Using Rhetorical Structure Theory in Automatic Text Summarization for Marcu-Authored Documents

Allan Borra, Almira Mae Diola, Joan Tiffany T. Ong Lopez, Phoebus Ferdiel Torralba, and Sherwin So
College of Computer Studies
De La Salle University
2401 Taft Avenue, Manila
allan.borra@dlsu.edu.ph, [apany.ong.lopez, phoebustorralba]@gmail.com, [sherwinso16, mhiagurl]@yahoo.com

ABSTRACT

An automatic text summarizer system using Rhetorical Structure Theory (RST) is discussed. RST is a formalism developed by Mann and Thompson (1988) that allows for discourse analysis of multi-sentence text. The descriptive representation of RST to texts was exploited for summarization. Some modules of the summarizer were designed on Daniel Marcu's composition style primarily on technical papers but the system was also evaluated to other domains. Preliminary test shows that the system may work well regardless of domain, as long as it has similar composition or writing styles with Marcu-authored documents. Subjective manual evaluation shows higher quality of the system's output compared to Microsoft Word's Autosummarize and Copernic Summarizer. Automated evaluation using Sentence Recall versus gold standards was done although it was inconclusive due to disparity between human-authored, which uses paraphrasing method, and system-authored summaries, which uses extraction method.

Keywords:

Automatic text summarization, natural language processing, keyword extraction, rhetorical structure theory, coherence.

1. INTRODUCTION

Given an information source, Automatic Text Summarization (ATS) produces the cluster of information, which is most relevant to the needs of the user [1]. The three approaches in ATS are the shallow approach, the deeper approach, and the hybrid approach. A common problem with these approaches is that they do not have sufficient coherence. Coherence is the way the parts of the text gather together to form an integrated whole. With today's automatic text summarizers, statements are commonly non-sequitur, meaning a statement that does not follow logically from what preceded it. There is no smooth transition from one idea to another. The driving force of the research is the development of a system that will be able to summarize a given document while still maintaining coherence and saliency in the text. To do this, the system architecture integrated two main existing techniques in ATS: keyword extraction and discourse analysis based on Rhetorical Structure Theory (RST).

2. RELATED SYSTEMS

SUMMARIST [2] aims to generate both extracts and abstracts from any domain. Its technique lies on the 'equation': summarization = topic identification + interpretation + generation. The first step is to identify the most important and central topics in the paper. The interpretation stage processes the topics, rephrases and compresses them, removes redundancies and merges topics into more general ones (e.g. simple concept generalization, script identification). Generation aims to reformulate the interpreted data into new text.Its architecture is designed such that the output you want depends on how much processing you do on the text. SUMMARIST's architecture is designed such that the output depends on the amount of processing that was done on the text.

Text Analyst [3] is a text analysis software which performs a variety of natural language functions, among them text summarization. It uses linguistic and neural network technologies, which ensures high speed and accuracy in the analysis of unstructured texts. It automatically generates a semantic network of the text, which is used in scoring individual sentences.

Summons [1] is a multi-document summarizer for the news domain. It uses a template with instantiated slots of predefined semantics; from this, it generates the summary using a sophisticated natural language generation phase. This includes a content selection substage, a sentence planning substage and a sentence generation stage. Because of the well-defined semantics in the templates, this system is said to produce summaries of a quality that approaches that of human abstracts.

PARE [4] describes a system of representation for ideas present in a text. It is based on syntactic and semantic relationships between words. It is used in the program called PARE (Pruer and Redundancy Eliminator), which is a multi-document summarizer. The system is based on the concept of a semantic network. It creates a set of nodes—a graph, which represents the relationships between single-word ideas. The nodes are connected by arcs which contain information describing the relationship between the words, These relationships are determined by using a link-grammar based parser. Additionally, PARE also uses coloring schemes—colors are assigned so that all nodes with the same color are part of the same idea, thus creating larger groups of words.

