Semantic Web Project Report on

Autosummarization of Technical Paper using Grammatical Rule Inference

S.Gopinath (12IT64)

S.Vinay Kumar(12IT66)

Under the Guidance of,

Mr. Dinesh Naik

Department of Information Technology, NITK Surathkal

Date of Submission: 8th May, 2015

in partial fulfillment for the award of the degree of

Bachelor of Technology

In

Information Technology At

Department of Information Technology

National Institute of Technology Karnataka, Surathkal

April 2015

Department of Information Technology, NITK Surathkal
Semantic Web Technology Project
End Semester Evaluation Report (April 2015)

Course Code : IT 412

Course Title: Semantic Web Technology

Project Title: *Autosummarization of Technical Paper using Grammatical Rule Inference*

Project Group:

Name of the Student	Register No.	Signature with Date
S.Gopinath	12IT64	
S.Vinay Kumar	12IT66	

Place:

Date: *(Name and Signature of Guide)*

Abstract

Over the years, the amount of information available electronically has grown manifold. There is an increasing demand for automatic methods for text summarization. Domain independent techniques for automatic summarization by paragraph extraction have been proposed in many papers [3, 4]. In this paper, we attempt to summarize a technical paper written in html using grammatical rule inference technique so that it becomes easier for us humans to compare between many technical papers just by reading their summary.

Keywords: *Auto summarization, Grammar Rule Inference Technique, Semantic Web*

Contents

1 **Introduction** 1

1.1 ***Motivation*** 2

2 **Literature Survey** 3

2.1 ***Problem Statement*** 3

3 **Methodology** 4

3.1 ***System Architecture*** 4

3.2 ***Different Components of the Project*** 4

4 **Work Done** 7

5 **Results and Analysis** 8

6 **Conclusion & Future Work** 9

6.1 ***Future Work*** 9

List of Figures

3.1 System Architechture 4
5.1 Input Data-HTML Page-Part 1 8
5.2 Input Data-HTML Page-Part 2 8
5.3 Input Data-HTML Page-Part 3 9
5.4 Output - Summary of the Technical Paper - Part 1 9
5.5 Output - Summary of the Technical Paper - Part 2 10

1 Introduction

Nowadays the Web has proved to be as a rich and extraordinary data source of information, here multiple domains can be accessed and mined. Mining Web data is referred as Web Mining. Some of the objectives of mining web data include finding relevant information discovering new knowledge from web personalized, web synthesis and learning about individual users. Amongst these the most common use is finding relevant information. We simply specify a set of keywords or query as a request or a reference and we get a list of pages, ranked as per similarity of query.

Currently searching web face with one problem that many times outcome is not satisfactory because of irrelevance of the information. Searching the exact information from such a huge repository [2] of unstructured web data is still main area of research interest. One solution to this problem is Semantic Web. The Semantic Web is an extension of current Web in which information is given as well defined meaning, hence enabling computers and people to work with better coordination. By using the existing web semantically new semantically structure can be exploited, and then the results of web mining can be improved, thereby building semantic Web [1].

Automatic summarization is the process of reducing a text document with a computer program in order to create a summary that retains the most important points of the original document. As the problem of information overload has grown, and as the quantity of data has increased, so has interest in automatic summarization. Technologies that can make a coherent summary take into account variables such as length, writing style and syntax

Generally, there are two approaches to automatic summarization: extraction and abstraction. Extractive methods work by selecting a subset of existing words, phrases, or sentences in the original text to form the summary. In contrast, abstractive methods build an internal semantic representation and then use natural language generation techniques to create a summary that is closer to what a human might generate. Such a summary might contain words not explicitly present in the original. Research into abstractive methods is an increasingly important and active research area, however due to complexity constraints, research to date has focused primarily on extractive methods.

1.1 *Motivation*

Current automatic summarizers usually rely on sentence extraction to produce summaries. Human professionals also often reuse the input documents to generate summaries; however, rather than simply extracting sentences and stringing them together, as most current summarizers do, humans often "edit" the extracted sentences in some way so that the resulting summary is concise and coherent.

2 Literature Survey

Currently works have been done in two common approaches to summarisation. The first approach comprises of analysis of text, and rewriting or rephrasing it in a short way. Until today this approach hasn't achieved any substantial results. The second approach, which is similar to algorithm used in this project, tries to extract the key sentences from the text, and then tries to put them together properly. One famous algorithm that implements this approach is TextRank [5].

TextRank is a general purpose graph-based ranking algorithm for NLP. Essentially, it runs PageRank on a graph specially designed for a particular NLP task. For keyphrase extraction, it builds a graph using some set of text units as vertices. Edges are based on some measure of semantic or lexical similarity between the text unit vertices. Unlike PageRank, the edges are typically undirected and can be weighted to reflect a degree of similarity. Once the graph is constructed, it is used to form a stochastic matrix, combined with a damping factor (as in the "random surfer model"), and the ranking over vertices is obtained by finding the eigenvector corresponding to eigenvalue 1 (i.e., the stationary distribution of the random walk on the graph).

There also have been works where one can perform a robust decomposition of the text of an article into individual segments that correspond to basic natural parts, such as title, section title, text, etc. This was aimed at automation of the task of building an Ontology of documents. We combine these two approaches for summarisation of a technical papers.

