Mapping Dependency Relationships into Semantic Frame Relationships

N. H. N. D. de Silva, C. S. N. J. Fernando, M. K. D. T. Maldeniya, D. N. C. Wijeratne, A. S. Perera
Department of Computer Science and Engineering,
University of Moratuwa,
Moratuwa, Sri Lanka.

B. Goertzel
OpenCog Foundation.
United States

Abstract— **We describe the refactoring process of the Natural Language Understanding pipeline of OpenCog Artificial General Intelligence Framework, a method for expanding the concept ontology of the pipeline using statistical learning algorithms. Further an experimental approach to automatically generating a common sense knowledge base specifically with relation to concept relationships derived from the natural language pipeline using data mining techniques is detailed and evaluated.**

Keywords-FrameNet; WordNet; Drools; Common Sense Knowledgebase;

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 behavior [1].

RelEx, which is a component, developed for the OpenCog [2] framework is an English-language semantic dependency relationship extractor, built on the Carnegie-Mellon Link Grammar parser [3]. Subject, object, indirect object and many other syntactic dependency relationships between words in a sentence can be identified by RelEx.

A mapping of English sentences into semantic frame relationships, similar to those of FrameNet, is provided by RelEx via the component known as RelEx2Frame [4]. Compared to the default RelEx output, this provides a higher-level, more abstract, but semantically more tractable description of the parsed sentence. The goal of such framing is to assist cognitive reasoning; rather than requiring a large common-sense database, this approach enables a reasoning or question answering system to deduce information which can be directly inferred from natural language sentences based on the linguistic structure and a relatively small set of framing rules. Existing framing rules are specified as simple IF..THEN rules, and are evaluated using a simple forward-chaining reasoner. The RelEx2Frame forward-chaining reasoner code was written by RelEx developers themselves without using any existing forward-reasoning code.

In the field of artificial intelligence a common sense knowledge base is a collection of facts that an average individual is expected to know that is structured in such a way as to allow artificial intelligence agents to use natural language or make inferences about the surrounding world [5] [6].

Common Sense Knowledge Bases have been an area of interest in the field of artificial intelligence for more than two decades and even now there are number of ongoing Common Sense Knowledge Base creation endeavors. Most of these endeavors involve manually inserting the common sense relationships and statistics and represent huge amount of human effort over a large period of time. Thus these knowledge bases suffer from issues such as difficulty of updating and ensuring the validity of the statistics.

SeMap looks into three main aspects of OpenCog in order to enhance its natural language understanding capabilities. Firstly, SeMap introduces a widely known open source rule engine, Drools to standardize the current framing rules and refactor the Relex2Frame architecture to accommodate the changes in the rule base. The modifications to the Relex2Frame architecture were carried out in such a way so as to mitigate the drawbacks of using a standard rules engine.

Words in English language can be categorized into considerable number of concepts (e.g.: Pronoun, Storing, Time etc.). Words belong to each concept are called concept variables. There are manually categorized concept variables, present already. In the second phase we expanded that repository by adding new concept variables which were found by a WordNet [7] based supervised learning mechanism and a statistical learning based approach using LEXAS algorithm [8].

Finally we describe a novel approach on building a common sense knowledgebase, where we automate the generation of common sense knowledge by using statistical techniques on the results of RelEx natural language understanding pipeline for a text corpus. The automation of the process leads to its own set of unique problems including the difficulties in ensuring the semantic value of the inference rules generated.

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. Additionally 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

A. RelEx Natural Language Pipeline

RelEx pipeline of the OpenCog Artificial General Intelligence open source project converts an English sentence to a set of semantic frames. This pipeline is used for standardization of the rule base as well as for the automated generation of common sense knowledge approach considered 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.

RelEx2Frame is used to map RelEx relations into a set of semantic frames using hand coded large set of rules. Frames are taken from two semantic resources, namely FrameNet and Novamente. Input to RelEx2Frame is the output of RelEx on a given sentence. Thus accuracy of the system heavily depends on the accuracy of RelEx.

Ex: "Put the ball on the table".

RelEx Relations	Semantic Frames
imperative(Put) [2]	*^1_Placing:Agent(put, you)*
_obj(Put, ball) [2]	*^1_Placing:Theme(put, ball)*
on(Put, table) [2]	*^1_Placing:Goal(put, table)*
singular(ball) [2]	*^1_Locative_relation:Figure(ball)*
singular(table) [2]	*^1_Locative_relation:Ground(table)*

Lack of readability, difficulties in maintenance and debugging are few issues of the existing RelEx2Frame.

B. Concept Word List Expansion

The exact requirement of the paper is to introduce a methodology to learn new words that fall in to a certain concept based on the words that are already in the list relevant to the said concept. This is a very specific implementation, thus implementations or algorithms that closely cater to the likes of this situation are nonexistent. Therefore, statistical word learning algorithms used for observation based learning and sense identification were examined.

1) LEXAS algorithm

LEXAS algorithm (Ng and Lee 1996) [8] is an algorithm to disambiguate word senses by applying statistical learning over multiple knowledge sources.

In the training phase, a set of sentences; S with words that are sense tagged is given. The algorithm extracts information about w; Parts of Speech (PoS) relevant to the words that occur near w, morphological form of w in the sentence and words frequently co-occur with w. If the word w is a noun, the verbs that take w as the object are also taken into account. In the testing phase all the above information is extracted from the given sentence and the result is compared with all the training examples and the sense which has the closest match is presented as the solution.

2) Word Independent Context Pair Classification Model

Word Independent Context Pair Classification Model (Niu et al. 2005) [9] is a word sense disambiguation model which uses maximum entropy modeling to train the word independent context pair classification model via an annotated corpus. Then that classification results are used to cluster the word mentions in the raw corpus. This algorithm is guaranteed to deliver the efficiency of the supervised Naïve Bayes system. The Maximum Entropy Modeling for Context Pair Classification step of this algorithm proved useful for the requirement discussed in this paper.