3. RHETORICAL STRUCTURE THEORY

Rhetorical Structure Theory (RST) was initially developed by Mann, Thompson and Matthiessen in 1983 to allow a formal representation of discourse structure to texts. This also provides for representing coherence in texts. Originally, this was developed as part of studies for automatic text generation systems. It conceives a variety of possible structures, which could be considered as "building blocks" that a text document might have. These "blocks" may be classified into two levels, the principal one dealing with "nuclearity" and "relations" (often called coherence relations in the linguistic literature,) and second level of structures, called schemas [5].

3.1 Nucleus and Satellite Relations

Nucleus and Satellite Relations is the first level of representation of RST. Given two spans of texts (usually adjacent, but there are exceptions), a relation can be established between the two spans where one has a specific role relative to the other. An example of a structure is: "Claim" which is followed by "Evidence". RST claims that a "Claim" is more essential to the text than the "Evidence". Given this, a "Claim" is labeled as a nucleus while the "Evidence" is labeled as a satellite. The ordering of the text spans do not matter but the relations normally dictates the likely ordering of text spans. Figure 1 shows Mann and Thompson's list of relations. These identified relations are relatively comprehensive but is still being expanded to address newer and unique relations.

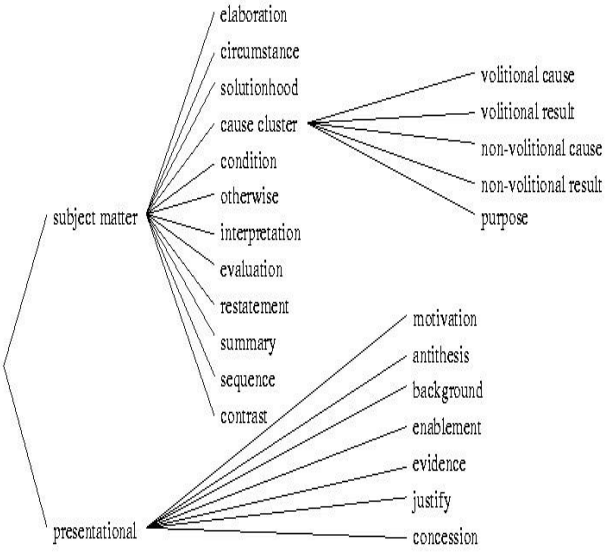

Figure 1. List of RST Relations

3.2 Schemas

Given the Nuclearity and coherence relations that holds between text spans, schema provides the structural constituency arrangements of these spans. There are different types of schema but a certain text span is fitted to a schema using schema applications.

The diagram in Figure 2 shows horizontal lines that represents the text spans, the labelled lines represent the relations between the spans, the nuclei can be distinguished by the vertical line (for multinuclear relations, diagonal lines are used), and the other spans are satellites

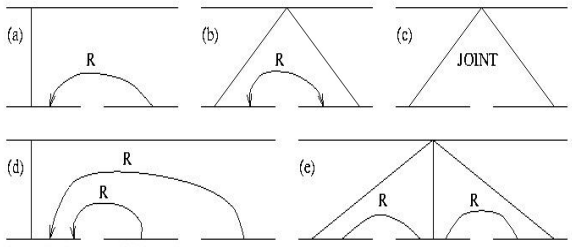

Figure 2. Types of Schemas in RST

For the following example of text spans (Listing 1), Figure 3 is the equivalent RST Tree with corresponding schema applications:

1) *Farmington police had to help control traffic recently*
2) *when hundreds of people lined up to be among the first applying for jobs at the yet-to-open Marriott Hotel.*
3) *The hotel's help-wanted announcement - for 300 openings - was a rare opportunity for many unemployed.*
4) *The people waiting in line carried a message, a refutation, of claims that the jobless could be employed if only they showed enough moxie.*
5) *Every rule has exceptions,*
6) *but the tragic and too-common tableaux of hundreds or even thousands of people snake-lining up for any task with a paycheck illustrates a lack of jobs,*
7) *not laziness.*

