УДК 004.891.2

Misbakhul Munir Irfan Subakti[1, 2]
[1] Institute of Automation and Information Technologies,
Tambov State Technical University, Russia, Tambov,
[2] Department of Informatics, Institut Teknologi Sepuluh Nopember (ITS),
Indonesia, Surabaya
(Tel. (+7920)4992813, e-mail: yifana@gmail.com)

EXTENDED INFERENCING IN CONTEXTUAL ONTOLOGY-SUPPORTED RULE-BASED SYSTEMS FOR READING MATERIAL CLASSIFICATION

Abstract: Reading Material Classification (RMC) determines Determining the particular readability graded reading material from an unclassified text based on its text readability. RMC have used Natural Language Processing (NLP) methods, i.e., machine-learning-based RMC, to overcome disadvantages of using syntactic features, i.e., insufficiency for modelling the levels of text reading difficulty. Concepts can be varied somewhat between different contexts, therefore «contextual concept-variants» wanted in our ontologies which will result in Contextual Ontology (CO) by using basic NLP techniques such as stemming and word sense disambiguation. Rule-Based Systems (RBSs) are Knowledge-Based Systems whose knowledge is structured by rules. Given RMC as the test-bed, we propose to combine CO with RBSs for synergising their advantages, to improve RBS performance by adding contextual ontological processing into RBSs. This addition will also provide an extended inferencing to RBSs, so that (a) in pre-processing of inferencing, a powerful CO can be extracted, (b) in in-processing of inferencing CO can be utilised for making inferences along with some features (rule strength incremental modification, the state of memory affection and system's weight), and (c) in post-processing in inferencing, CO can be exploited. Based on the evaluation experiments, we do not claim that our proposed method is better than machine-learning-based RMC. Instead, our system performance is just on a par with them. Rather than beating those methods, we aimed to use RMC to show that adding contextual ontological processing into RBSs (RMC-RBS + CO) presents a considerable benefit for RBSs than not adding it (RMC-RBS + O). 31.25% and 25.89% improvements can be obtained for validation and testing data, respectively.

Keywords: contextual ontology, machine learning, reading material classification, rule-based systems.

1. Introduction

Reading Material Classification (RMC) classifies the input text into a particular readability graded reading material based on its text readability. RMC has an important place in education and other domains so that it has been long investigated by researchers. Past researchers have used syntactic features, e.g., word frequency, syllable and character counts per word and sentences length; to indicate the text readability. The levels of text reading

difficulty, however, cannot be modelled sufficiently by them. Natural Language Processing (NLP) machine-learning-based techniques, e.g., Support Vector Machine (SVM), Multinomial Naïve Bayes and Latent Semantic Indexing (LSI), have been used to cover the syntactic features' drawback.

The purpose of ontologies is to enable knowledge sharing and reusing (Gruber, 1993). An ontology is a description of concepts and relationships which can exist for an agent or a community of agents (Gruber, 1995). There is a contextual component in our ontologies, namely «contextual concept-variant», which can be abbreviated as «concept-variant» for convenience purposes. It is defined as a specific level of description about a concept. Furthermore, concept-variant can be distinguished with «context» which has its ordinary meaning. A «concept variant» in our system application, may not be a variant of any of the particular «concepts» arising in that application. Any concept-variant is not necessarily a variant of one of the concepts, so concept-variants should be called potential or candidate concept-variants. However, we will omit the potential or candidate for brevity. The fact that a concept can be varied somewhat between different contexts inspired us to integrate concept-variants into ontologies (namely Contextual Ontology (CO)). In obtaining the CO, some basic NLP techniques, e.g. stemming and word sense disambiguation will be used. Rule-Based Systems (RBSs) whose knowledge is structured by rules (Ignizio, 1991). They have been adapted by the internet paradigm embodied in the Web rule systems. There is still room to improve the performance of an RBS, as it depends on a particular domain in where an RBS is applied. RBSs and ontologies are used to serve different functions, even though they can be used somewhat overlapping. RBSs can be used for making inferences, whereas ontologies can be used for holding knowledge and thereby perhaps supporting the inference as well as serving other functions (such as communication). Given RMC as the test-bed, an ontology can be synergised with an RBS to do RMC, namely RMC-RBS + O. Likewise, the CO can be synergised with an RBS, namely RMC-RBS + CO, to improve RBS performance (i.e., RMC-RBS + O).

Given the classified text (the source-text) served as the training data, the source-info, i.e., concepts and concept-variants, will be extracted. Concepts and concept-variants texts along with statistics information, e.g., organisational features (such as the root name of folder/folder name) and surface-language features (such as a number, minimum, maximum and average of syllables, polysyllables, words, long-words, difficult-words and sentences), are extracted by concept extraction, concept-variant extraction and statistics extraction, respectively. Then, our rules, namely Rules, will be extracted by rule extraction from the source-text. Rules contains concepts, concept-variants and statistics information. IF-THEN rules are also generated as a portion of Rules. From the unclassified text (the input-text), Facts will be extracted similarly as in the source-text

by using the less version of rule extraction. Facts contains concepts, concept-variants and statistics information, e.g., surface-language features. RMC-RBS + O works as follows: the surface-language features, concepts and concept-variants from Facts will be fed to IF-THEN rules with supported by the source-info, then our system will fire as many rules as possible as long as the condition in the antecedent of a rule has been met. The grade level with the highest average weight will be chosen to be the classified grade level of the input-text. A weight is defined as a measure of the evidence accumulated so far for a particular grade level. Likewise, RMC-RBS + CO works similarly as in RMC-RBS + O. It utilises CO as the integration of concepts and concept-variants, rather than process them separately as in RMC-RBS + O.

K12Reader (K12Reader, 2017) provides reading material resources for K-12 Education from experienced reading teachers and curriculum specialists in the USA. In addition to K12Reader dataset, a set of English essays written by high school students from the University of the Philippines Integrated School High School Division and Philippine Science High School (UPIS, 2017), is also used as the dataset for our experiments. The performance of our method will be compared to the Readability formulas (Readability Formulas, 2017) – a well-known statistics-based widely used to evaluate RMC, for the baseline of comparison. Then, RMC-RBS + O and RMC-RBS + CO will be compared to see the improvement.

The remainder of the paper is organised as follows. Section 2 provides a detailed description of how our system works. Section 3 explains the experiments for evaluating our system's performance before we conclude the paper in Section 4, along with our future works.

2. Proposed Method Description

2.1. *Automatic Extraction*: *Concept*, *Concept-Variant*, *Contextual Ontology and Rule Extractions*

The word extractor proposed by Ahmad and Gillam (2005) produces the words that they call «concepts», i.e., a thesaurus of terms, each of which is expected to denote a concept. Their work will be used to produce our concept, i.e., lexical shared as collocated words as well as their hyponyms and synonyms which are found in the source (i.e., a text or a number of source texts) of a given domain. WordSieve and RAKE (Rapid Automatic Keyword Extraction) are the word extractor proposed by Bauer and Leake (2001, 2001) and Rose et al. (2010), respectively. They call these extracted words «contexts». We have considered this naming «contexts» is slightly inappropriate. We propose «contextual concept-variants» (which can be abbreviated to «concept-variants» for brevity) as a better name instead of «contexts» which can be defined as the frequently occurring word and a sequence of one or more words which provide a compact representation of the content of a text (or a number of source text) along with its weight

found in the source. Both concept and concept-variant definitions can be broadened into a structure, a tree and more complicated forms of these combinations (i.e., the combinations of word, structure and tree).