3) Statistical Word Learning Based on Cross-Situational Observation Algorithm

The cross-situational observation based statistical word learning algorithm (Yu et al. 2006) [10] is a statistical learning algorithm that learns word object pairs by analyzing co-occurrences. It is designed to handle word object pairs in ambiguous environments where multiple word candidates exist for any possible object and multiple object candidates exist for any possible word.

C. Common Sense Knowledgebases

Existing well known knowledge bases include OpenCyc by CyCorp[11] and NELL(Never Ending Language Learning) by Carnegie-Mellon University[12].

1) OpenCyc

This is the largest and the most well-known common sense knowledge base currently and it's the result of almost three decades of accumulation. The primary criticism of this project is the fact that the addition of assertions is carried out manually and the large size of the knowledge makes it difficult to update as well as test for the level of quality [11].

The knowledge base structure is based on concepts known as constants which include individuals, collections, truth functions etc.

The key predicates used in describing items in Cyc are inheritance (#$isa) and generalization (#$genls)

Ex:

"Kumar Sangakkara is a cricketer" would be represented as

#$isa #$KumarSangakkara #$Cricketer

The knowledge base also contains statements which accommodate variables which are called rules. Further Cyc is divided in to a number of domain based collections called micro theories which are constrained to have no contradicting statements. The native inference engine of Cyc supports general logical deductive operations such as modus ponens, modus tollens etc.

2) NELL

This is a much more recent project initiated by Carnegie-Mellon University that crawls the web and continuously extracts the semantic information from unstructured web pages. The project is based on a seed data set of categories and relations and expands this data set using the information extracted by crawling the web. While the approach for the development of NELL is quite similar to the one employed in this project, the common sense knowledge base developed in the project generates inference rules as opposed to category based relations accumulated in NELL[12].

While relatively new NELL has already accumulated around 850,000 beliefs such as

"American_forestry_association is a professional organization" and "information created contact"

The accuracy of the knowledge base is maintained through manual supervision and input from voluntary monitors.

III. STANDADIZATION OF RELATION-FRAMENET RULE BASE

The existing Relex2Frame architecture consists of over 5000 hard coded rules on .txt format. The project required to port these rules to a standard rule engine to accommodate both backward and forward chaining which would be useful for future developments in Frame2Relex.

In selecting an appropriate rule engine, factors such as platform independency, native Java and backward chaining support were considered. Thus Drools[13] was chosen as the most suitable rule engine for our purposes considering other alternatives.

After analyzing several Relex2Frame prototype architectures the performance results provided conclusive evidence that the incorporation of a standard rule engine using the Rete's algorithm with over 5000 rules in the rule base results in significant degradation of performance. In order to achieve performance comparable to the framework with the native rule engine, it was decided to incorporate concurrency in to its operation as well as to use techniques such as indexing, buffering and batch processing. In that aspect, the following (Fig. 1) Asynchronous Concurrent architecture was designed to mitigate the drawbacks of using Drools in Relex2Frame.

The proposed Asynchronous Concurrent architecture based on events is designed for batch processing of sentences in executing upon a text corpus to improve the overall throughput. The Drool rule knowledge base consisting of 5,341 rules were divided in to 54 small knowledge bases with each 100 rules and the knowledge bases are kept serialized for quick loading. A knowledge base buffer is maintained of fixed number of small rule bases at a time to service requests from processed sentences. Centralized control over sentence processing and knowledge base scheduling is kept to optimize the overall functionality of the system while the responsibilities of processing of each sentence are delegated to its own sentence object.

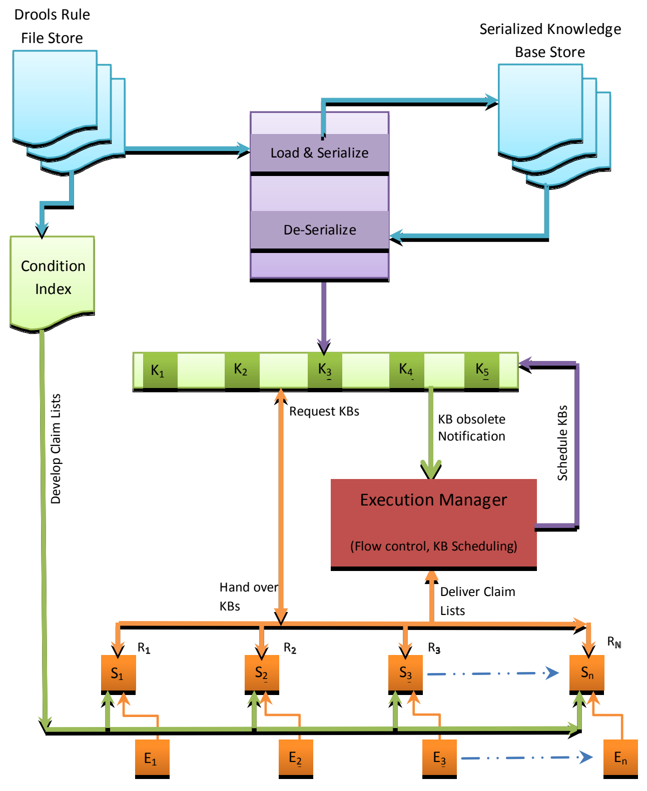

Figure 1. Relex2Frame Asynchronous Concurrent Architecture

A. Component Overview

The proposed architecture is composed of Sentence, Evaluator, Knowledge Base Manager, Knowledge Base Buffer, Execution Manager and Condition index components.

