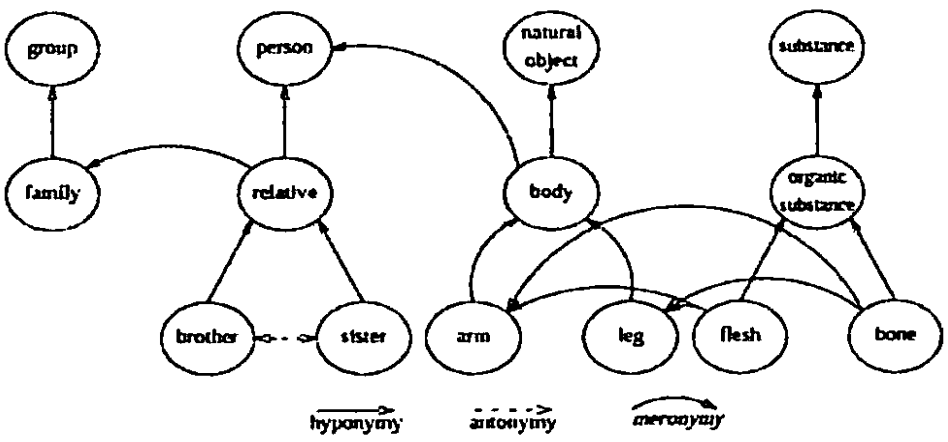


Fig. 3 WordNet Network representation of three semantic relations; hyponymy, antonymy and meronymy

*1) Hyponym - hypernym mapping:* A hyponym is a word that is a member of a lexical class. The title of the said lexical class is called a hypernym for the words that fall into the class. For an example words cat and dog are hyponyms of the lexical class called Mammal. Conversely, the word Mammal is a hypernym for the words cat and dog. An example for a Hyponym – hypernym graph is shown in Fig. 1.

This mapping basically utilizes the class hierarchy explained under the section Ontology. The definition of classes which are called hypernyms in a lexical ontology are derived from the concept of superclass in the definition of a generic ontology Similarly the definition of classes which are called hyponyms in a lexical ontology are derived from the concept of subclass in the definition of a generic ontology. Further, as described above the relationship is addressed as a parent-child relationship.

WordNet uses the hyponym – hypernym naming convention In this paper we have used the same convention for the ease of reference. But in cases where nodes that are higher in the hierarchy are needed to be referred or hyponyms of the same hypernym are referring to each other, when describing the algorithm, we have used child-parent-sibling naming convention for the sake of clarity.

*2) Synonym mapping:* As described above, synonyms are put together in sets called Synsets in WordNet. Given a word the

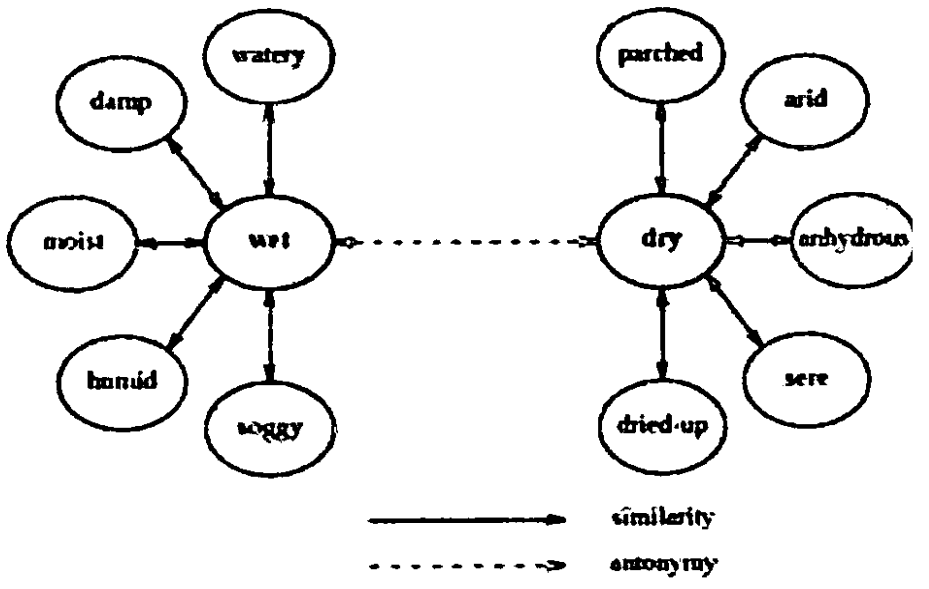


Fig. 2 Synonym- Antonym graph

synonym set and the antonym set can be easily accessed via the API. Fig. 2 shows an example synonym- antonym graph modeled in WordNet.