Figure 1 shows how RMC starts by extracting the source-info from the source-text, so that concepts, concept-variants and the statistics can be obtained by the concept, concept-variant and statistics extractions, respectively.

Figure 2 shows a flow chart for the concept extraction. From the text, the frequency of each word's occurrences (f) is calculated as well as the weirdness index (w). A measure of the use of a word in special language compared to its use in a representative corpus of general language texts is represented by weirdness index (Equation 1).

$$\text{weirdness} = \frac{N_B f_T}{(1+ f_B) N_T}. \quad (1)$$

Where from a given the word, f_T is its frequency in our text, f_B is its frequency in British National Corpus (BNC). NT is the number of all words in our text. NB is the number of all words in BNC. From the period of 1960 – 1993, BNC contains more than 100 million words. To get the lexical shared words among the words in the folder of source-text, z-score for both f and w is calculated (equation 2) as a measure of disproportionately used words.

$$z_i(\text{word}) = \frac{(\text{word}_i - \overline{\text{word}})}{\sigma_{\text{word}}}. \quad (2)$$

Where z_i is the i-th z-score for word word_i, $\overline{\text{word}}$ and σ_{word} are the mean and standard deviation of word word, respectively (Ahmad and Gillam, 2005). The words whose z-score above a given threshold are extracted from the text as the concepts.

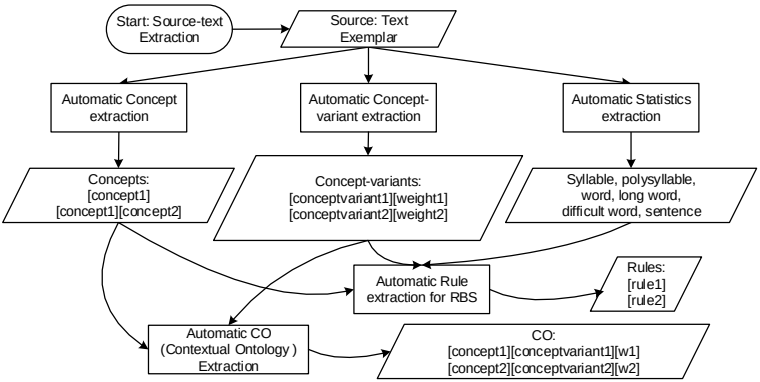

Figure 1. Source-text extraction

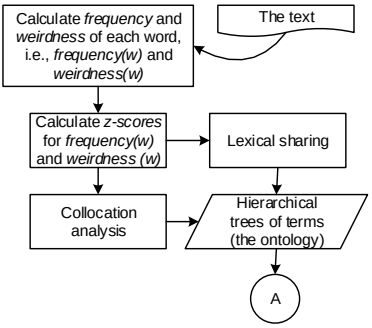

Figure 2. Concept extraction

The collocated words from Smadja (1993) are used, where they have the importance's position within their neighbourhoods, are also extracted as the concepts. Smadja has argued that significant collocates of a word are within a neighbourhood of five words either side of the word (five words to the left, L1-L5, and five words to the right, R1-R5). Metrices for quantifying the strength of the collocation are the one that isolates peakedness (U-score) of the collocation in the various positions of the neighbourhood, along with the z-score. Peakedness is described by the equation 3 as in the following.

$$\text{U-score} = \text{U}_i = \frac{\sum_{j=1}^{10} (p_i^j - \overline{p}_i)^2}{10}. \qquad (3)$$

Where p_i^j is a frequency of word word_i with word word such that they j words apart, $\overline{p}_i$ is the mean of word word (Smadja, 1993). The words whose U-score above a given threshold are extracted from the text as the concepts. The words obtained from lexical sharing and collocation analysis procedures are the concepts. Figure 2-4 show two connectors: A and B. Connector A means the concepts extracted in figure 2 will goes to CO extraction in figure 4, along with connector B, i.e., the extracted concept-variants (figure 3).

Figure 3 shows a flow chart for concept-variant extraction. WordSieve network consists of three small, interdependent levels of nodes. Level 1 is sensitised to the words that are currently in the text; level 2 keeps a record of the words that have tended to occur at different times and only slowly forget them, and level 3 works in the opposite direction of level 2. Level 3 keeps a low level of activation value until the word stops occurring and increase this value for as long as the word does not occur. These three levels then produce our concept-variants.

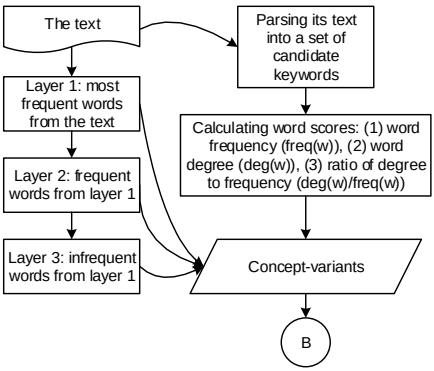

Figure 3. Concept-variant extraction

RAKE is used to get more concept-variants, where a set candidate keywords are obtained by parsing the source-text. The text from the source-text is split into an array of words by the specified word delimiters, then split into sequences of contiguous words at phrase delimiters and stop word positions. Words within a sequence are assigned the same position in the text, and together, they are treated as a candidate keyword. After every candidate word is found and the graph of word co-occurrences (i.e., the rows and columns are filled with candidate keywords, so that co-occurrence of keywords can be evaluated) is complete, a score is calculated for each candidate keyword, and it is defined as the sum of its member word scores. Equation 4 shows metrices for calculating the word scores, based on the degree and frequency of word vertices in the graph.

$$\begin{gathered} \text{freq(word)} = \text{word frequency} \\ \text{deg(word)} = \text{word degree} \\ \text{deg(word)} / \text{freq(word)} = \text{ratio of degree of frequency.} \end{gathered} \quad (4)$$

Where freq(word) is the occurrence number of word word in the text and deg(word) is the sum of the co-occurrence numbers of a word word in its row in the word co-occurrence graph. deg(word) favours words that often occur and in longer candidate words. Words that frequently occur regardless of the number of words with which they co-occur are favoured by freq (word). Words that predominantly occur in longer candidate words are favoured by deg(word) / freq(word). The score for each candidate word is computed as the sum of its member word scores.

Figure 4 shows CO extraction, where concept-variants (i.e., extracted by concept-variant extraction) will be integrated with concepts (i.e., extracted by concept extraction). Additional information, i.e., organisational features such as the folder structure of a number of text files in the source-text, is used to build CO. WordNet (WordNet, 2017) is used for the electronic thesaurus, to find the hyponyms and synonyms of a word.

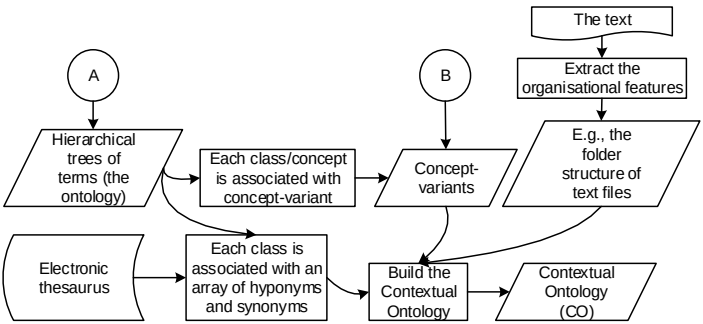

Figure 4. CO extraction

Sangers et al. (2013) also used WordNet for finding monosemous words to establish a starting context. Basic techniques of NLP such as stemming and word sense disambiguation is used in CO extraction. Algorithm 1 shows our proposed CO extraction algorithm.