Listing 1. Sample Text Spans for RST

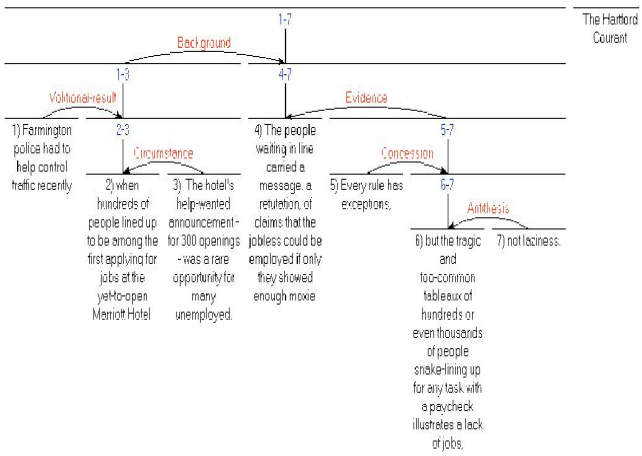

Figure 3. Sample RST Tree given the example in Listing 1.

Given the RST Trees as a result of schema applications, the study exploits the possibility of removing the satellites in the projection that the resulting remaining nuclei is enough to serve as summary to a given paragraph.

4. SUMMARIZER ARCHITECTURAL DESIGN : SUMMERRXT

Initially, the system was designed to contain the distinct phases of Natural Language Understanding, Information Extraction and Natural Language Generation. But as automatic text summarization had been classified within the scope of Natural Language Generation, the architecture design and structure was constructed contained the major components of an NLP system. The major system components were defined more clearly. The sub-modules were rearranged to ensure that that inter-module inputs and outputs are correctly connected. The design was continually refined, even through the coding and debugging stages, to reflect the actual issues experienced during implementation. Additional phases were also added to better achieve the objectives of the system and of the research. In addition to the refinement of the summarization system itself, the proponents also implemented a standalone program that is able to do automatic RST annotation of plain text. The output of the system is a rhetorical structure, represented by an XML file that follows the schema defined in Mick O'Donnell's RSTTool [6]. This module is integrated in the system. The result of this integration in is that the system is now able to accept both XML/RS3 as well as plain TXT files.

Figure 4 outlines the architecture of the automatic text summarizer system. Subsequent sub-sections discuss the details of each of the modules. The domain of documents accepted by the system is technical documents authored by Daniel Marcu. The reason for this domain is that the proponents have read and found Marcu documents consistent in structure, but more importantly, in addressing also the complexity of quantitatively evaluating the system's performance. Evaluation of the system's output, which is the summary, entails comprehension of the original text. Since Marcu documents were reviewed and comprehended as part of the research and related literature, summaries produced by the system can readily be evaluated. In the same manner, since manual tagging of documents are also needed for the gold standard or evaluation, it was decided that it would be a good exercise that tagging is done, alongside reading the Marcu-authored documents, which are relevant to the research.

The system employs the use of discourse structures and keywords which translates the source text into tokens; includes the Parts-of-Speech Tagger module using an existing parts-of-speech tagger; the RST Analyzer module which uses RST in identifying the corresponding discourse structure of the units of sentences; the Keyword Extracting module which identifies and extracts keywords from sentences; the Consolidation module, which is responsible for consolidating the results of the RST Analyzer module and the Keyword Extracting module; the Reduction module, which is responsible for reducing the generated results of the consolidation module; the Referring Expression Analyzer module which is responsible for determining and resolving anaphors in the summary produced; and the Coherence Checker module, which will evaluate the coherence of the generated summary derived from Rhetorical Structure Analyzer and the Sentence Keyword Analyzer components simultaneously.