The concept of synonymy is handled by the property that dictates; individuals that show the property should be in the same class. These classes are called Synset as explained above. Thus in WordNet synonymy is a class level relationship. Antonyms, however are defined as relationships between words themselves instead of Synsets. Thus in WordNet antonymy is an individual level relationship.

*RelEx Natural Language Pipeline*

RelEx pipeline of the OpenCog Artificial General ntelligence open source project converts an English sentence o a set of semantic frames. The concepts lists discussed in this

```
$Imperative_relation
imperative
imperative-infinitive

$atLocation
_%atLocation
about
above
across
abutting_on
adjacent_to
adjoining_to
against
all_over
along
amid
amidst
among
around
astride
at
athwart
atop
```

Fig. 4 Part of the RelEx concept variable file

paper are used by this pipeline. The expansion of the said concept lists is used as a proof of concept for the algorithm proposed in this paper.

RelEx which is an English-language semantic dependency relationship extractor identifies the subject, object, indirect object and many other syntactic dependency relationships between words in a sentence.

The input context of the RelEx as shown in Fig. 4 has the concept name stated with the "$" symbol. The variables (i.e. words) that fall into the said concept are listed below the concept name. There are more than 4500 concepts grouped into 295 concepts.

## III. ONTOLOGICAL DATABASE BASED EXPANSION OF WORD LISTS

The algorithm is explained in this section in the context of the expansion of the variable base of RelEx in which the variable base is modelled as a collection of word lists, where each list covers an ontological concept and is comprised of a set of ontologically close set of words and a title which is a hyponym of the said set of words.

The proposed algorithm uses the hyponym-hypernym connections and the synonym connections of an ontological database which already has organized its word repository into a tree structure preserving the relations between the words. As described above, the words are already categorized into ontological groups in RelEx. Thus the aforementioned structure that the RelEx developers have used so far can be exploited, making it an ideal candidate for a demonstration.

### *A. Concept expanding using hyponym-hypernym structure*

The pseudocode shown in (Pseudocode 1) gives a basic outline of the proposed algorithm at a high level. Fig. 5 shows the flow diagram of the basic implementation which was used to expand the concept variable lists in RelEx as the proof of concept.

After each theoretical section explaining the main methodologies utilized in the algorithm, for the ease of understanding, we have included a worked out example of how the concept "$travel" of RelEx would be expanded by means of the said section of the algorithm through the lexical ontological database WordNet.

The *old Concept database* is the current concept variable store, which is a hand written list of words categorized into concepts. But due to the fully manual input method that the original developers used to develop this; the set is non