Algorithm 1. Contextual Ontology (CO) Extraction

Input: concepts, concept-variants and organisational features (e.g., the folder structure of a number of text files in the source-text).

Obtaining concepts and concept-variants

1. Get concepts from concept extraction and save them to ConceptList.

E.g., ConceptList : = {[conceptA], [conceptB], …, [conceptN]}

2. Find a given number of hyponyms and synonyms of each concept from ConceptList, with the support of an electronic thesaurus, e.g., WordNet.

FOR each concept in ConceptList

Get a given number of hyponyms and synonyms of a concept, and save them to ConceptList if any.

E.g., ConceptList : = {[conceptA] ([hyponymA1], [hyponymA2], …, [hyponymAN], [synonymA1], [synonymA2], …, [synonymAN]), [conceptB] ([hyponymB1], [hyponymB2], …, [hyponymBN], [synonymB1], [synonymB2], …, [synonymBN]), …, [conceptM] ([hyponymM1], … , [synonymMN])}

3. Get concept-variants from concept-variant extraction and save them to ConceptVariantList.

E.g., ConceptVariantList : = {[conceptvariantA][weightA], [conceptvariantB][weightB], …, [conceptvariantN][weightN]}

CO before updating

4. Build the contextualised concept (CO) from ConceptList. A node in CO, namely CONode, is defined as [concept][conceptvariant][weight].

FOR each concept in ConceptList
Get a concept and add them to CONodes.
/* CONode = concept */
CONode : = [conceptA][][]
/* Add a new CONode to CONodes */
CONodes : = CONodes ∪ CONode

5. Build the CO from ConceptVariantList. Based on CONodes, check whether a concept-variant already exists in CONodes.

FOR each concept-variant in ConceptVariantList

IF a concept-variant is not in CONodes, assign a concept-variant to be a new CONode

/* CONode = concept-variant */

CONode : = [][conceptvariantA][weightA]

/* Add a new CONode to CONodes */

CONodes : = CONodes ∪ CONode

Updating CO

6. Get the organisational features such as the folder/directory structure of a number of text files in the source-text, and save it to FolderNameList. A folder name is important because it represents the class/group/categorisation of the texts, e.g., the grade levels.

FOR each concept in ConceptList and each concept-variant in ConceptVariantList

Save each concept to the folder name it belongs.

Count a number of folder/directory name from each concept and concept-variant, and save these numbers along with their folder names.

Save all the information above to FolderNameList.

FOR each folder name in FolderNameList

Get the percentage of each number of a folder name against the total number of all folder names.

/* 1st case of updating */

7. For each concept in ConceptList, find any concept-variant (from ConceptVariantList) whose the same word, if it exists, and populate a new CONode.

FOR each concept in ConceptList

IF a concept shared the same word with a concept-variant in ConceptVariantList, join a concept-variant to a concept together as a new CONode.

/* Join a concept and a concept-variant */

[conceptA][conceptvariantA][weightA] : = [conceptA] +

[conceptvariantA][weightA]

/* CONode = concept + conceptvariant */

CONode : = [conceptA][conceptvariantA][weightA]

/* Add a new CONode to CONodes */

CONodes : = CONodes ∪ CONode

/* 2nd case of updating */

8. If from the step 7 above, a CONode has an empty concept-variant and an empty weight (e.g., [conceptA][][]), fill the empty concept-variant with the folder name of its concept (from FolderNameList) and also fill the empty weight with the percentage of the folder name.

FOR each CONode in CONodes

IF concept-variant and the weight of a CONode are empty, e.g., [conceptA][][]

Fill the empty concept-variant with the folder name of its concept (from FolderNameList).

Fill the empty weight with the percentage of the folder name.

E.g., [conceptA][][] [conceptA][classA][weightA], [conceptA][classB] [weightB], [conceptA][classC][weightC], …, [conceptA][classN][weightN]

/* 3rd case of updating */

9. Based on ConceptList, by using Semantic Relatedness Metrics (SRM) (WS4J, 2017) to calculate the similarity between a concept and the folder name's concepts and concept-variants. The information about the folder names are obtained from FolderNameList, and only a limited amount of concepts and concept-variants for each folder name will be used. Populate a new CONode by combining concepts and the folder names according to their SRM. As a result, then all of the CONodes now had grade levels (i.e., it is a folder name represented a grade level) as a concept-variant, and it is put preceded the weight, i.e., [concept][conceptvariant] [grade][weight].

FOR each concept in ConceptList

Get a concept, and calculate the SRM between this concept and a limited number of concepts as well as concept-variants of the folder names (i.e., they represent classes/grade levels) from FolderNameList.

/* Calculate SRM */

E.g., [weightA] SRM between [conceptA] and concepts & concept-variants from [gradeA]

[weightB] SRM between [conceptA] and concepts & concept-variants from [gradeB]

/* Populate a new CONode */

/* by filling the empty weight with the percentage of the folder name */

E.g., [conceptA][][]

[conceptA][conceptvariantA][gradeA][weightA],
[conceptA][conceptvariantA][gradeB][weightB],
[conceptA][conceptvariantA][gradeC][weightC], …,

[conceptA][conceptvariantA][gradeN][weightN]

/* 4th case of updating */

10. As an additional feature, concept-variants from ConceptVariantList can be combined with a concept from the ConceptList.

FOR each concept-variant in ConceptVariantList
FOR each concept in ConceptVariantList
Combine a concept with concept-variants.
E.g., [conceptA][conceptvariantA][weightA] : = [conceptA] ∪ [conceptvariantA][weightA]
[conceptA][conceptvariantB][weightB] : = [conceptA] ∪ [conceptvariantB][weightB]
[conceptA][conceptvariantN][weightN] : = [conceptA] ∪ [conceptvariantN][weightN]

Output: CONodes, i.e., Contextual Ontology (CO).

Algorithm 2 shows Statistics extraction, where organisational features (such as the root name of folder and directory names) and surface-language features (such as number, minimum, maximum and average of syllables, polysyllables, words, long-words, difficult-words and sentences of texts) can be obtained.

Algorithm 2. Statistics Extraction

Input: the source-text.

Get the source-text/root name.
Get the folder/directory names.
FOR each folder/directory in the source
Calculate number, minimum, maximum and average of syllables, polysyllables, words, long-words, difficult-words and sentences

Output: statistics of the source-text and its folder/directories

---

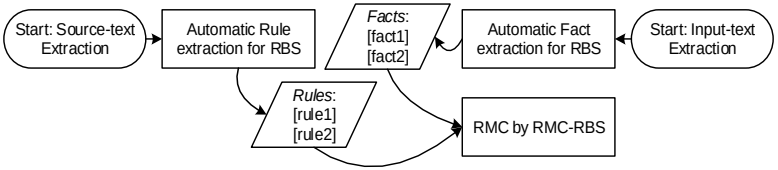

Figure 5. RMC-RBS: Rules and Facts

Algorithm 3 shows Rule extraction, where Rules and Facts can be extracted from the source-text and the input-text by rule extraction and fact extraction, respectively (figure 5).

Algorithm 3. Rule Extraction

Input: the source-text.

Obtaining knowledge

- Get statistics from the source-text by Alg. 3.3.
- Get concepts from the source-text by Alg. 3.1.
- Get concept-variants from the source-text by Alg. 3.2.

Writing the facts

- Writing statistics.

Write the folder/directory names.
FOR each folder/directory in the source
Write the folder/directory name.
Write the number, minimum, maximum and average of syllables, polysyllables, words, long-words, difficult-words and sentences.

- Writing concepts from lexical sharing procedure.