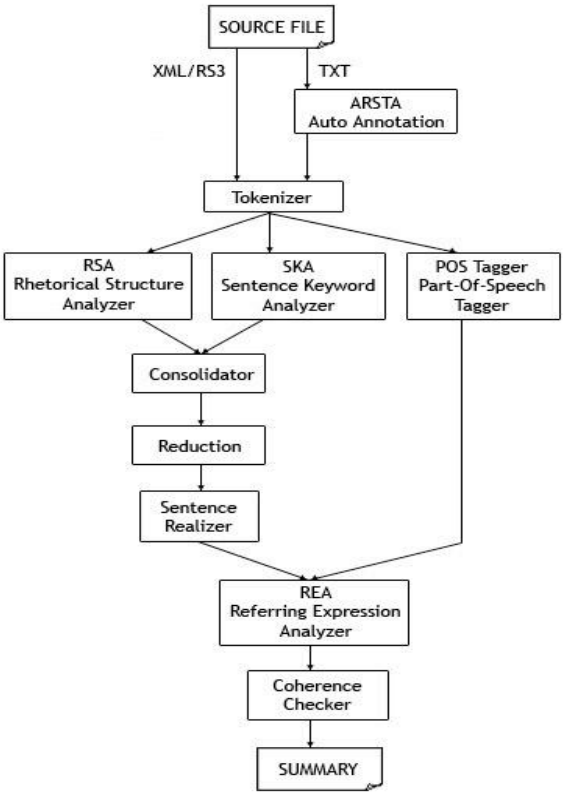

Figure 4. SummerRXT System Architecture

4.1. ATS SKA (Sentence Keyword Analyzer) Component

This module connects the system with KEA [7], an open source keyphrase extractor based on the Naïve Bayes algorithm. The KEW was trained using Marcu-authored document and the various pre-fed documents that comes with KEA (the CSTR Training set).

To help in deciding the number of keywords to be extracted by KEA, the proponents tested the quality of the resulting summary generated. The keyphrases have a 30% role in affecting the summary generated as discussed later in Section 4.5 ATS Reduction Components. The proponents compiled the system with 100, 90, 80, 70, 60, 50, 40, 30, 20, 10 and 5 keywords to be extracted. The source text used for this testing was the D. Marcu conference paper [10]. Each time, the resulting summaries were manually read and analyzed. The resulting analysis was that:

- 5-, 10-, 20- and 30-keyphrase summaries are exactly the same;
- 40- and 50-keyphrase summaries are exactly the same;
- 60-, 70-, 80- and 90-keyphrase summaries are exactly the same; and
- 100-keyphrase is unique.

Having determined the four unique summaries, each summary was compared against each other to find out which amount generated the better summary. There were several differences in

the document, in terms of the inclusion and omission of sentences.

4.2. ATS RSA (Rhetorical Structure Analyzer) Component

This module is responsible for reconstructing the system representation of the RSTrees and later on determining the units important for the summary. The output of the module has been designed to work seamlessly with Mick O'Donnell's RSTTool.

To outline the operation of the RSA Component, consider the text units below in Listing 2 as input:

1. *For the annotation phase, we were unable to collect data for two essay prompts*
2. *because of our annotators' availability.*
3. *This means that we only have inter-annotator agreement statistics on 4 prompts,*
4. *although some data from all six prompts was available for training and testing our models (with the extra two prompts being represented in the training phase of annotation).*

Listing 2. Example Text Units for RSA Processing

The result for the above units is as follows:

2 is a Cause of 1
4 is a Concession of 4
Units 3 and 4 is an Interpretation of Units 1 and 2

Simply put, the main operation of the RSA Component is determining the text units and relationships these units have with each other as well as determining which nodes are the nucleus and which are satellites.

Marcu's instructions outlined in [8] where then applied automatically to determine which units are viable and significant by calculating the score of a certain textual unit in relation to the properties of the whole tree in general. Based on these scores, nodes were promoted from bottom up and will be used later to build the summary.

4.3. ATS Part-Of-Speech Tagger Component

This component connects the system with QTag [9], a probabilistic part-of-speech (POS) tagger.