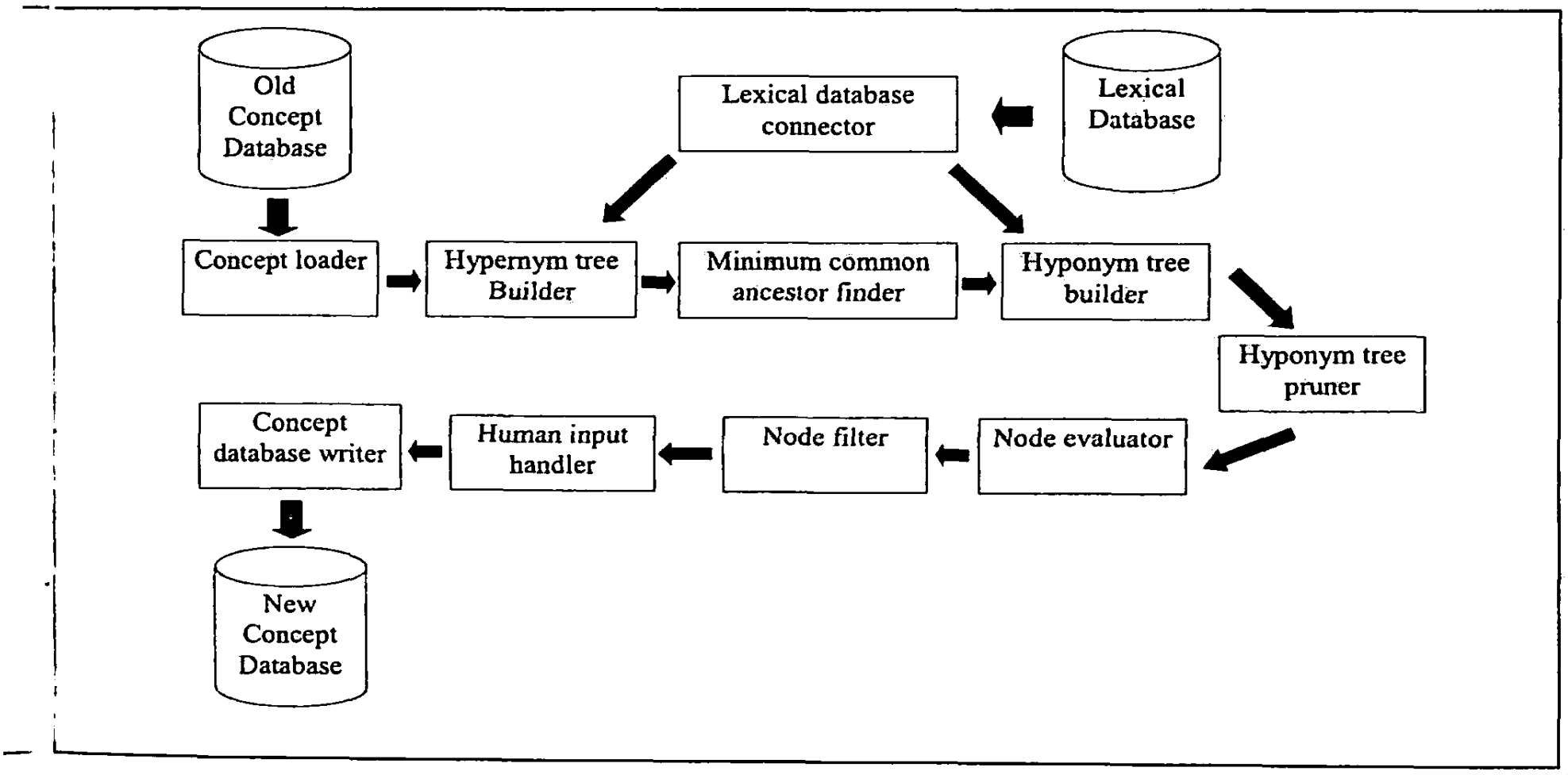


Fig. 5 Flow diagram for the Simplified architecture for concept expanding using WorldNet. (hyponym-hypernym)

exhaustive when compared with the words in the English language. The Concept loader load concepts from the current concept database and insert them into a data structure that can be easily handled by the subsequent modules.

```
EXPAND – USING – HYPONYM – HYPERNYM ○
    • LOAD CONCEPTS FROM OLD DATABASE •
    While (Concepts.exhausted == FALSE)
• LOAD CURRENT WORDVARIABLES FOR NEXT CONCEPT •
        • BUILD VARIABLE PAIRS •
      While (Variable Pairs.exhausted == FALSE)
          • BUILD HYPENYM TREES FOR THE WORD PAIRS •
          • FIND THE MINIMUM ANCESTORS FOR SENSES •
          • ADD THE MOST FREQUENT ANCESTOR TO HASH LIST •
    • BUILD HYPONYM FOR ANCESTORS IN HASHLIST •
    • PRUNE HYPONYM TREES BY CROSSCHECKING ORIGNAL LIST •
    foreach Ancestor A where A.contribution ≥ 50% Original word list do
        • BUILD NODELITS FROM HYPENYM TREES OF A •
    • AGGRIGATE NODELISTS •
    foreach Node N do
        • EVALUATE N USING ANCESTORS AND SIBLINGS •
    foreach Node N where N.words ⊂ {Top 80% Original word list} do
        • ADD N.words TO SELECTED WORDSET •
    • SUBSTRACT ORIGNAL WORDSET FROM THE SELECTED WORDSET •
    • PASS WORDSET FOR HUMAN APPROVAL •
```

Pseudocode 1 Basic outline of the proposed algorithm to expand Concepts using hyponym-hypernym structure

The *lexical database connector* (the *Wordnet connector* in this example) is used to query the lexical database (*Wordnet* in this example) and extract hyponym trees and hypernym trees rooted at the given word. For each word several hyponym/ hypernym trees are returned according to the senses of the given word. In the case of the RelEx concept variable "$travel", the words *commute, journey, tour, travel, voyage* are loaded as the current members.

*Hypernym tree builder* queries the lexical database through the lexical database connector for each word in a given list. The resultant trees are then converted into sense trees using the internal node structure. The words loaded using the RelEx concept variable "$travel", in the previous stage returns two common ancestors within WordNet; *journey* and *travel*.