FOR each folder/directory in the source
IF lexical shared word is found
FolderName1_LexicalSharingFound : = true.
Write the lexical shared word(s) for this folder.

- Write concepts from collocation analysis procedure.

FOR each folder/directory in the source
IF collocated word is found
FolderName1_CollocationAnalysisFound : = true.
Write the collocated word(s) for this folder.

- Write concept-variants from concept-variant extraction procedure.

FOR each folder/directory in the source
IF concept-variant is found
FolderName1_ConceptVariantFound : = true.
Write concept-variant(s) for this folder.

Writing the constants
FOR each folder/directory in the source
Write the lower and upper bounds for syllables, polysyllables, words, long-words, difficult-words and sentences.

Writing the rules

FOR each folder/directory in the source, do the following where {statistics} will be assigned for syllable, polysyllable, word, long-word, difficult-word and sentence, respectively.

Write the rule name : = {statistics} name.

Write the default Confidence Factor (CF) for the rule of {statistics} name.

Write the range checking based on the lower and upper bounds along with their default CFs for the antecedent, so that the {statistics} can be found in the consequent.

Write the rule name : = {statistics} weight.

Write the default CF for the rule of {statistics} weight.

Write in the antecedent whether {statistics} can be found along with its default CF, so that in the consequent, the weight of {statistics} can be assigned.

Write the rule name : = statistics weight.

Write the default CF for the rule of statistics weight.

Write in the antecedent whether the flag variable for the statistics weight is true along with its default CF, so that in the consequent, the statistics weight can be assigned from the summation of the weights of syllables, polysyllables, words, long-words, difficult-words and sentences.

Write the rule name : = lexical sharing weight.

Write the default CF for the rule of lexical sharing weight.

Write in the antecedent whether the flag variable for lexical sharing is true and also the shared of lexical sharing (between the input-info and the source-info) can be included along with their default CFs, so that in the consequent, the lexical sharing weight can be assigned from the inclusion result.

Write the rule name : = collocation analysis weight.

Write the default CF for the rule of collocation analysis weight.

Write in the antecedent whether the flag variable for the collocation analysis is true and also the shared of collocated words (between the input-info and the source-info) can be included along with their default CFs, so that in the consequent, the collocation analysis weight can be assigned from the inclusion result.

Write the rule name : = concept-variant weight.

Write the default CF for the rule of concept-variant weight.

Write in the antecedent whether the flag variable for concept-variant is true and also the shared of concept-variants (between the input-info and the source-info) can be included along with their default CFs, so that in the consequent, the concept-variant weight can be assigned from the inclusion result.

Write the rule name : = averaged weight.

Write the default CF for the rule of averaged weight.

Write in the antecedent whether the flag variable for averaged weight is true along with its default CFs, so that in the consequent, the averaged weight can be assigned from the weights of statistics, lexical sharing, collocation analysis and concept-variant.

Output: Rules about the source.

Figure 6 shows Rules contains (1) the source-statistics, i.e., the information about organisational and surface-language features, (2) the source-info and (3) IF-THEN rules. Similarly, figure 7 shows that from the input-text, Facts can be extracted by using fact extraction (i.e., a less version of rule extraction), so that (1) the input-statistics (i.e., surface-language features) and (2) the input-info (i.e., concepts and concept-variants) can be obtained.

Figure 8 shows the small portion of Rules obtained by rule extraction. Given Rules extracted from the source-text by rule extraction, the grade level of the input-text can be assigned by inferencing. Facts from the input-text will be extracted by fact extraction first; then Facts will be used in inferencing by finding as many rules of Rules, i.e., IF-THEN rules which can be fired so that the conclusion(s) can be obtained to determine the grade levels of the input-text. The input-statistics and the input-info from Facts will be fed to IF-THEN rules with supported by the source-info, then our system will fire as many rules as possible as long the condition in the antecedent of a rule has been met. The grade level with the highest average weight will be chosen to be the classified grade level of the input-text.

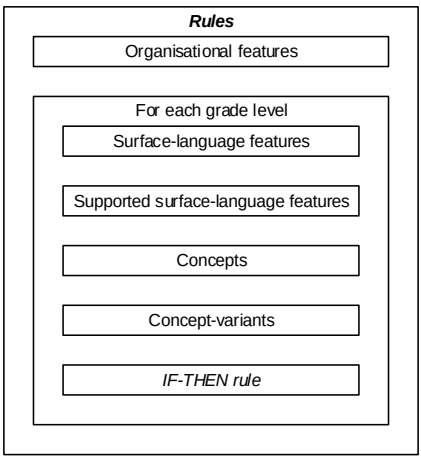

Figure 6. Rules: the component

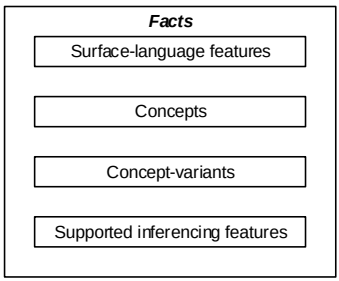

Figure 7. Facts: the component

```
<FACT>
folderName = {D:\G2, D:\G3, D:\G4, D:\G5, D:\G6, D:\G7, D:\G8, D:\G9};
// Information from each folder
// D:\G2: STATISTICS
D:\G2_SyllableNum = 2468;
…
// D:\G2: CONCEPT
D:\G2_LexicalSharing = {object, grow, push, give, violin's, … blades, louder, pulls};
…
// D:\ G2: COLLOCATION ANALYSIS
D:\ G2_CollocationAnalysis = {[timeline][][][][timeline], [sound][][][make], [make][fruit],
… [stay][][][][ground]};
…
// D:\G2: CONCEPT-VARIANT
D:\G2_ConceptVariant = {[plants][1.0], [things][1.0], [timeline][1.0], … [things fall][0.11111111]};
…
</FACT>
<CONSTANT>
// Constant from each folder
// D:\G2: STATISTICS
D:\G2_SyllableLowerBound = 244;
…
</CONSTANT>
<RULE>
// D:\G2
…
// SyllableWeight
RuleName D:\G2_SyllableWeightRule {
 RuleCF 18;
 if (D:\G2_SyllableFound = = true [CF 83]) {
  D:\G2_SyllableWeight = Inc(baseWeight, 0.25);
 }
}
…
// Averaged Weight: Statistics, Lexical Sharing, Collocation Analysis and ConceptVariant
RuleName D:\G7_AveragedWeightRule {
 RuleCF 80;
 if (D:\G7_AveragedWeightEnabled = = true [CF 80]) {
```

```
 D:\G7_AveragedWeight =
Avg(D:\G7_StatisticsWeight,
D:\G7_LexicalSharingWeight,
D:\G7_CollocationAnalysisWeight,
D:\G7_ConceptVariantWeight);
 }
}
…
</RULE>
```

Figure 8. A small portion of Rules

2.2. *RMC by RBS*: *RMC-RBS + O*

Facts and Rules for RMC-RBS + O contain statistics, concepts and concept-variants. Concepts and concept-variants are separated, not as an integral part, so RMC-RBS + O will use concepts and concept-variants as what it exists in inferencing.