1) Sentence

A sentence represents an execution unit of the Relex2Frame framework, in that it represents a single input sentence to the RelEx Framework and retains the RelEx output related to the sentence as a collection and is fed in to the RelEx2Frame framework in blocks. Sentence plays a key role in the proposed architecture which is designed as an autonomous unit that "processes" it with necessary services requested from management objects. It is responsible for generating a knowledge base claim list for execution against it using the Condition Index, requesting and acquiring knowledge bases from the Execution Manager as well as retaining the list of semantic nodes that fit the represented sentence.

2) Evaluator

The evaluator is responsible for comparing the RelEx relations for a sentence with the relations or relation families required to be present in the rules for satisfaction. The evaluator categorizes the space of rules in RelEx2Frame in to four primary categories and evaluates the presence of unique relationships using an index of concept variables and a working memory that refreshes per rule to hold temporary variables.

3) *Knowledge Base Manager*

The Knowledge Base Manager is a "wrapper" for the standard Knowledge Base object of the Drools Rule Engine designed to serialize the object on creation (if the serialized version doesn't exist) in order to minimize the time taken to load the object. Currently each Knowledge Base Manager represents a knowledge base with a hundred mapping rules though this can be changed accordingly.

During testing, it was discovered that a given knowledge base could be held by multiple sentences for processing without generating operational issues since the knowledge base itself doesn't experience any modification during processing. This approach improves performance significantly in comparison to when a knowledge base is held exclusively by a single sentence at a given time.

4) *Knowledge Base Buffer*

This is a container for a set of Knowledge Bases (Knowledge Base Manager Objects) where the knowledge base loading and removal is scheduled based on Execution Manager calls. The container size is a variable that may be changed to achieve optimal performance based on available hardware.

5) *Execution Manager*

The Execution Manager represents the (limited) centralized control mechanism of the architecture. It is responsible for scheduling the loading of knowledge bases in to the working memory of the framework by using the claim list submitted by executing Sentences prior to execution.

The Execution Manager employs the knowledge base claim lists of individual sentences in execution at a given time to generate a priority queue for scheduling the knowledge bases in to the buffer by considering the number of requests for a knowledge base as the basis for assigning priority.

6) *Condition Index*

The Condition index is an index of all relationships or relationship families in the space of mapping rules where the values pointed to by a particular key (relationship) are pointers to the knowledge bases that included the rules which contained that key.

In addition to further improve the performance of indexing, the rules are organized in the knowledge base files such that the most frequently fired rules being clustered together to limit the number of knowledge bases necessary for processing of a sentence. The architecture was tested with the following rule clustering algorithms which yielded different results.

i. *Higher priorities to the frequent relation headers*
ii. *Association of relations*
iii. *Relation frequency in rules*
iv. *Hybrid of 2 and 3 algorithms*
v. *Minimizes the distribution of relation headers*
vi. *Self Organizing Map*
vii. *k-means clustering algorithm*

B. *System Functionality*

The run time operation of the RelEx2Frame framework involves multiple Sentences processing themselves by requesting the relevant Knowledge Bases. The sequence of operations involved in the processing of a batch of sentences is as follows.

Once a Sentence block is transferred to RelEx2Frame, each sentence will generate its Knowledge Base Request list by querying the condition index based on its relationship collection. These request lists will be used by the Execution Manager in initializing the Knowledge Base buffer with a set of knowledge bases that will be most requested by the Sentences in the block being processed at the given time reducing the overall volume of I/O activity.

The sentences process independent from each other while required knowledge bases are acquired from the knowledge base buffer. The knowledge base manager would handle the knowledge bases among the sentences and remove obsolete and bring in new knowledge bases to the buffer.

IV. CONCEPT WORD LIST EXPANSION

The variable base of RelEx is 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.

A. *Ontological database based expansion of word lists*

The first of the two methodologies that were used in this project for expanding the current variable base was intended to exploit the aforementioned structure that the RelEx developers used so far. Since the words are already categorized in to ontological groups, an ontology based expansion process was suggested. It was understood that an ontological database which already has organized its word repository into a tree structure preserving the relations between the words is ideal to achieve this end. From the said tree structure, the hyponym-hypernym connections and the synonym connections were used.

1) *Concept expanding using hyponym-hypernym structure*

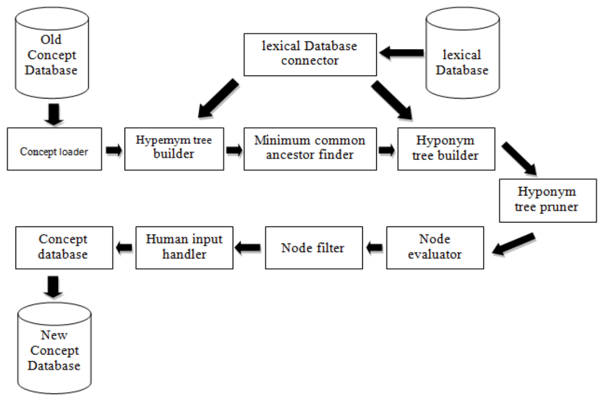

Figure 2. Flow diagram for the simplified architecture for concept expanding using hyponym-hypernym structure

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

The lexical database connector is used to query the lexical database 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.

Hypernym tree builder and 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 in to sense trees using the internal node structure.

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 I shows the ancestor finder matrix.

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 in to the same ontological class.

After all the word pairs are analyzed, the common ancestors for each word pairs are put in to 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 to hyponym tree builder.