*Minimum common ancestor finder* takes two words and their relevant sense trees from the hypernym tree builder at a time and simultaneously does a bottom up search for each pair of tress to find the common ancestor. Table 1 shows a sample ancestor finder matrix.

*Hyponym tree builder* queries the lexical database through the lexical database connector for each word in a given list. The resultant tree is then converted into sense trees using the internal node structure.

For each of the selected common ancestors, the collective distance to the two seed words is calculated. This calculated distance is observed to be directly proportional to the ontological distance between the two senses. Thus it was possible to conclude that the sense tress with the ancestor that has the minimum collective distance will be having a high probability of falling into the same ontological class.

After all the word pairs are analysed, the common ancestors for each word pairs are put into a list and sorted in the descending order of frequency. For each pair, only the ancestor with the highest frequency is put to the hash list that is passed *hyponym tree builder*.

*Hyponym tree pruner* calculates the cardinality of the set t is taken by applying set intersection to the original word set a the word set acquired from the tree rooted by each ancestor the list that was taken from the Hyponym tree builder. (Equati 1) The ancestor nodes are sorted in the descending order cardinality and only the top subset of ancestor nodes t collectively contribute to 50% or more are handed over to *node evaluator*. The two common ancestors found wit WordNet; *journey* and *travel* for the RelEx concept varial "$travel", get evaluated for values; 4 and 6 respectively.

$$\{original\ words\} \cap \{rooted\ tree\ words\} = \{selected\ word$$

Equation 1: Selection criterain for words

*Node evaluator* takes the shortlisted ancestor nodes a traverses the hypernym tree rooted by each ancestor node wh listing the nodes of which the intersection between the s t words in the node and the original word set is non empty. T cardinality of the said resulting set intersection is given as initial weight. From the two common ancestors found wit WordNet; *journey* and *travel* for the RelEx concept varia "$travel", only *travel* gets passed to this level because of 50% coverage rule.

TABLE 1

MINIMUM COMMON ANCESTOR FINDER MATRIX

| | | Word 1 | |
|---|---|---|---|
| | | *Sense 1* | *Sense 2* |
| Word 2 | *Sense 1* | ancestor 1 | ancestor 2 |
| | *Sense 2* | ancestor 2 | ancestor 1 |
| | *Sense 3* | ancestor 1 | ancestor 3 |

After building the selected node list, the said initial wei of each node is then increased by a constant factor multiplied the cardinality of the intersection of set of ancestors of the s node and the selected node list. The resultant weight of e node is again increased by a constant factor multiplied by cardinality of the intersection of set of siblings of the said n and the selected node list. (Equation 2) The evaluated node is passed to the *node filter*. From the WordNet ancestor *tra* 10 words are selected with the base significance 5.

$$weight = |\{original\ words\} \cap \{node\ words\}| + cons$$
$$\times |\{sibling\ nodes\} \cap \{selected\ node$$

Equation 2 Evaluation criterain for nodes

*Node filter* takes the evaluated node list and orders then the descending order of weights. The upper 80% of aggregated value of the entire batch are selected. The wo from the selected node are put into a set. The original word se subtracted from the said set and the resultant set is handed o to the *human input handler*. From WordNet ancestor *tra* only two words (*sail* and *navigate*) passes to the *human in handler*.

*Human input handler* takes the word set given by the N filter alongside the original set of words for that given conc in a graphical user interface. The human curator can then se the words that are desirable in the context of the given conc The human input handler takes the union of the word selected by the human curator and the original word set hands it over to the Concept database writer.

- *Concept database writer* takes the set of words from the human input handler and writes them into a new concept database following all the conventions that the original RelEx developers have used in developing the original concept variable database.

### 3. *Concept expanding using synonym structure*

The synonym based concept expansion process is a much simpler process than the Hyponym-Hypernym based concept expansion.

The pseudocode shown in (Pseudocode 2) gives a basic outline of the proposed algorithm at a high level abstraction. Fig. 6 shows the flow diagram of the basic implementation which was used to expand the concept variable lists in RelEx as