Giving the information from Facts, the weights for each grade level of the source-text (i.e., the weights of statistics, lexical shared, collocation analysis and concept-variant) will be produced by feeding them to the rules of IF-THEN rule from Rules in inferencing, and then the obtaining highest weight will be chosen as the final result. Inferencing is performed by using the combination of the forward and the backward chaining. The ordering of the rules is unimportant. The user can add up more rules in any arbitrary place later. The fact(s) and the antecedent part(s) of a rule will be evaluated. If they are met, the consequent part(s) of the rule can be fired (forward chaining). If they are not met, then other rules which produce the consequent part(s) that in turn can be evaluated with the previous unmet antecedent(s) will be processed (backward chaining). Our system read each fact from the input-text, feeds it to the rules, fires as much rules as possible as long the condition in the antecedent of a rule has been met. As a result, the grade level with the highest average weight (which is ranging from 0.0 to 1.0) of statistics, lexical shared, collocation analysis and concept-variant weights will be chosen to be the classified grade level the input-text belongs. Our system also updates the facts designated for each input-text, so that the next inferencing will be performed conveniently for that particular input-text. The weights will be incrementally developed for different grade classifications during rule firing.

2.3. *RMC by RBS*: *RMC-RMBS + CO*

For inferencing purposes, RMC-RBS + O uses concepts and a concept-variants as separated part, however, in RMC-RBS + CO, they will be used together as an integral part, i.e., CO.

Giving Facts and Rules, by using a similar approach which has been applied in RMC-RBS + O, the information from Facts will be fed to the rules of IF-THEN rule from Rules in inferencing to get the final weight. However, instead of using concepts and concept-variants directly for inferencing, RMC-RBS + CO will calls CO extraction first to extract CO in pre-processing for inferencing. To extend the inferencing as happened in RMC-O, we have considered that by treating concepts as the gradation of concept-variants, its weights can be utilised for some purposes, e.g., in this case, is RMC. These weights will be exploited by our inclusion function which makes used them to produce a measure indicating their relation. Weights belong to concept-variants in RMC-O themselves can be seen useful, as in inclusion procedure for inferencing. However, we have considered that weights from the integrated ones (i.e., CO) would be better than from the separated ones (i.e., concept-variants), in addition to the existence of the variation of weights of concept-variants (i.e., it is a bonus point after the gradation of concept-variants themselves). It also shows the significant difference between a concept and a concept-variant. So, inferencing in RMC-RBS + CO works as in RMC-RBS + O, but the evaluation of contextual information will be based on CO. However, we also have considered that only exploiting the weights within a CO can be continued by exploiting the words, i.e., concepts and concept-variants themselves, as in the following.

Aside from performing the inferencing by using CO (among others), we are looking for any additional advantages that can be obtained from SRM (WS4J, 2017) since its support for CO extraction. Since it calculates the similarity between two words, then it turns out that CO components (i.e., [concept] and/or [concept-variant]) can be evaluated individually. SRM is then will be used further for determining the weight for concept similarity (SRM concept weight), for concept-variant similarity (SRM concept-variant weight), and for concept-concept-variant similarity (SRM concept-concept-variant weight), namely, SRM concept procedure, SRM concept-variant procedure and SRM concept-concept-variant procedure, respectively, between the source-info (i.e., concepts and concept-variants) from the source-text and the input-info (i.e., concepts and concept-variants) from the input-text.

SRM concept. The input-info will be matched with the source-info's [concepts]. The partial matched is allowed. A weight obtained is simply 0 (no matched) or 1 (it is matched). SRM concept-variant. The input-info will be matched with the source-info's [concept-variants]. The partial matched is allowed. A weight obtained is simply 0 (no matched) or 1 (it is matched). SRM concept-concept-variant. The input-info will be matched with the source-info's [concepts] and [concept-variants]. The partial matched is allowed. A weight obtained is the gradation of 0 (no matched at all) to 1 (it is matched completely).

An example of SRM procedures as follows: when the source-info has {…, [grand][canyon][G7][0.62], [grand][villa] [estate][G8][0.77], [grand][canyon][state][G9-10][0.66], …} and the input-info has {…, [canyon][Arizona][0.20], …} then by SRM concept, there are two cases are matched: a particular concept [canyon] is matched with both [grand] [canyon][G7][0.62] and with [grand][canyon][state][G9-10][0.66]. With [grand][canyon][G7][0.62], the weight assigned is (0.62 + 0.20) / 2 = 0.41 for concept-variant [G7], and with [grand][canyon][state][G9-10][0.66], the weight assigned is (0.66 + 0.20) / 2 = 0.43 for concept-variant [G9-10]. Since 0.43 > 0.41, then from [grand][canyon][state][G9-10][0.66] the particular result [G9-10] (i.e., Grade 9-10), will be assigned to the input-info if we want to use only SRM concept. By SRM concept-variant, concept-variant [Arizona] does not match with any particular CONode in CO, so that the weight assigned are 0 (zero) for all of [G7], [G8] and [G9-10]. For SRM concept-concept-variant, the result is similar to of SRM concept, since only concept [canyon] can be matched and there is no match for concept-variant [Arizona], i.e., for concept-variant [G7], [G8] and [G9-10], the weights 0.41, 0 and 0.43 are assigned to the input text, respectively.

Three SRM weights (i.e., the weights of SRM concept, SRM concept-variant and SRM concept-concept-variant, respectively) will be served as the additional weights in RMC-RBS + CO. SRM procedures will use some basic techniques of NLP, e.g., stemming and word sense disambiguation.

3. Extended Inferencing

We propose an approach to extend the inferencing, i.e., pre-processing, in-processing and post-processing of inferencing to improve the performance of RMC-RBS.

3.1. *Pre-processing of Inferencing*

Alongside an ordinary extraction of information (i.e., statistics by statistics extraction) which has syntactic features, both Facts and Rules will get concepts (by concept extraction) and concept-variants (by concept-variant extraction) by rule extraction (note: fact extraction is a less version of rule extraction). So, both Facts and Rules will have ontologies (concepts and concept-variant) added to their structure, alongside their statistics. Figure 9 shows a diagram of a simplified version of obtaining Rules and Facts. The numbers in arrows represent the order of processes which take place in Inference Engine and Working Memory. It works as follows: (1) from the source-text, by statistics extraction, (2) statistics can be obtained, as well as by (3) concept extraction which (4) extract concepts, by (5) concept-variant extraction which (6) extract concept-variants. (7) statistics, (8) concepts and (9) concept-variants are used by rule extraction to (10) generate Rules. Likewise, from the input-text, by (11) statistics extraction, (12) statistics can be extracted, as well as by (13) concept extraction which (14)

extract concepts, by (15) concept-variant extraction which (16) extract concept-variants. (17) statistics, (18) concepts and (19) concept-variants are used by rule extraction (i.e., it has been used as fact extraction) to (10) generate Facts.

When dealing with contextual information in its inferencing, rather than evaluating concepts and concept-variants directly, our system will use CO (which is an integration of concept-variants into concepts) instead. So, in Working Memory, CO will be extracted by CO extraction before the inferencing began. CO will be built by utilising some basic techniques of NLP, e.g., stemming and word sense disambiguation. A portion of our idea of the previous work, i.e., VCIRS (Subakti, 2005) and its revision of knowledge building (selecting the condition from the cases and cases reconstructing perspectives) (Subakti, 2006) can be extended in this case. The idea of VCIRS which has an ordered of variables/nodes/rules based on their importance and their usage can be used in CO extraction. However, rather than have a 'strict' order with totally different entities, i.e., different variables, in ContextOntoRBS we will build the gradation of a concept, i.e., concept-variants which can be put into CO. It should be noted that our previous works only gave a portion of an idea. However, the complete idea and the big picture of this thesis are completely our novel method as well as its application.