Hyponym tree pruner calculates the cardinality of the set that is taken by applying set intersection to the original word set and the word set acquired from the tree rooted by each ancestor in the list that was taken from the Hyponym tree builder. The ancestor nodes are sorted in the descending order of cardinality and only the top subset of ancestor nodes that collectively contribute to 50% or more are handed over to the node filter.

$$\{original\ words\} \cap \{rooted\ tree\ words\} = \{selected\ words\} \tag{1}$$

TABLE I. 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

Node evaluator takes the shortlisted ancestor nodes and traverses the hypernym tree rooted by each ancestor node while listing the nodes of which the intersection between the set of words in the node and the original word set is non empty. The cardinality of the said resulting set intersection is given as the initial weight.

After building the selected node list, the said initial weight of each node is then increased by a constant factor multiplied by the cardinality of the intersection of set of ancestors of the said node and the selected node list. The resultant weight of each node is again increased by a constant factor multiplied by the cardinality of the intersection of set of siblings of the said node and the selected node list. The evaluated node list is passed to the node filter.

$$\begin{aligned} weight = {} & |\{original\ words\} \cap \{node\ words\}| + constant1 \\ & \times |\{ancestor\ nodes\} \\ & \cap \{selected\ nodes\}| + constant2 \\ & \times |\{sibling\ nodes\} \cap \{selected\ nodes\}| \end{aligned} \tag{2}$$

Node filter takes the evaluated node list and orders them in the descending order of weights. The upper 80% of the aggregated value of the entire batch are selected. The words from the selected node are put in to a set. The original word set is subtracted from the said set and the resultant set is handed over to the human input handler.

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

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

2) Concept expanding using synonym structure

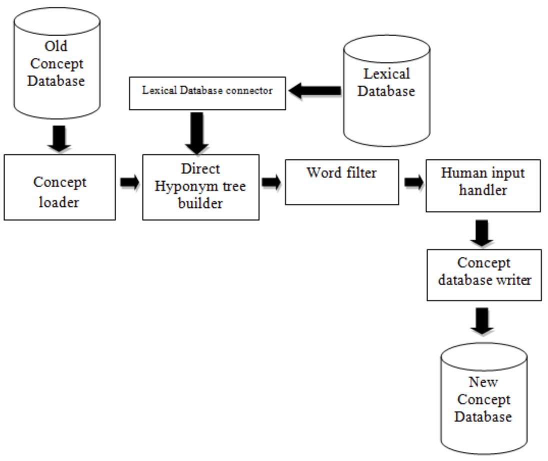

Figure 3. Flow diagram for the simplified architecture for concept expanding using synonym structure

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.

B. Statistical learning based on corpus for expansion of word lists

The second methodology used in this project for expanding the current variable base was intended to exploit the inherent features of the English language. These include the fact that the words that belong to the same concept tends to fall in the same position of a sentence. The following set of sentences can be used as an example.

- The cat sat on the fence
- The cat ran under the fence
- The cat stood near the fence.

It can be observed that the words occurring around the set of words; *{on, under, near}* tends to show statistical pattern. The said words fall in to the same concept in the old version of the concept store. Namely; *at_location*. What this module of the program was expected to achieve is, if given a sentence; *"The cat jumped over the fence",* in an instance where the word over is not on the current concept list for *at_location*; predict the probability of the word over being belonged to the concept *at_location*.

It was found that this can be achieved from using a statistical learning algorithm. It was discovered that no direct implementation of statistic based concept classification similar to the requirement of this project has been done. Thus the closest implementation; statistical word learning algorithms that are used for observation based learning and sense identification were taken in to consideration. As described in the related work section, three widely used such algorithms; LEXAS algorithm [8], Word Independent Context Pair Classification Model [9] and Statistical Word Learning Based on Cross-Situational Observation Algorithm [10] were analyzed with the requirements of the project.

1) Observations and Design decisions in building a suitable algorithm

It was observed that the Vector Space Model (VSM) based similarity measure of the Word Independent Context Pair Classification Model to have the same effect as the Unordered Set of Surrounding Words based similarity measure of the LEXAS algorithm. Further, the Latent Semantic Analysis measure of the Word Independent Context Pair Classification Model was observed to be the same as the Local Collocations measure of the LEXAS algorithm. Latent Semantic Analysis (LSA) based relationship similarity measure of the Word Independent Context Pair Classification Model was observed to be a unique feature.

It was possible to determine that the methodology used in Statistical Word Learning Based on Cross-Situational Observation Algorithm could be absorbed in to the Unordered Set of Surrounding Words based similarity measure of the LEXAS algorithm.

From the above set of observations and micro level decisions; if was ultimately decided to use an adaptation of the LEXAS algorithm for this project. The unique feature; Latent Semantic Analysis (LSA) based relationship similarity measure of the Word Independent Context Pair Classification Model was decided to be incorporated in to the Unordered Set of Surrounding Words based similarity measure of the LEXAS algorithm.

2) Statistic collection from corpus

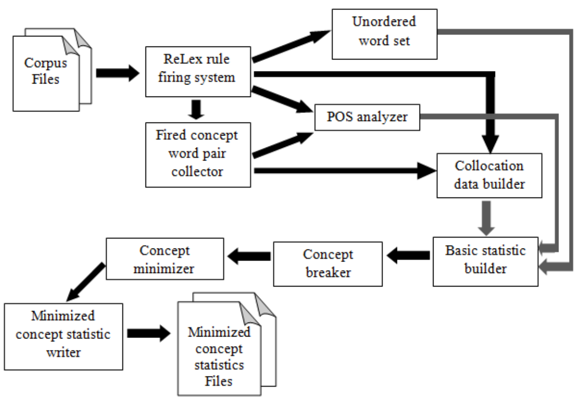

Figure 4. Simplified architecture for statistic collection from corpus

Fired concept word pair collector was integrated in to RelEx so that when a rule gets fired with relevant to a given sentence and returns true, the concept word pairs that were cross validated in the process are put in to a list, indexed by the concept so that ultimately it would return a list of successfully fired concepts and a list of words each of those concepts that made the successful firing instances.