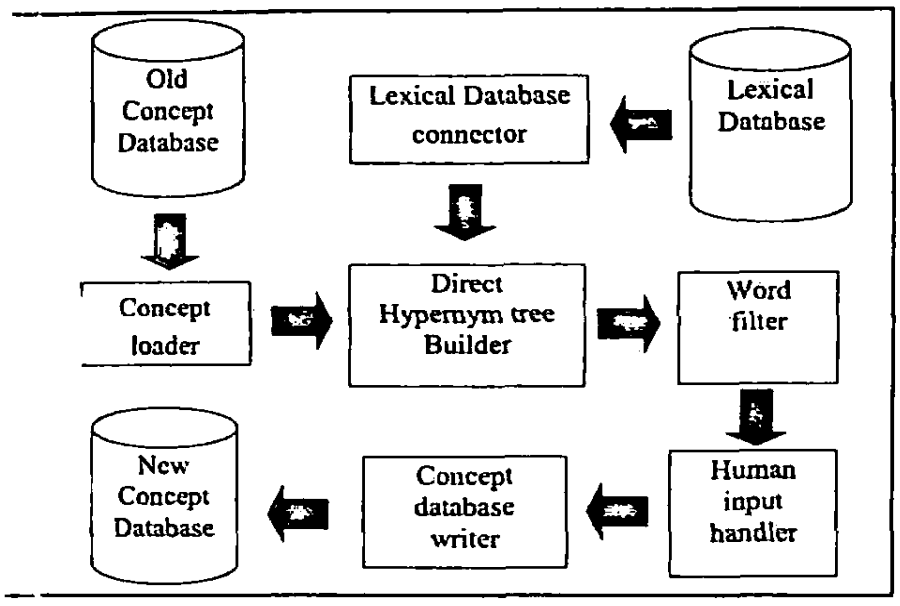


Fig. 6 Flow diagram for the Simplified architecture for concept expanding using WorldNet. (synonym)

the proof of concept.

The *old Concept database* is the current concept variable store, which is a hand written list of words categorized into concepts. An important point to note here is the fact that we conducted the experiment of expanding the RelEx concept variable store using hyponym-hypernym structure and expanding using synonym structure in parallel since this work is a proof of concept and we intended to understand the differences and similarities of the two approaches.

```
EXPAND – USING – SYNONYM()
    • LOAD CONCEPTS FROM OLD DATABASE •
    While (Concepts.exhausted == FALSE)
        • LOAD CURRENT WORDVARIABLES FOR NEXT CONCEPT •
        • BUILD DIRECT HYPONYM TREES FOR THE WORDS IN LIST •
        • BUILD DIRECT HYPONYM TREES FOR CONCEPT TITLE •
        foreach Hyponym trees do
            • EXTRACT WORDLIST FROM HYPONYM TREE •
            • SUBSTRACT ORIGNAL WORDSET FROM EXTRACTED WORDSET •
            • PASS WORDSET FOR HUMAN APPROVAL •
```

Pseudocode 2 Basic outline of the proposed algorithm to expand Concepts using synonym structure

The *lexical database connector* that we used here is the same *Wordnet connector* that we used for the hyponym-hypernym structure in the previous section.

Instead of the *Hyponym tree builder* used in Hyponym-Hypernym based concept expansion, here *the direct Hyponym tree builder* is used. The only difference between the Hyponym tree builder and the direct Hyponym tree builder is the fact that the latter imposes limit of 2 to the depth of the trees that are returned.

The *Word filter* in the synonym based concept expansion process is quite simple in comparison to the combined functionality of the *Node evaluator* and the *Node filter* in the Hyponym-Hypernym based concept expansion. The *Word filter* will take all the words occurring in the direct hyponym tree and put them into a set called the "candidate word set". Then the seed wordlist acquired from the old concept database for the purpose of building the direct hyponym trees in the previous step are put into the "original word set". Finally the set difference it taken using the said two word sets by subtracting the original word set from the candidate wordset. The resultant set is passed to the *Human input handler.*

The *Human input handler* and the *concept database writer* are as same as what was discussed under Hyponym-Hypernym based concept expansion.

## IV. Results and Performance

The input content for the proof of concept application were the current concept variable file of the RelEx Natural Language Pipeline and the Wordnet concept ontology. In development, both the concept variable file and the Wordnet concept ontology were used as direct inputs to come up with the new concept variables. The Wordnet file structure was discussed in the section II.B.