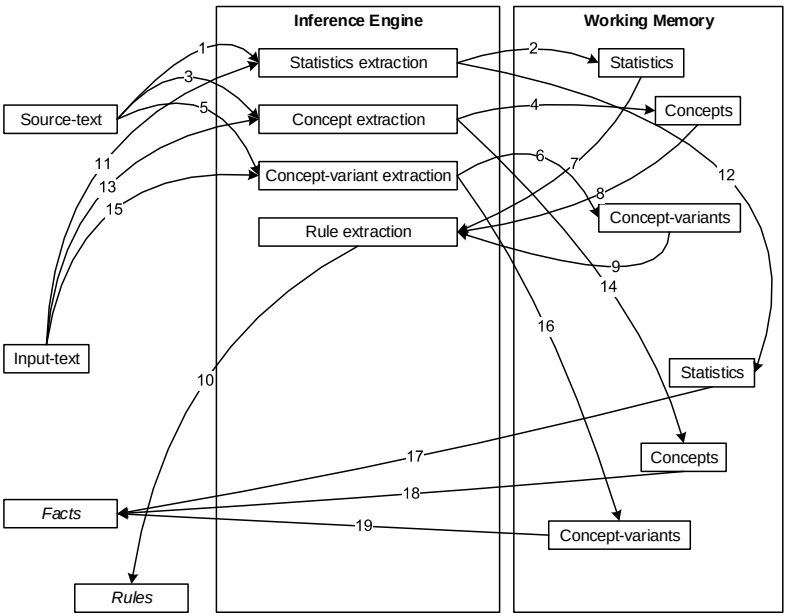

Figure 9. RMC-RBS + O: obtaining Rules and Facts

The original purpose of an ontology which employed in RMC-RBS + O is for sharing and reusing knowledge. However, RMC-RBS + O can do the (specific) text classification, i.e., RMC. It is an added value for an ontology used by RBS, i.e., its usage has been extended.

One feature of CO in RMC-RBS + CO can be seen as the extended coverage of an ontology. It is because concept-variants are the gradation of a concept. Concept-variants are not just a general variation of a concept, but it is the gradation of a concept. Empowered by its weight, this integration of concept-variants into concepts can provide a range of broader coverage than a (general) ontology. The weight itself can be seen as the coverage measurement from a concept for its concept-variants.

3.2. *In-processing of Inferencing*

In any rule of IF-THEN rule, any function which manipulates the information from both the input-text and the source-text is allowed. E.g., In(), which evaluate whether an entity is included/contained in other entity and then exploiting their weights to produce a measure for indicating their relation, Inc(), which increase the value of its first parameter by its second parameter, Avg(), which average the value o entities from given parameters, and so on. Our system allows the immediate change of value, i.e., weight, in any rule when it is triggered by any rule firing.

The rule strengths will be incrementally modified caused by success/failure in rule-firing. E.g., by calling the functions above (In(), Inc(), Avg(), etc.), the strength of rule (i.e., weight of grade level: statistics, concept, etc.) will updated/modified incrementally.

When a rule is fired, changes also happen in Working Memory, Since, we allow the immediate change of value, the changed state of memory will affect further rule firing, but it will not fire any rule being processed now. The changed state of memory will only update a particular weight in the rule when that particular weight has been updated in the state of memory.

The weights will be incrementally developed for different grade level classifications during rule firing.

As an example, a very small portion of diagram about our extended inferencing can be seen in figure 10. Up to this stage, the inferencing is processing rule G7Lex, where its antecedent calls function In() to evaluate whether fact-A's concepts and fact-B's concept- variants are included/contained in the source-text G7's concepts (i.e., lexical sharing's concepts). Assumed it is included, then the weight G7LexWeight will be updated with the weight produced by function In(), then the weight in rule G7Lex's consequent is also updated. Next, rule G7Col is processed similarly by calling function In() which evaluate whether fact-A's concepts

and fact-B's concept-variants are included in G7's concepts (i.e., collocation analysis concepts). Assumed it is included, then the weight G7ColWeight will be updated with the weight produced by function In(), then the weight in rule G7Col's consequent is also updated. Next, rule G7ConVar is also processed by calling function In() which evaluate whether fact-A's concepts and fact-B's concept-variants are included in G7's concept-variants. Assumed it is included, then the weight G7ConVar will be updated with the weight produced by function In(), then the weight in rule G7ConVar's consequent is also updated. Giving a supported surface-language feature, i.e., G7AverageWeightEnabled is set to true, rule G7AveragedWeight evaluates its antecedent (whether G7AveragedWeight is true or not) and this expression will be true so that this rule can be fired. In its consequent, rule G7AveragedWeight will call function Avg() which calculates the average of G7StatisticsWeight (it is assumed that it has been updated in the previous stage), G7LexWeight, G7ColWeight and G7ConVarsWeight weights, so that G7AveragedWeight can be obtained ad updated, then the weight in rule G7AveragedWeight's consequent is also updated. The rule strengths will be incrementally modified by success/failure in rule firing (in this case, all were successes), all of those rules fired gave changes in Working Memory and these changed states of memory would affect further rule firing, and these weights would be incrementally developed during rule firing for different grade classifications.

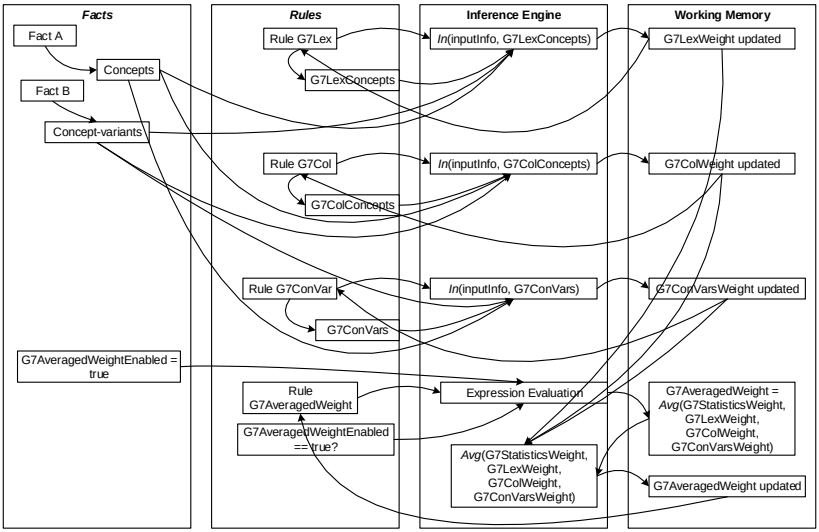

Figure 10. A small portion of our extended inferencing

3.3. *Post-processing of Inferencing*

Since both an ontology (i.e., concept-variants) and CO have the weight attached in each concept-variants and CO, which indicated the significant degree of respective concept-variants and CO, then these weights can be used directly for any purpose of future inferencing. E.g., updating rules in Rules or modifying CO.

For CO, since it is the gradation of concepts, i.e., concept-variants, then its weight can be utilised for any purpose as an interesting feature. E.g., an alternative view of grade levels.

4. The Experiments

K12Reader dataset has 186 files in the 8-grade levels. There are 80 files used in cross-validation, where those 80 files are not yet differentiated into training and validation sets. In each iteration of cross-validation, a random partitioning into 10 subsets has done, taking one of the ten to be the validation subset. The 10 results from the folds then can be averaged for producing a single estimation. UPIS dataset has 399 files in the 6-grade levels. There are 240 files used in cross-validation. In each iteration of cross-validation, a random partitioning into 10 subsets has done, taking one of the ten to be the validation subset. The 10 results from the folds for K12Reader and UPIS, respectively, can be averaged for producing a single estimation. The rest of the files from the test set of both datasets (i.e., 106 and 159 files for K12Reader and UPIS, respectively) will be used for our system's evaluation.