Unordered word set collector takes all the words in the given sentence from RelEx, where initial list members are the added in to a new list where the each word is accompanied by the number of occurrences of that word in the said sentence. This list gets handed over to the basic statistic builder.

POS analyzer takes the concept indexed list from the Fired concept word pair collector and the RelEx relation list for the analyzed sentence from the RelEx rule firing system. The Part Of Speech data for each word considered was extracted from the RelEx relations and was handed over to the basic statistic builder.

Collocation data builder takes the concept indexed list from the Fired concept word pair collector and the words in the given sentence from RelEx. It then builds collocation data for each of the words in the concept indexed list. This collocation data gets handed over to the basic statistic builder.

Basic statistic builder takes the collected statistic data from the unordered word set collector, POS analyzer and the collocation data builder and aggregates this new statistical data to the current statistical data that is maintained for a given concept. In the case where there was no previous record of the given concept, a new statistic slot gets opened for it and the new statistical data is inserted. Concept breaker takes the concept statistics and breaks them in to concept headers.

TABLE II. COLLOCATION EXAMPLES

Left offset	Right offset	Collocation Example
-3	-1	only enables him [word]
-2	-1	from the [word]
-2	1	waist down [word] uses
-1	-1	Superhuman [word]
-1	1	down [word] uses
-1	2	powered [word] that not
1	1	[word] invent
1	2	[word] the waist
1	3	[word] walk but gives

Concept minimizer takes a raw concept statistic and prunes it according to a given set of rules.

- Words in the word frequency list, POS data and Collocation data are filtered out by a minimum frequency threshold.
- Common words such as a, an, the, to and of are removed from the word frequency list.
- Words accompanied by punctuation marks are ripped of the punctuation marks
- Words that are actually numbers, are completely dropped from the statistic

The minimized concept statistic is handed over to the minimized concept statistic writer. Minimized concept statistic writer writes the minimized concept statistics in to minimize concept statistics files.

3) Statistic based prediction from corpus

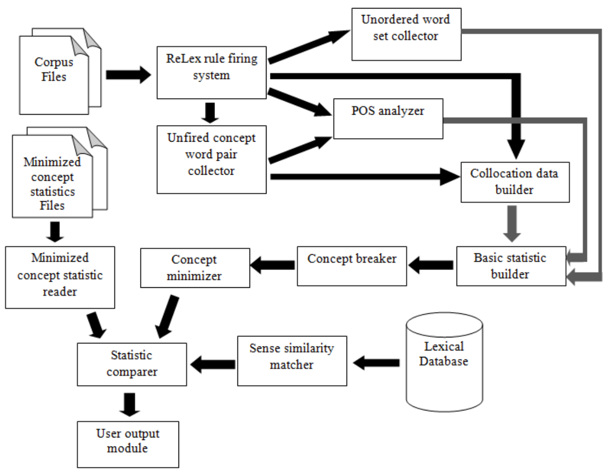

Figure 5. Simplified architecture for statistical prediction from corpus

Unfired concept word pair collector was integrated in to RelEx so that when a rule tries to get fired with relevant to a given sentence but returns false, the concept word pairs that were cross validated in the process are put in to a list, indexed by the concept so that ultimately it would return a list of tested concepts and a list of words each of those concepts that were checked against the said concept. This concept indexed list was handed over to the POS analyzer and the collocation data builder.

Sense similarity matcher uses a lexical database to come up with a value to represent how close two words are according to the semantics of the word. When the similarity increases the value approaches 1 and when similarity decreases the value approaches 0. The same word given to compare with itself will be given the value 1.

Statistic comparer takes the minimized statistic for the sentence from concept minimizer and loads the saved statistic for the relevant concepts via the minimized concept statistic reader. The POS and collocation data are compared directly and in cases where a match is found, the occurrence frequency at the minimized statistic for the sentence is multiplied by the occurrence frequency at the loaded minimized concept statistic and a constant. The said value is then added to the aggregated similarity measure. Each word in the unordered word set of the minimized statistic for the sentence is compared with each word in the unordered word set of loaded minimized concept statistic using the sense similarity matcher. These values are added to the aggregated similarity measure without applying a specific weight. (I.e. the weight is 1) The aggregated similarity measure for concepts against each word is used to calculate the percentage probability of the word belonging to that concept. The concepts are then sorted in the descending order of probability. The top five (if the concept count is less than five that number) possible concepts are handed over to the user output module which displays the word concept belonging probabilities for a human to consider.

V. COMMON SENSE KNOWLEDGE BASE GENERATOR

The discussed approach automatically generates a common sense knowledge base in the form of a set of inference rules based on the semantic frames extracted using the RelEx pipeline using a version of English Wikipedia as a corpus.

The derived rules take the basic form indicated by the given examples which are representation of the common sense knowledge regarding intelligence and comprehension.

^1_Mental_property (stupid)& ^1_Mental_property: Protagonist ($var0) →^1_Grasp: Cognizer (understand, $var0) <0.3>

^1_Mental_property (smart) & ^1_Mental_property: Protagonist ($var0) →^1_Grasp: Cognizer (understand, $var0) <0.8>

Each rule indicates the probability with which the semantic nodes in the consequent are valid for a given text input given those in the premise are known to be valid.

A. *Association Mining Approach*

The closest standard approach in data mining that considers problems similar to generating the common sense rules of the form shown in the given examples from a mass of semantic frames acquired from processing a text corpus is Association Mining.