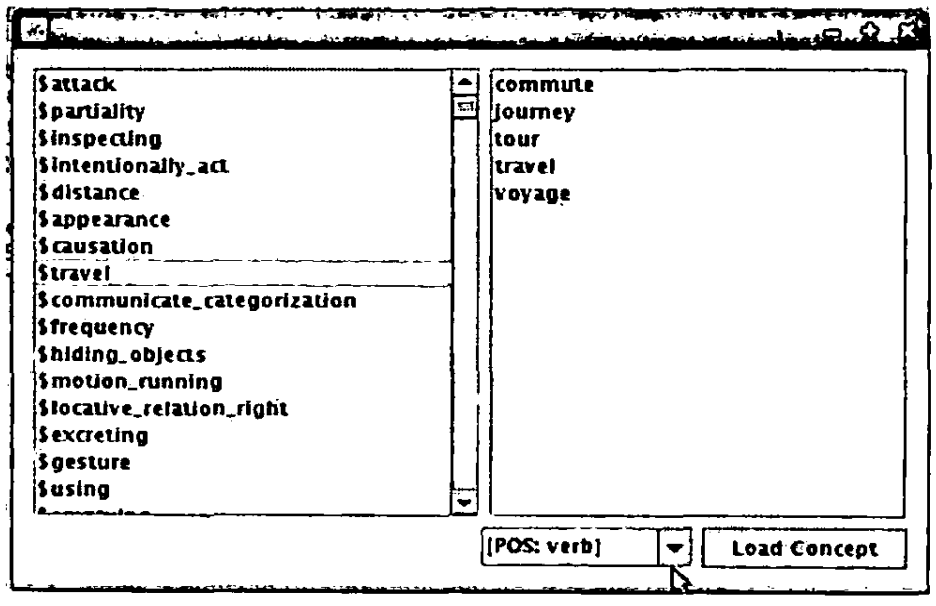


Fig. 7 Concept selection user interface

As shown in Fig. 7, a GUI was provided to the user to select the concept to be expanded through Hyponym-Hypernym structure. The concept list was shown on the left panel and the current variables for the selected concept was loaded on to the right panel. The concept $travel is used as an example in this paper to showcase the results of the algorithm.

Once a concept is selected, a new window was shown to the human user as shown in Fig 8 with the current concept variables in the selected concept on the left panel and the potential concept variables found through the Hyponym-Hypernym based expansion shown on the right panel. The synonym based expansion was given a similar interface to what was given for the Hyponym-Hypernym based expansion where again the current concept variables in the selected concept were shown on the left panel and the potential concept variables found through

the Synonym based expansion shown on the right panel. shows the result window for the concept $travel.

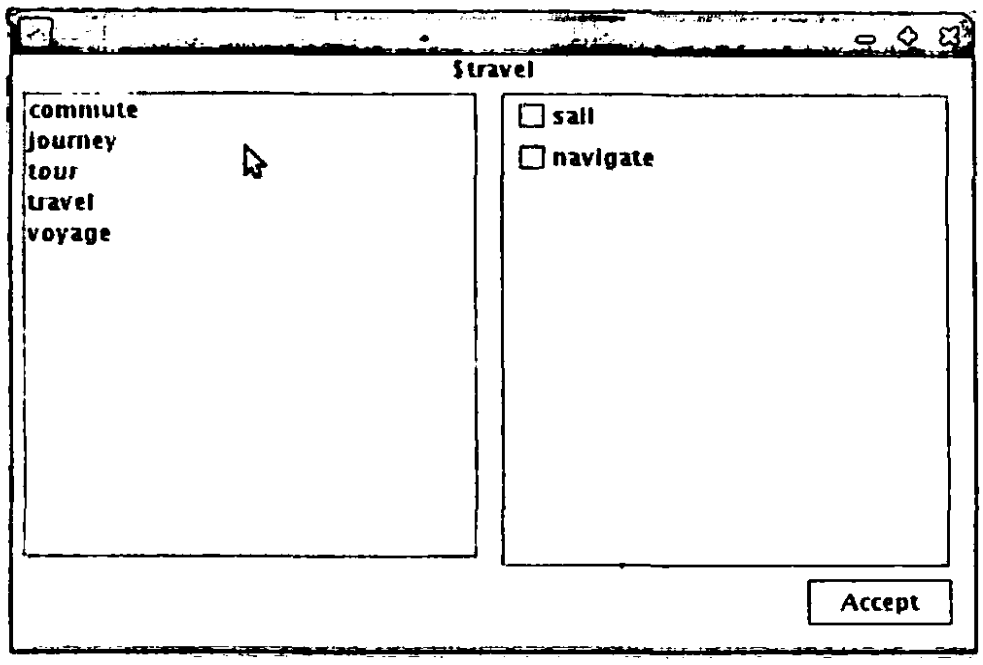


Fig. 8 Concept variable adding user interface for Hyponym-Hypernym based expansion

The concept variable expansion was based on the 295 concepts and the 4716 current members of the said concepts. After sending these concepts through the algorithm discussed, there were a total of 5089 concept variables under the said concepts. Thus it can be concluded that 373 new concept variables (words) has been added to the concept variable store in this iteration. For example the concept *$Travel* which only had the words; *commute, journey, tour, travel, voyage* was expanded by adding the words; *sail, navigate* by this process.