It is freely distributable and downloadable, and it comes in a JAR file (qtag.jar). It uses two additional files, a lexicon (qtag-eng.lex) and a matrix (qtag-eng.mat) file, which is also included in the download package. QTag is directly executable, so one can use these commands independently in the command line interface. Using the tag method returns an array of the same size as tokens, containing the part-of-speech tags of each corresponding element. When running the ATS program, remember to always include the QTag package in the classpath. The POS Tagger module accepts a source text and returns the POS tag for each corresponding token in the text. It uses QTag, a freely available part-of-speech tagger that comes in a JAR package. The system first tokenizes the text into words.

4.4. ATS Consolidation Components

The Consolidator module is responsible for incorporating the results of the SKA and the RSA modules and storing relevant information in a Vector containing instances of ConsolidatedObject as its elements.

Since the results of RSA will be used as the backbone, the structure of ConsolidatedObject will be containing most elements of an RSA data structure (Noder), which are the following: Noder.id, Noder.unit, Noder. Each key concept and keyword will be matched to all units contained in the RST Vector. Its score, which is a variable in the data structure ConsolidatedObject, is incremented The consolidator module accepts the results of the SKA and the RSA modules and consolidates the results into a variable of type ConsolidatedObject.

4.5. ATS Reduction Components

The Reduction module is responsible for determining which units are significant in creating a summary.

The study involved determining the best ratio to use in combining the RSA and SKA scores to obtain the overall score. Since the results of RSA are to be used as the structure of the summary.

To determine what ratio to use, the study involved studying every summary produced with intervals of 5, meaning summaries were tested from 50-50, 55-45, 60-40 up until 90-10. In other words, manually evaluating and reading through the summary outputs, the "best" ratio was established. And, although subjectively, the test yield the best result with the following ratio: 70% RSA and 30% SKA.

This was decided because after comparing each summary output, it was found out that below this ratio e.g. 60-40, the effect of RSA scores significantly decreases because most of the important units that RSA scores deemed important were removed. And if the ratio goes above 70-30, e.g. 80-20, the effect of SKA scores is not reflected on the output summary. Therefore it was decided that a ratio of 70-30 best maximizes the strengths of both modules. From this theory the reduction formula was developed:

$$\text{Reduction Formula} = 0.70 * \text{RSA Scores} + 0.30 * \text{SKA Scores}$$

Equation 1. Reduction Formula

Another responsibility of this module, aside from determining the most important units, is to determine how many of these units should remain in the summary. This is where the conciseness of the summary is determined. One major issue was the percentage of summarization, or the amount of compression. Because of this, the system provides options to allow the user to specify the percentage of summarization he needs. To determine the default rate of summarization, generated summaries of varying percentages ranging from 10% to 50% were tested. Testing, once again, involves manual reading and subjective evaluation of the outputs mainly by the proponents. The explanation for the range was that anything below 10% is not a summary but resembling an abstract, and anything above 50% is no longer a summary. After analyzing the summaries the proponents agreed to set the default percentage of summarization to 20%. The reduction module accepts the results of the Consolidator module, which is a Vector containing ConsolidatedObject. It reduces the Vector using the reduction algorithm and returns the reduced Vector.

4.6. ATS Sentence Realizer Component

This module is responsible for arranging the important units in order by first determining as to which sentence each unit belongs. This process is not that complicated since the system uses the original text as its reference. After finding out the sentence where the unit occurred, it uses the sentence numbering to arrange the units in ascending order. It is also necessary to enforce structure in the overall summary. Therefore, each section should be properly divided and included into the correct heading. To do this it was necessary to determine the heading each sentence belongs to. Since this information is included in the ConsolidatedObject. Initially, the unit numbers were used as the main sorting variable. However, since textual units may not be whole sentences, sentence numbers replaced unit numbers as the main sorting variable. Using unit numbers for sorting would cause factual and information fragmentation which will surely affect the readability of the resulting text. Arranging the sentences in order is not enough to be able to get a structured text, which is a characteristic of a coherent document. It studies each sentence and uses information which has been collected from the preceding modules. It outputs an organized vector which it then passes to the next module, which is REA.