The proposed approached that has been developed for the Common Sense Knowledge Base generation is derived from this field of data mining. The unique nature of the problem places a number of constraints on the association mining algorithm that can be employed for the purpose of generating the inference rules. The following considerations were particularly significant in the selection of an appropriate algorithm.

i. *The low frequency of occurrence of semantic frames requiring a significantly low support threshold.*

ii. *The need to capture rules from a wide range of probabilities requiring a low confidence threshold.*

iii. *The skewed nature of the semantic frame occurrence distribution.*

The FP-Growth [14] family of algorithm was selected based on these considerations and forms the basis core of the rule generation mechanism.

1) *Feature Vector Design*

Since the inference rules contained in the Common Sense Knowledge Base are essentially association relationships among semantic frames, that the feature vector should contain semantic frames is an obvious conclusion.

The first iteration of the mining application for the generation of inference rules directly used the existing set of semantic frames output from RelEx pipeline as the feature vector. This amounted to the vector consisting of 4359 features.

However using the original frames as features led to inference rules that were of a generic nature due to the coarse grain of the concept division in the concept variable store used by the pipeline. For example the store contained the concept $mental_property which included values (words) for concepts such as being "intelligent" or "stupid". Thus the derived rules would only encompass relations common to both being "intelligent" and "stupid" and rules that consider the consequences of being only "intelligent" or being only "stupid" would not be generated.

The second iteration used a set of frames based on a concept variable store where the concept division was fine grained to the extent that feature vector contained multiple features that were only differentiated by the newly introduced concept division. For example instead of having a given frame (feature) which contained the feature $mental_property there would now be two features which differed due to having the sub concepts $mental_property-intelligent and $mental_property-stupid. The resulting feature vector contained 25,782 features/frames. The approach for generating new concept division will be discussed in some detail under the heading 'Sub Concept Clustering'.

While this approach resulted in rules significantly closer to the expectation the decrease in frequency of a given frame due to the original frames splitting in a set of new frames means that the size of the corpus required for generating reasonable results is significantly greater than that was originally apparent.

2) *Core Common Sense Generation Algorithm*

Three key modification/additions have been made to the standard FP-Growth algorithm in designing the approach for generating the Common Sense Knowledge Base.

Use of All_Confidence(α) objective measure

The frequency of occurrence of semantic frames has been observed to be consistent with a skewed distribution with a low support threshold.

In order to avoid the loss of valid rules due the ineffectiveness of support and standard confidence as objective measures for pruning rules in such circumstances, All_Confidence which is a more appropriate measure is used [15].

$$\alpha \text{ of an itemset} = \frac{\text{support of itemset}}{\text{frequency of the most frequent item in the itemset}} \quad (3)$$

Fast Updated Algorithm for Incremental Updating of the Common Sense Knowledge Base.

The generation and maintenance of a common sense knowledge base is a long term process and involves processing huge quantities of text in this case. Therefore a mechanism that allows incremental updating of the common sense whenever new data is provided is critical to ensure maintainability of an automatically generated common sense knowledge base.

For this purpose the Fast Updated Algorithm (Fig. 6) which partitions item-sets in the context of frequency and presence in processed and new transactions and treats each case uniquely has been incorporated in to the core algorithm.

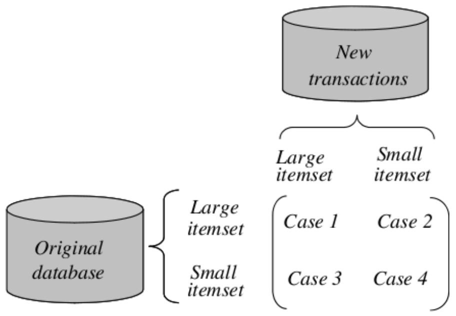

Figure 6. Fast Updated Algoritm

The algorithm proposed by Tzung-Pei Hong et al is used as the basis for designing the algorithm for supporting the incremental updating of the Common Sense Knowledge Base [16].

Use of Subjective Semantic Relatedness Measure

In general association mining is carried out to discover high support, high confidence or at least high confidence associations between items. However in this circumstance an association mining algorithm is used to find associations between concepts that likely occur in a general corpus comparatively infrequently leading to a low support threshold requirement and the confidence threshold has to also be small to capture inverse relationships. This inevitably results in some low quality associations that are likely coincidental not being filtered by the objective measures.

Thus filtering these low quality rules requires a subjective mechanism unique to the problem being considered. In this case the concept of semantic relatedness of the premise of the inference rule to the consequent is used as an experimental approach with some degree of success.

The semantic relatedness measure is calculated by considering the average value of relatedness between unique words in the premise and the consequent based on the approach specified by Jiang and Conrath which measures the semantic similarity between word pairs using corpus statistics and lexical taxonomy [17].

$$\text{Subjective Score for a Rule} = \frac{\text{aggregate similarity} * 2}{\text{premise word count} + \text{consequent word count}} \quad (4)$$

3) Sub Concept Clustering

The sub concept clustering sub system was designed with two components where each of them can be replaced with another similar component with same input and output. This view allows us to experiment with multiple possible component designs and decide the best to suit our needs and also provides highest accuracy. The two components can be namely identified as,

Sub Concept Cluster Generator

In this component, the objective is to cluster concept variables of a given concept into sub concept clusters where variables within a sub concept cluster have the same semantic meaning.

The association between two concept variables is used as the criteria for clustering variables together into sub concepts. For this, Wordnet Ontology is used to find the synonyms set of given two variables and the level of intersection between the two synonym sets is considered as the Semantic Similarity Measure (SSM). The synonym sets are taken for each concept variable and the association matrix is constructed with the SSM. Only the word pairs over the chosen threshold value is considered for clustering together. Finally the word associations with SSM over the threshold value are sorted from highest SSM to lowest.

Sub concept clusters are generated based on the word pair and existing words in the sub clusters. A word is added to a sub concept cluster only if the particular word is associated with at least two other words that are already in the sub concept cluster. The number of clusters depends on the semantic meaning differences between the variables of the clusters.