4.1. *RMC-RBS + O*

Based on the validation data, figure 11–12 shows all the results from the 10-fold cross-validation for K12Reader and UPIS, respectively, which produced by Readability Formulas (RF) (Readability Formulas, 2017) and RMC-RBS + O, respectively. The performance accuracy averages of RMC-RBS + O are better than of RF, i.e., 27.50% against 10.89% (K12Reader) and 27.50% against 19.94% (UPIS). Likewise, for the testing data, the results can be seen in figure 13–14, where the performance accuracy averages of RMC-RBS + O are better than of RF, i.e. 43.40% against 4.99% (K12Reader) and 25.66% against 20.31% (UPIS). However, the result from K12Reader is far better (it is almost double, 43.40% against 25.66%) than from UPIS. Even though it only has a few training data in its dataset (i.e., K12Reader training data only has 72 files compared to 216 files contained in the training data of UPIS dataset), RMC-RBS + O seems work nicely. However, more investigations are needed to understand this phenomenon.

No	Experiment	Accuracy	
		Readability Formulas	RMC-RBS+O
1	1st fold	21.43%	50.00%
2	2nd fold	14.29%	12.50%
3	3rd fold	16.07%	25.00%
4	4th fold	8.93%	25.00%
5	5th fold	5.36%	12.50%
6	6th fold	5.36%	25.00%
7	7th fold	12.50%	12.50%
8	8th fold	12.50%	50.00%
9	9th fold	7.14%	25.00%
10	10th fold	5.36%	37.50%
Average		**10.89%**	**27.50%**

Figure 11. RMC-RBS + O: K12Reader, validation data

No	Experiment	Accuracy	
		Readability Formulas	RMC-RBS+O
1	1st fold	18.45%	20.83%
2	2nd fold	25.00%	33.33%
3	3rd fold	22.62%	25.00%
4	4th fold	19.05%	29.17%
5	5th fold	19.05%	20.83%
6	6th fold	17.26%	25.00%
7	7th fold	26.19%	20.83%
8	8th fold	21.43%	37.50%
9	9th fold	13.10%	20.83%
10	10th fold	17.26%	41.67%
Average		**19.94%**	**27.50%**

Figure 12. RMC-RBS + O: UPIS, validation data

No	Experiment	Accuracy	
		Readability Formulas	RMC-RBS+O
1	1st fold	4.99%	45.28%
2	2nd fold		46.23%
3	3rd fold		44.34%
4	4th fold		43.40%
5	5th fold		38.68%
6	6th fold		43.40%
7	7th fold		40.57%
8	8th fold		43.40%
9	9th fold		45.28%
10	10th fold		43.40%
Average		**4.99%**	**43.40%**

Figure 13. RMC- RBS + O: K12Reader, testing data

No	Experiment	Accuracy	
		Readability Formulas	RMC-RBS+O
1	1st fold	20.31%	23.90%
2	2nd fold		25.79%
3	3rd fold		27.04%
4	4th fold		24.53%
5	5th fold		23.90%
6	6th fold		27.04%
7	7th fold		27.04%
8	8th fold		25.16%
9	9th fold		26.42%
10	10th fold		25.79%
Average		**20.31%**	**25.66%**

Figure 14. RMC- RBS + O: UPIS, testing data

4.2. *RMC-RBS + CO*

Based on the validation data, figure 15–16 shows all folds' results for validation data of K12Reader and UPIS, respectively, produced by RF and RMC-RBS + CO, respectively. The performance accuracy averages of RMC-RBS + CO are better than of RF, i.e., 17.50% against 10.89% (K12Reader) and 72.50% against 19.94% (UPIS). Likewise, for the testing data, the results can be seen in figure 17–18, where the performance accuracy averages of RMC-RBS + CO are better than of RF, i.e., 38.49% against 4.99% (K12Reader) and 72.08% against 20.31% (UPIS). The performance accuracy of RMC-RBS + CO's result for UPIS if far better (it is more than double, 72.08% against 38.49% for the testing data, and it is also more than quadruple, 72.50% against 17.50% for validation data) than that of K12Reader. We suggest that is because the more data of training and validation are provided for UPIS than for K12Reader, and also more cohesive topics and words are available for UPIS than for K12Rader, then the result will be better than otherwise. However, once again, more investigations are needed to understand this phenomenon.

No	Experiment	Accuracy	
		Readability Formulas	RMC-RBS+CO
1	1st fold	21.43%	37.50%
2	2nd fold	14.29%	25.00%
3	3rd fold	16.07%	12.50%
4	4th fold	8.93%	0.00%
5	5th fold	5.36%	0.00%
6	6th fold	5.36%	12.50%
7	7th fold	12.50%	25.00%
8	8th fold	12.50%	12.50%
9	9th fold	7.14%	12.50%
10	10th fold	5.36%	37.50%
Average		**10.89%**	**17.50%**

Figure 15. RMC-RBS + CO: K12Reader, validation data

No	Experiment	Accuracy	
		Readability Formulas	RMC-RBS+CO
1	1st fold	18.45%	79.17%
2	2nd fold	25.00%	79.17%
3	3rd fold	22.62%	79.17%
4	4th fold	19.05%	70.83%
5	5th fold	19.05%	54.17%
6	6th fold	17.26%	75.00%
7	7th fold	26.19%	58.33%
8	8th fold	21.43%	83.33%
9	9th fold	13.10%	66.67%
10	10th fold	17.26%	79.17%
Average		**19.94%**	**72.50%**

Figure 16. RMC-RBS + CO: UPIS, validation data

No	Experiment	Accuracy	
		Readability Formulas	RMC-RBS+CO
1	1st fold	4.99%	35.85%
2	2nd fold		48.11%
3	3rd fold		36.79%
4	4th fold		37.74%
5	5th fold		39.62%
6	6th fold		29.25%
7	7th fold		35.85%
8	8th fold		40.57%
9	9th fold		41.51%
10	10th fold		39.62%
Average		**4.99%**	**38.49%**

Figure 17. RMC-RBS + CO: K12Reader, testing data

No	Experiment	Accuracy	
		Readability Formulas	RMC-RBS+CO
1	1st fold	20.31%	74.21%
2	2nd fold		79.25%
3	3rd fold		74.21%
4	4th fold		71.07%
5	5th fold		72.33%
6	6th fold		72.33%
7	7th fold		71.07%
8	8th fold		69.81%
9	9th fold		71.07%
10	10th fold		65.41%
Average		**20.31%**	**72.08%**

Figure 18. RMC-RBS + CO: UPIS, testing data

4.3. *Contextual Information Significance*

The number of files is taken into account to make a fair comparison, i.e., the result will be in weighted averages. Figure 19–20 shows a comparison of the results to show how significance the integration of contextual information into ontologies to improve their performance accuracies for validation and testing data, respectively. Our proposed method beats the performance accuracy of RF in both validation and testing data. For the validation data, without integrating contextual information, RMC-RBS + O obtained 27.50%, a gap of 9.82%, compared to RF, i.e., 17.68%. By integrating contextual information, RMC-RBS + CO able to get 58.75%, a gap of 41.07%, compared to RF, i.e., 14.18%. It means there is an increment of 31.25% (i.e., 58.75%-27.50%). When it comes to the testing data, the performance accuracy improvement can also be seen better as follows: without integrating contextual information, RMC-RBS + O obtained 32.75%. By integrating contextual information, RMC-RBS + CO able to get 58.64%, an increment about 25.89%, compared to RMC-RBS + O.