4.7. ATS REA (Referring Expression Analyzer)

This module is responsible for resolving dangling pronouns in the document. The proponents tested 5 summaries and found out that only 2 out of the 4 pronouns needed to be resolved, which are "they" and "their." In the system's algorithm of anaphora resolution, the first step is to determine the focus of each sentence. The proponents tried to use preexisting syntactic parsers, however, the parsers investigated asked for requirements such as dictionaries, English rule grammar, etc, that the proponents found it more convenient to implement a simpler syntactic parser that would perform only the operations required. This module first uses the results of the POS Tagger Module to determine which word in the sentences are pronouns and which are nouns.

4.8. ATS Coherence Checker

This module is responsible for checking whether the summary is able to achieve coherence, at least the minimum definition of coherence as stated by [11].

Although there is no computationally formal definition for coherence in texts, the criteria set by [11] provided the guidelines of this module in enforcing the coherence in generated texts.

The first definition of coherence is that the text should have a structure. This module checks whether main sections headings were correctly extracted by the Sentence Realizer module. If a structure is missing, it passes back the summary to the Sentence Realizer module to allocate the corresponding structure into the summary. Aside from the structure, the second definition of coherence is that a text document should have unity. There are three ways on how the system can ensure that a text is united:

1. the grouping or sectioning of sentences with similar topics or focus;
2. checking for transition keywords;
3. and resolving anaphors.

The last item had been addressed by REA. The first item, determining or grouping the sentences according topics, was not much of an issue because REA also is responsible for determining the focus of each sentence. These units of similar focus only needed to be grouped together to form a single paragraph. The second item, transitional keywords, are also added to enforce smooth transitions between text spans (phrases or sentences).

4.8. ARSTA (Automatic Rhetorical Structure Theory Annotator)

This module takes the source text, automatically annotates and creates a representation of discourse structure based on RST relations. The module could only automatically annotate a text using only a subset of the RST relations. Table 1 provides the listing of relations handled by the module. The module determines elementary units of text using cue phrases and punctuation and these cue phrases in turn determine what rhetorical relation to tag.

Table 1. Subset of RST Relations handled by the System

Cause	because due to Since
Result	as a result consequently as a consequence hence Thus
Concession	although even though in spite of notwithstanding regardless

5. TESTING AND EVALUATION

The system output was subjected to both manual and automatic evaluation. Two different documents and the corresponding summary outputs from SummerRXT were evaluated alongside the ouputs of Copernic Summarizer and Microsoft Word AutoSummarize. 30 respondents of various backgrounds evaluated the 2 sets of three ouptuts based on the following criteria:

1. The elements of the summary work towards a common goal.
2. The structure of the summary helps facilitate the smooth flow of ideas.
3. The summary is grammatically correct.
4. The summary does not contain unnecessary nor insignificant information.
5. Overall, the summary is an adequate substitute of the original text.

Questions 1 and 2 tackles on coherence and is rated highly if the summary shows sentences that juxtaposed in a logical flow as well as in a logical relation to the whole. Question #3 is on grammaticality and is rated according to correct spelling and grammar which includes, but not limited to, proper punctuation, use of parts of speech and sentence construction. Question #4

deals with informativeness and establishes if the summary maintains important information from the original document as well as if it constructed redundant or insignificant information. Question #5 addresses substitutability of the summary to the original document. A sixth question about comprehension is also provided also to measure if SummerRXT's output summaries are comprehensible although not all respondents answered this survey question and is therefore inconclusive.

The output of SummerRXT garnered the highest averages in all criteria compared to the other two systems' outputs. This is true for both documents involved in the survey as shown in Figure 5 & Figure 6.