Sub Concept Cluster Merger

The sub concept clustering process further increases the number of concepts (since after sub concept clustering each sub concept is treated like a concept) which results in increased number of possible different frames from Relex2Frame. This would complicate common sense knowledge base generation process due to hardware limitations in processing. Thus Sub Concept Cluster Merger component was designed to further reduce the number of clusters through merging sub concept clusters together which have close semantic meaning.

Merging is done at a concept variable level rather than sub concept level. Each variable's semantic similarity with all the variables in sub concept cluster, which is considered for merging, is taken in to consideration in deciding whether to merge the concept variable to the second sub concept cluster.

B. Implementation

A version of the proposed common sense knowledge base generation mechanism was implemented in JAVA for the purpose testing, evaluation and validation of accuracy and efficiency.

The core association mining algorithm was based on the implementation of FP-Growth in the WEKA data mining library, while implementations for the following functionality was developed internally [18].

i. *Automatic generation of input (Attirbute Relation Files) from a given text corpus.[19]*
ii. *Incremental Updating Support for the Common Sense Knowledge Base*
iii. *Subjective Semantic Quality based Filtering of generated rules.*

VI. Results And Performance

A. Standadization of Relation-FrameNet rule base

Performance testing of the proposed Relex2frame Asynchronous Concurrent architecture was carried out on an Intel Core i5 2410M 2.3GHz computer with maximum JVM of 4GB with a batch of 5 sentences.

Table III shows the results obtained for different sentence and knowledge base buffer sizes. The sentence buffer indicates the level concurrency between processing of sentences, while KB buffer indicates memory allocated in terms of Knowledge bases in run time. It is clearly visible that the system performs well, under higher number of sentences being processed simultaneously with higher KB buffer.

Throughput of the system decreases as the number of concurrently processing sentences increases due to high competition for acquiring knowledge bases. It was found that optimum combination of sentence and KB buffers as 3 and 10, where it almost provides 1 sentence completely processed in one second. But compared to existing Relex2Frame system, this is still inefficient where it processes a sentence within 500ms.

TABLE III. PERFORMANCE FOR DIFFERENT SENTENCE AND KNOWLEDGEBASE BUFFER SIZES

Sentence Buffer	KB Buffer	Latency (s)	Approximate-Throughput(/sec)
2	2	4.6	0.43
2	3	3.4	0.59
2	5	3.4	0.59
2	10	3.4	0.59
3	5	5	0.6
3	10	3.1	0.97
4	15	4.4	0.91

Performance obtained from different rule clustering algorithms in Relex2Frame architecture, was tested based on the number of knowledge base claims for a given sentence. Higher the knowledgebase claims, lesser performance is expected as Relex2Frame system will have to process higher number of rules for same results. Table IV shows that clustering of rules such that, it minimizes the distribution of relation headers, gives the best performance by a significant margin. All the other algorithms require over 40 knowledge bases for execution, while by minimizing the distribution of relation headers average number of claims is 35.4.

TABLE IV. RULE CLUSTERING ALGORITHM PERFORMANCES

Algo. No.	Max. KB claim per relation	KB claims					Average KB Claim per sentence
		Sample 1	*Sample 2*	*Sample 3*	*Sample 4*	*Sample 5*	
1	52	52	52	52	52	38	49.2
2	45	51	44	44	53	42	46.8
3	20	49	50	48	50	41	47.6
4	35	50	44	44	51	40	45.8
5	18	45	32	32	43	25	35.4
6	45	51	49	48	51	22	44.2
7	36	49	47	44	50	28	43.6

B. Concept Word List Expansion

1) Ontological database based expansion of word lists

The first part of concept variable expansion was based on the 276 concepts and the 4716 current members of the said concepts. After this iteration 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 iteration

2) Statistical learning based on corpus for expansion of word lists

In the duration of implementation of this project only a portion of the minimized Wikipedia corpus was used to create the concept statistic. Shown below in Table V is a collection of concept predictions we were able to extract after running on few files of the said corpus.

TABLE V. CONCEPT PREDICTIONS

Concept	Suggested word	Calculated value of belongingness
$atLocation	to	8.0
	of	7.0
	from	18.0
$relTime	of	33.0
	on	8.0
	from	5.0
	during	32.0
	between	5.0
	around	3.0
$Intentionally_act	walk	1.0

Higher the calculated value of belongingness, the algorithm deems the possibility of the word belonging to the said concept to be higher. From the predictions shown in Table V, it is obvious to a human that the words to, of, from are accurately predicted to fall in to *$atLocation* and words *on, from, during, between, around* are accurately predicted to fall in to *$relTime.*

C. Common Sense Knowledge Base Generator

The initial execution results of the second iteration in Common Sense Knowledge Base generation indicate a promising increase in specificity and quality, and in our belief justifies the approach used for automatic generation of common sense rules.

The example rule shown below is extracted from results output of around 1000 rules (containing only one premise argument and one consequent argument each) for 300,000 sentences at a minimum support requirement of 0.01. While these rules indicate a clear increase in semantic value, the confidence figure itself can't be considered truly reliable due to the comparatively small size of the corpus used.

^1_Bringing: Theme ($bringing-carry, var0) ==>
^1_Removing: Theme ($removing- evacuate, $var0) <(0.51)>

This rule indicates that when an item to be brought to a destination sometimes it has first been evacuated from a source.

Consider the statement "The fireman carried the unconscious victim out of the burning building". A system using the above rule would recognize the possibility that the fireman is evacuating the victim.