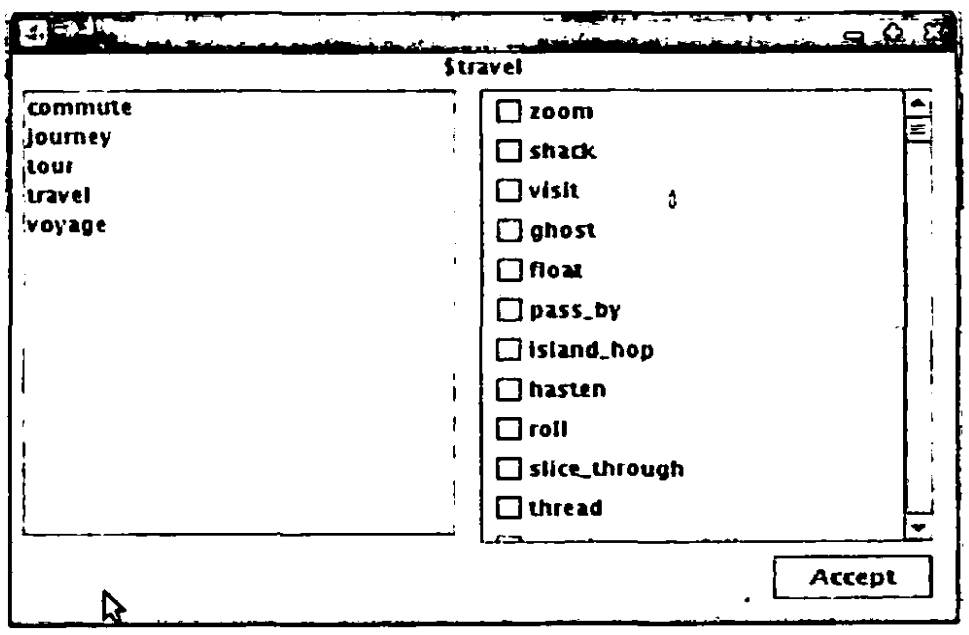


Fig. 9 Concept variable adding user interface for Synonym based expansion

As described above the experiment of expanding the RelEx concept variable store using hyponym-hypernym structure and expanding using synonym structure was conducted in parallel. But in a practical situation, it is quite desirable to use the concept database expanded by the hyponym-hypernym structure of the ontology as the source (old) concept database for the expansion based on the synonym based structure.

From the results of the human input trials, it was evident that the words that were returned by expanding the RelEx concept variable store using hyponym-hypernym structure showed a higher degree of coherence with the original word set that was used to create the hyponym-hypernym trees since the acceptance rate of word suggestions from the hyponym-hypernym based expansion was higher than the acceptance rate of word suggestions from the synonym based expansion. The words that were returned by expanding the RelEx concept variable sto using synonym structure showed a lesser degree of coheren with the original word set that was used to create the direc hyponym trees and was observed to be quite divergent.

The process of expanding the concept variable list of Rell contributed to enhance the natural language understanding of t OpenCog AGI Framework by adding more than 300 ne concept variables to the 295 concepts. This would empower t AI agents that use OpenCog to respond to new concepts th they were not equipped to respond before.

WordNet is not the only concept ontology API availab There are many concept ontology databases that are availat commercially or freely. Such a different concept ontolo database can be plugged into the concept expansion process the RelEx pipeline. As explained in this paper in the earl sections, only the WordNet connector will have to be replac with a suitable new connector to the new database.

## V. Future Work

The expansion algorithm is currently exploiting the m general relationships that are present in concept ontologi Hyponym-Hypernym structure is common to any ontolc while the Synonym structure is present in most word ontologi Depending on the lexical database that is to be used, other for of relationships that they represent between words and word s can be exploited to gain results even more accurate than w we have obtained in this work.

As mentioned in both the methodology and results sectic expanding the RelEx concept variable store using hypony hypernym structure and expanding using synonym structure conducted in parallel. It was also observed in the results sect that the results of the expansion using synonym structure showing a greater degree of divergence from the original se words used as input to the algorithm. An interesting expans on this would be using the semantic similarity [12] between set of original words and the resultant words and appl suitable threshold before exposing the result to the human in handler. However since the algorithm discussed in this paper not involve using statistical analysis by going through a cor the similarity measure will have to be calculated using a trained semantic relationship measuring component.

For the purpose of filtering words with a large drift from original concept, a trained data set from a sense disambigua tool such as "It Makes Sense" (IMS) [13] is also suitable. I has been trained using the popular statistical learning algori called LEXAS [14] and is apt in classifying words depending the sense.