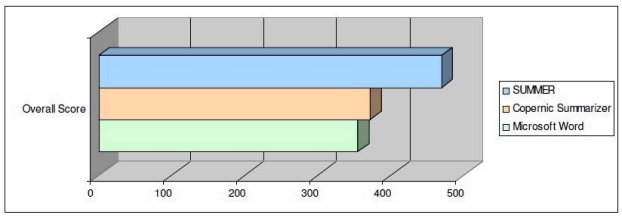

Figure 5. Overall Scores of Summarizers for Set A (1st Document)

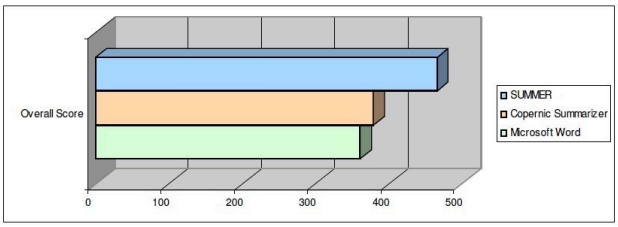

Figure 6. Overall Scores of Summarizers for Set B (2nd Document)

In automatic evaluation, sentence recall was performed by providing and comparing the output with a gold standard. The gold standard were provided manually by one of the researchers and a computer science graduate. The results were inconclusive to establish informativeness of the summary outputs due to the following factors:

– The two reference summaries were made with different levels of competence.
– Each person has his or her own style in creating a summary.
– It is difficult to distinguish between many possible summaries.

5.1. Further Testing on non-Marcu Documents

Further testing was also conducted to the system by subjecting it to non-Marcu-authored documents, i.e., documents not part of the domain. This was just an initial/preliminary test for robustness and/or domain independence.

Two main constraints are considered in this test:

1. The keyword extraction module of summarizer is trained on Marcu-authored documents
2. The summarizer takes Daniel Marcu's composition style into consideration:
   - Header format
   - Use of pronouns
   - Document structuring

Considering these constraints, in theory, given any technical paper, which manifests similar constraints regardless of the author, the summarizer could still generate a summary that is up to standard.

In validating the theory, two papers were evaluated which manifested the similar constraints. [12] presents a discourse parser named D-LTAG, and even compares their work with Marcu's rhetorical parser in [13]. On the other hand, [14] discusses integrating cohesion and coherence for ATS. SUMMERRXT was able to generate summaries out of both papers. Additionally, the group decided to test SUMMERRXT with a paper from a domain unrelated to Computer Science. [15] describes CMEX Mars, a comprehensive set of concept maps describing all aspects of Mars exploration. A summary was likewise generated from this paper.All these papers were automatically tagged by the ARSTA module. Ratherthan give these summaries away for evaluation, the proponents decided to evaluate these summaries themselves, due to the complexity and time requirement of having the evaluators read and analyze all these documents. Although the evaluation is not ideal and not performed through to a survey with ample number of respondents, upon evaluation of the proponents to the summaries produced, the output and the result is still relatively decent: both informativeness and coherence are inherent.

6. CONCLUSION AND RECOMMENDATION

The study focused on combining two approaches in ATS - keyword extraction and nuclearity, with the latter forming the bulk of the work.

The group found that the use of discourse structure proved to be effective in building summaries. Above all, it attends to an essential characteristic of a good summary - informativeness. Another discovery was that keyphrases have a 30% role in affecting the summary generated. Just as Marcu asserted in [10], the study recognized that nuclearity alone was not sufficient in creating summaries of very high quality. Particularly, the summary's coherence - a primary goal in the research, was not addressed. To sort out this issue, a formal and computable definition or specification of coherence had to be made.
The highlight of the research was the formulation of the reduction formula that establishes the recommendable percentage of keyphrases and rhetorical structure units in generating summaries. Moreover, the development of an automatic annotation and tagging of RST components and relationships from textual units removed the laborious manual tagging, as well as, truly allowing fully automated summary generation using RST.

The proponents performed a preliminary test for domain independence. Although the test produced positive results, it still was not a sufficient basis for concluding that SummerRXT is domain independent. Therefore, the proponents suggest further tests to be conducted in this aspect.