VII. FUTURE WORK

As discussed earlier, the general propose rule engine Drools, performs suboptimal on low order hardware. This is due to the overheads the Drools have introduced in making the rule engine general. Since in this project the incorporation of Drools has been done in a loosely coupled way, a different rule engine can be easily integrated in to the system. A rule engine that is optimized to fire rule sets of thousands is recommended for this application since there are more than 5000 mapping rules.

Currently the statistical prediction is done per sentence. This can be enhanced to be done on a document basis, run basis or as a continued process where it is prompted only after a certain occurrence threshold is exceeded. If the aforementioned threshold method is implemented, it would be possible to fully automate the concept variable addition process. If the hardware permits, a concurrent architecture can also be incorporated in to the structure.

The following can be the logical next steps in improving the common sense knowledge base generation mechanism.

i. The quality measures that are used to filter the rules can be improved. This will result in an increase of quality of the selected rules themselves.

ii. The rule building base can be expanded to paragraph based or document based scope from the current per-sentence scope. This will result in richer rules which represent more complex common sense rules that were described in the length of paragraphs or documents.

VIII. CONCLUSION

The experiment on standardizing the existing rule base of RelEx2Frame provides us with clear conclusion that use of standard rule engine such as Drools is not the optimum approach. A standard rule engine using the Rete's algorithm with over 5000 rules in the rule base results in significant degradation of performance thus making it non-viable solution.

From the experiments done on the concept variable store of the OpenCog AGI Framework, it was observed that the above discussed methodologies were successful in adding more than 500 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. Since statistical learning is a continued process, this variable addition can be carried out further by processing more corpuses. With this success in the experiment, it is safe to conclude that the two fold methodology discussed in this paper is suitable for the expansion process of ontology based wordlists.

The proposed approach for automatically generating common sense knowledge through a corpus based data mining shows significant promise as evidenced by early results. A more mature version of the generating mechanism based improved subjective quality measures can be used on a significantly larger text corpus than the one used in testing, to generate a common sense knowledge base, that would prove useful in applications such as intelligent chat agents, text critiquing applications and other similar intelligent agents.

ACKNOWLEDGMENT

Firs we would like to thank, Department of Computer Science and Engineering of University of Moratuwa, for giving us this opportunity by providing the final year project course module. We are grateful to Dr. Ben Goertzel who gave us his fully support and we are honored to have him as our external supervisor. It is our pleasure to thank Dr. Shehan Perera, our internal supervisor. Without his guidance this would not have been possible. We would also like to thank Dr. Shantha Fernando for coordinating the course 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 and 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 honorable mention goes to our families and friends who were always behind us and encouraged us to do our best.

REFERENCES

[1] "Narual Language Processing,"[Online].Available: http://research.microsoft.com/en-us/groups/nlp/.[Accessed:21-Nov-011]

[2] "The Open Cognition Project – OpenCog Wiki," [Online]. Available: http://wiki.opencog.org/w/The_Open_Cognition_Project. [Accessed: 23-Sep -2010].

[3] "RelEx Dependency Relationship Extractor – OpenCog Wiki," [Online]. Available : http://wiki.opencog.org/w/RelEx. [Accessed: 23- Sep -2010].

[4] "RelEx2Frame – OpenCog Wiki," [Online]. Available: http://wiki.opencog.org/w/RelEx2Frame. [Accessed: 23- Sep -2010].

[5] Henry Lieberman, " Usable Artificial Intelligence Needs Common Sense Knowledge Workshop on Usable Artificial Intelligence", *in ACM Conference on Computers and Human Interaction (CHI-08), Florence, Italy, April 2008*

[6] "Commonsense knowledge base - Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Commonsense_knowledge_base. [Accessed: 13- Feb -2011].

[7] "Princeton University. WordNet – A Lexical Database for English" [Online]. Available: http://wordnet.princeton.edu/. [Accessed: 24- Feb -2010].

[8] H. T. Ng and H. B. Lee, "Integrating Multiple Knowledge Sources to Disambiguate Word Sense: An Exemplar-Based Approach", in Proc. *34th Annu. Meeting on Association for Computational Linguistics*, 1996, pp. 40 – 47.

[9] C. Niu, W. Li, R. K. Srihari, and H. Li, "Word Independent Context Pair Classification Model for Word Sense Disambiguation", in Proc. *9th Conf. on Computational Natural Language Learning*, 2005.

[10] C. Yu, L. B. Smith and S. Brandfon, "Statistical Word Learning Based on Cross-Situational Observation", in Proc. *5th Int. Conf. Development and Learning*, 2006.

[11] "OpenCyc.org." [Online]. Available: http://opencyc.org/. [Accessed: 05-Apr -2011].

[12] "Read the Web:: Carnegie Mellon University." [Online]. Available: http://rtw.ml.cmu.edu/rtw/. [Accessed: 23- Mar -2011].

[13] The JBoss Drools Team. "Drools Expert Documentation," [Online]. Available:

http://downloads.jboss.com/drools/docs/5.1.1.34858.FINAL/drools-expert/html/index.html [Accessed: 15-Nov-2010]

[14] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," ACM SIGMOD Record, vol. 29, no. 2, p. 1–12, 2000.

[15] H. Xiong, P. Tan and V. Kumar, "Mining Strong Affinity Association Patterns in Data Sets with Skewed Support Distribution," in Proc. 3rd IEEE Int. Conf. Data Mining, 2003, pp. 1-4.

[16] T. P. Hong, C. W. Lin, and Y. L. Wu, "Incrementally fast updated frequent pattern trees," Expert Systems with Applications, vol. 34, no. 4, p. 2424–2435, 2008.

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

[18] "Weka 3 - Data Mining with Open Source Machine Learning Software in Java." [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/. [Accessed: 26-Aug-2011].

[19] "Attribute-Relation File Format (ARFF)." [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/arff.html. [Accessed: 26-Aug-2011].