Further, it is possible to integrate more than one lex database to the system at once and come up with a mechan to validate the results obtained through one lexical databas means of the results obtained through the other conne lexical databases thereby either greatly reducing the amour work forwarded to the humans through various interfaces.

In the experiment that we did as a proof of concept, we, researchers did the act of selecting the suitable words tc added to the concept variable store ourselves. In the case larger implementation, it is advisable to use a crowdsour [15] platform where the judgement of linguistic experts ca used instead.

## VI. Conclusion

In this paper we were able to explain a novel ontology based supervised learning approach to expand word lists. As a proof of concept we were able to design and execute an experiment on the concept variable store of the OpenCog AGI Framework, it was observed that the above discussed methodologies were successful in adding more than 300 new concept variables to the 295 concepts. It is understood that this expansion would empower the AI agents that use OpenCog to respond to new concepts that it was not equipped to respond before. With this success in the experiment, it is safe to conclude that the ontology based two fold methodology discussed in this paper is suitable for the expansion process of wordlists in the absence of similar tools for the same purpose.

## Acknowledgment

Firstly we would like to thank, the Department of Computer Science and Engineering of University of Moratuwa, for giving us this opportunity by providing the time and resources to conduct this research. We are grateful to Dr. Ben Goertzel of OpenCog foundation, who gave us his full support. We would like to thank Dr. Shantha Fernando for coordinating the course module that covered this research and the department staff for the support extended. We would like to extend our gratitude to all the developers of OpenCog framework especially Dr. Joel Pitt, Linas Vepstas, Jared Wigmore, for their enormous support. Our special thanks go to Mrs.Vishaka Nanayakkara, former head of the Department and Dr. Chandana Gamage, head of the Department for encouraging and guiding us throughout the research. Finally, an honourable mention goes to our families and friends who were always behind us and encouraged us to do our best.

## References

[1] "The Open Cognition Project – OpenCog Wiki," [Online]. Available: http://wiki.opencog.org/w/The_Open_Cognition_Project . [Accessed: 30- Jun -2013].

[2] "RelEx Dependency Relationship Extractor – OpenCog Wiki," [Online]. Available : http://wiki.opencog.org/w/RelEx. [Accessed: 30- Jun -2013].

[3] "RelEx Concept Variables," [Online]. Available : http://bit.ly/RelEx_Concept_Vajables. [Accessed: 01- Oct -2013].

[4] T. R. Gruber, "A translation approach to portable ontology specifications", *Knowledge Acquisition*, vol. 5, pp 199-220, 1993

[5] F. Arvidsson, A. Flycht-Eriksson, "Ontologies I" [Online]. Available: http://tomgruber.org/writing/ontolingua-kaj-1993.pdf. [Accessed: 15- Jun -2013].

[6] "Princeton University. WordNet – A Lexical Database for English" [Online]. Available: http://wordnet.princeton.edu/. [Accessed: 30- Jun -2013].

[7] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross and K. Miller, "Introduction to WordNet: An On-line Lexical Database", *International Journal of Lexicography*, vol.3, pp.235-244, 1990.

[8] G. A. Miller, "Nouns in WordNet: A Lexical Inheritance System", *International Journal of Lexicography*, vol.3, pp.245-264, 1990

[9] C. Fellbaum, D. Gross, and K. Miller, "Adjectives in WordNet", *International Journal of Lexicography*, vol.3, pp.265-277, 1990.

[10] C. Fellbaum, "English Verbs as a Semantic Net", *International Journal of Lexicography*, vol.3, pp.278-301, 1990.

[11] Beckwith, R., Miller, G. A. and Tengi, R., "Design and Implementation of the WordNet Lexical Database and Searching Software.", *International Journal of Lexicography*, vol.3, pp.278-301, 1990.

[12] J. J. Jiang and D. W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," Arxiv preprint cmp-lg/9709008, 1997.

[13] NUS Natural Language Processing Group. [Online]. Available :http://www.comp.nus.edu.sg/~nlp/software.html [Accessed: 30- Jun -2013].

[14] Hwee Tou Ng and Hian Beng Lee, "Integrating multiple knowledge sources to disambiguate word sense: an exemplar-based approach," in *ACL '96 Proceedings of the 34th annual meeting on Association for Computational Linguistics* , Stroudsburg, 1996, pp. 40-47.

[15] "Crowdsourcing," [Online]. Available: http://www.merriam-webster.com/dictionary/crowdsourcing. [Accessed: 30- Jun -2013].