No	Dataset	Number of Files	Accuracy: Validation Data		
			Readability Formulas	RMC-RBS+O	RMC-RBS+CO
1	K12Reader	80	10.89%	27.50%	17.50%
2	UPIS	240	19.94%	27.50%	72.50%
Weighted Average			**17.68%**	**27.50%**	**58.75%**

Figure 19. Weighted average comparison: validation data

No	Dataset	Number of Files	Accuracy: Testing Data		
			Readability Formulas	RMC-RBS+O	RMC-RBS+CO
1	K12Reader	106	4.99%	43.40%	38.49%
2	UPIS	159	20.31%	25.66%	72.08%
Weighted Average			**14.18%**	**32.75%**	**58.64%**

Figure 20. Weighted average comparison: testing data

4.4. *CO-based RMC and Machine-learning-based RMC Comparison*

By using UPIS dataset, RMC-RBS + CO obtained 72.50% and 72.08% performance accuracies for validation and testing data, respectively. As for the comparison purposes, the best results from machine-learning-based RMC as follows: 63%-79% (Multinomial Naïve Bayes, Collins-Thompson and Callan (2004)), 75.4% (Expectation Minimisation, Si and Callan (2001)), 86.81% (validation data) and 87.16% (testing data) by SVM (Liu et al, 2004), 75% precision and 87% recall (n-gram language models of SVM, Schwarm and Ostendorf (2005)), 79.72% (Decision Tree, Wang (2006)), 76.18% (Naïve Bayes, Wang (2006)), 84.06% (SVM, Wang (2006)) and 88% (LSI, Landauer (2011)). From these results comparisons, we do not claim that our proposed method is better than machine-learning-based RMC, our system performance is just on a par with them.

5. Conclusion and Future Work

RMC-RBS + CO added contextual ontological processing into RBSs, i.e., CO, while RMC-RBS + O does not have such contextual ontological processing, i.e., it is only utilised an ontology.

By using weighted average calculation, 27.5% and 58.75% performance accuracies can be produced by RMC-RBS + O and RMC-RBS + CO, respectively, for the validation data. It shows that 31.25% accuracy improvement can be obtained for validation data. For the testing data, 32.75% and 58.64% can be produced by RMC-RBS + O and RMC-RBS + CO, respectively. An impressive 25.89% accuracy improvement (more than a quarter) can be obtained for testing data. The accuracy improvement shows more impressive by observing that RF performs poorly, being only able to achieve 14.18% accuracy performance so that there is an impressive

gap of 11.71% between the accuracy performance of RF and the improvement of our system. The gap that almost reaches the values of RF itself (11.71% against 14.18%). So, it shows that added CO into an RBS does improve the quality of performance of an RBS.

Even without the contextual information, RMC-RBS + O able to beat RF by 9.82% (i.e., 27.5%-17.68%) and 18.57% (i.e., 32.75%-14.18%) for validation and testing data, respectively. By added CO into an RBS, RMC-RBS + CO able to increase the gaps by 41.07% and 44.46% for validation and testing data, respectively.

The usage of an ontology can be extended, so that, it can be used for RMC, not just for sharing and reusing knowledge anymore. The inferencing in RBS can also be extended to improve the performance of RMC by RBS which employed the contextual ontological processing. It can be concluded that added contextual ontological processing into an RBS and also utilised the extended inferencing afterwards are beneficial for the field of RBSs.

Integration and exploitation of rule structure in contextual ontology structure, e.g., Rules, would be beneficial to enhance the structure of CO for a better use.

References

1. **Ahmad, K.** and Gillam, L. (2005) Automatic Ontology Extraction from Unstructured Texts. Lect. Notes in Comp. Sci., 3761. – P. 1330 – 1346.

2. **Bauer, T.** and Leake, D. B. (2001) WordSieve: A Method for Real-Time Context Extraction. Lect. Notes in Comp. Sci., 2116. – P. 30 – 44.

3. **Collins-Thompson**, K. and Callan, J. (2004) A Language Modelling Approach to Predicting Reading Difficulty. In: Proceedings of HLT/NAACL 2004. – P. 193 – 200.

4. **Gruber, T. R.** (1993) A Translation Approach to Portable Ontologies. Knowledge Acquisition, 5 (2). – P. 199 – 220.

5. **Gruber, T. R.** (1995) Toward Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal of Human-Computer Studies, 43 (4-5). – P. 907 – 928.

6. **Ignizio, J. P.**, (1991) Introduction to Expert Systems: The Development and Implementation of Rule-based Expert Systems. McGraw-Hill International Editions.

7. **K12Reader**. (2017) Available: http://www.k12reader.com [Accessed 14.09.2017].

8. **Landauer, T. K.** (2011) Pearson's Text Complexity Measure. Pearson's White Papers. – 2011.

9. **Liu, X.-Y.,** Croft, W. B., Oh, P. and Hart, D. (2004) Automatic Recognition of Reading Levels from User Queries. In: Proceedings

of Sheffield SIGIR 2004. The 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Sheffield, UK. – P. 548–549.

10. **Readability** Formulas. (2017) Available: http://www.readability-formulas.com [Accessed 14.09.2017].

11. **Rose, S.,** Engel, D., Cramer, N. and Cowley, W. (2010) Automatic Keyword Extraction from Individual Documents. Text Mining: Applications and Theory. John Wiley & Sons. – P. 1 – 20.

12. **Sangers, J.,** Frasincar, F., Hogenboom, F. and Chepegin, V. (2013) Semantic Web Service Discovery Using Natural Language Processing Techniques. Expert Systems with Applications, 40 (11). Tarrytown, NY, USA: Pergamon Press, Inc. – P. 4660 – 4671.

13. **Schwarm, S. E.** and Ostendorf, M. (2005) Reading Level Assessment Using Support Vector Machines and Statistical Language Models. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, Assoc. for Comp. Linguistics. – P. 523 – 530.

14. **Si, L.** and Callan, J. (2001) A Statistical Model for Scientific Readability. In: Proceedings of the 2001 ACM CIKM. 10th International Conference on Information and Knowledge Management, Atlanta, GA, USA. – P. 574 – 576.

15. **Smadja, F.** (1993) Retrieving Collocations from Text: Xtract. Computational Linguistics, 19(1). Oxford University Press. – P. 143 – 178.

16. **Subakti, I.** (2005) A Variable-Centered Intelligent Rule System. In: Proceedings of the 1st Annual International Conference: Information and Communication Technology Seminar (ICTS2005), 1 (1), Surabaya, Indonesia, 11 August 2005. Surabaya-Indonesia: Sepuluh Nopember Institute of Technology (ITS). – P. 167 – 174.

17. **Subakti, I.** (2006) Some Revisions in VCIRS and Cases Reconstructing Perspectives. In: Proceedings of the 2nd Annual International Conference: Information and Communication Technology Seminar (ICTS2006), 1 (1), Surabaya-Indonesia, 29 August 2006. Surabaya, Indonesia: Sepuluh Nopember Institute of Technology (ITS), Surabaya-Indonesia. – P. 233 – 238.

18. **University** of the Philippines Integrated School (UPIS). (2017) Available: http://www.upis.upd.edu.ph [Accessed 14.09.2017].

19. **Wang, Y.-L.** (2006) Automatic Recognition of Text Difficulty from Consumers Health Information. Computer-Based Medical Systems, 2006. CBMS 2006. 19th IEEE Int'lSymposium on, 2006. – P. 131 – 136.

20. **WordNet** Similarity for Java (WS4J). (2017) Available: https:// code.google.com/p/ws4j [Accessed 14.09.2017].

21. **WordNet.** (2017) Available: https://wordnet.princeton.edu/ [Accessed 14.09.2017].