The research supposes that the ways of summarizing documents vary for each domain, and that specializing in a specific field

makes the summarizer more efficient in producing summaries under that said field. However, it would be an advantage if SummerRXT could be able to successfully analyze and summarize texts from arbitrary fields, regardless of its structure and form. SummerRXT may also be extended to acquire knowledge autonomously and automatically, without prior development of a subject-specific dictionary by a human expert. Possible areas of improvement may also focus on some modules of SummerRXT, mainly ARSTA and REA. In ARSTA, the following extensions could be made: 1. dealing with the ambiguity of cue phrases; 2. dealing with cue phrases signaling more than 1 RST relation; and 3. moving beyond shallow processing As for REA, it could be extended to include more pronouns in its domain. Further work could also include multi-document summarization. At this day and age wherein information keeps on growing and time is a critical resource, it is important to explore methods of allowing people to access and browse information quickly within a multitude of documents.

7. REFERENCES

[1] Wan, S. (2003) Summarization. [online]. Available: http://www.ics.mq.edu.au/~swan/summarization/ (January 21, 2010)

[2] Hovy, H. & Lin, C.Y. (1998) Automatic Text Summarization and the SUMMARIST system. Proceedings of a Workshop on the Annual Meeting of the ACL. Baltimore, Maryland. [online] Available: http://www.isi.edu/natural-language/projects/SUMMARIST.html (January 21, 2010)

[3] Text Mining or Text Analysis Software: Text Analyst. http://www.megaputer.com/products/ta/index.php3

[4] T. Johnson, S. Thede, A. Vlahov. PARE: An Automatic Text Summarizer. First Midstates Conference for Undergraduate Research in Computer Science and Mathematics, 2003.

[5] Mann, William C. and Sandra A. Thompson (1988) Rhetorical Structure Theory: Toward a functional theory of text organization. *Text, 8* (3): 243-281.

[6] O'Donnell, M. RSTTool—An RST Markup Tool. [online] Available: http://www.wagsoft.com/RSTTool/ (January 20, 2010)

[7] Witten I.H., Paynter G.W., Frank E., Gutwin C. and Nevill-Manning C.G. (1999) KEA: Practical automatic keyphrase extraction. [online]. Available: http://www.nzdl.org/Kea/Nevill-et-al-1999-DL99-poster.pdf (March 18, 2005)

[8] Marcu, D. (1999) Instructions for Manually Annotating the Discourse Structure of Texts. [online] Available: http://www.isi.edu/~marcu/discourse/AnnotationManuals.html (January 20, 2010)

[9] O. Mason, (2003). Qtag 3.1. Department of English, School of Humanities, University of Birmingham, http://web.bham.ac.uk/O.Mason/s oftware/tagger/

[10] Marcu, D. (1998) To build text summaries of high quality, nuclearity is not sufficient. The Working Notes of the AAAI-98 Spring Symposium on Intelligent Text Summarization. [online] Available: http://www.isi.edu/~marcu/papers/summarization-aaai98.ps (February 20, 2010)

[11] Low, C. & Pan, D. (2004) The Write Right Guide. [online] Available: http://www.cdtl.nus.edu.sg/wrg/default.htm (February 7, 2010)

[12] Forbes, K., Joshu, A., Miltsakaki, E., Prasad, R. Sarkar, A. & Webber, B. (2001) D-LTAGSystem - Discourse Parsing with a Lexicalized Tree Adjoining Grammar. [online] Available: http://www.sfu.ca/~anoop/papers/pdf/esslli-2001-final.pdf (February 26, 2010)

[13] Marcu, D. (1997) The rhetorical parsing, summarization, and generation of natural language texts. [online] Available: http://www.isi.edu/~marcu/papers/phd-thesis.ps.gz (February 20, 2010)

[14] Alemany, L.A. & Fort, M.F. (2003) Integrating cohesion and coherence for Automatic Summarization. [online] Available: http://www.coli.uni-sb.de/conf/eacl03-student/Alonso_Fuentes.pdf (February 26, 2010)

[15] Briggs, G., Cañas, A.J., Carff, R., Novak, J., Scargle, J., & Shamma, D.A. (2004) Concept Maps Applied to Mars Exploration Public Outreach. [online] Available:http://infolab.northwestern.edu/infolab/downloads/papers/paper10140.pdf (February 26, 2010)