2.1 *Problem Statement*

"To develop a Travel Route Recommendation System for Indian cities, incorporating user friendly interaction through natural speech dialogues, and natural language processing techniques"

3 Methodology

3.1 *System Architecture*

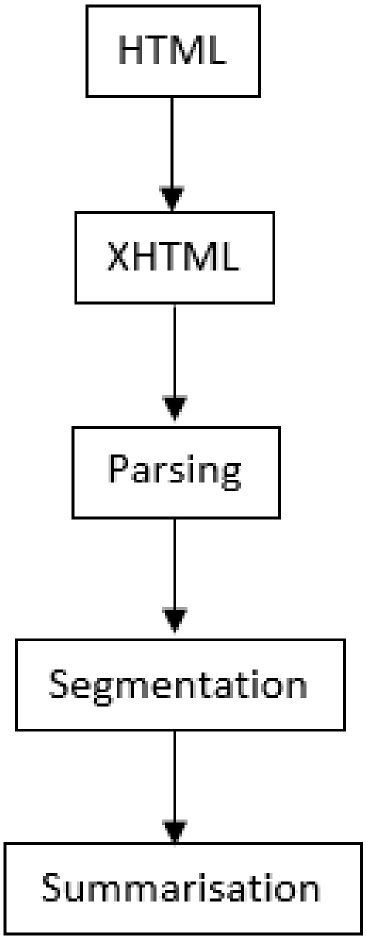


Figure 3.1: System Architechture

3.2 *Different Components of the Project*

1. Preprocessing:

   The extraction of information from web pages is the subject matter of the Wrapper Induction work, although their purpose is not the inference of an ontology, but to enable a database like use of that information. In particular, HTML aware wrapper induction approaches bear similarities to the first step of the method developed In particular, they share the preprocessing of HTML pages with the objective of cleaning frequent syntax errors and transformation into valid XML syntax (using the XHTML format) to ease further processing.

2. Summarization:

   Intersection function: This function receives two sentences, and returns a score for the intersection between them. We just split each sentence into words/tokens, count how many common tokens we have, and then we normalize the result with the average length of the two sentences.

   The sentences dictionary: This part is actually the Heart of the algorithm. It receives our text as input, and calculates a score for each sentence. The calculations is composed of two steps: In the first step we split the text into sentences, and store the intersection value between each two sentences in a matrix (two-dimensional array). So values[0][2] will hold the intersection score between sentence 1 and sentence 3.
   In fact, we just converted our text into a fully-connected weighted graph! Each sentence is a node in the graph and the two-dimensional array holds the weight of each edge!
   In the second step we calculate an individual score for each sentence and store it in a key-value dictionary, where the sentence itself is the key and the value is the total score. We do that just by summing up all its intersections with the other sentences in the text (not including itself).
   We calculates the score for each node in our graph. We simply do that by summing all the edges that are connected to the node. Building the summary: Obviously, the final step of our algorithm is generating the final summary. We do that by splitting our text intoparagraphs, and then we choose the best sentence from each paragraph according to our sentences dictionary. The Idea here is that every paragraph in the text represents some logical subset of our graph, so we just pick the most valuable node from each subset!

Why this works: There are two main reasons why this algorithm works: The first reason is that a paragraph is a logical atomic unit of the text. In simple words there is probably a very good reason why the author decided to split his text that way. The second reason is that if two sentences have a good intersection, they probably holds the same information.

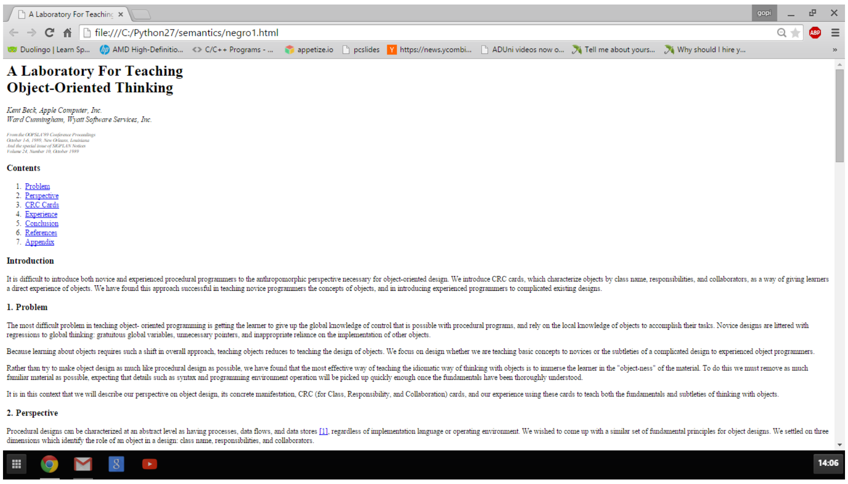
So if one sentence has a good intersection with many other sentences, it probably holds some information from each one of them- or in other words, this is probably a key sentence in our text.

4 Work Done

A technical paper in HTML is taken as the input. The HTML code may be having inconsistencies which may lead to problems. Hence the HTML document is standardized by converting it into a XHTML document. This makes the document structured and it get easier for extraction of segments. After converting the HTML document into XHTML we segment the document and extract paragraphs with the help of ¡p¿¡/p¿ tags that define a paragraph. Once we get the segments (paragraphs), we use our summarization algorithm to summarize each of these paragraphs. We combine the summaries of each paragraphs and create a new document containing the summaries.

5 Results and Analysis

A sample input HTML paper that has been taken as input is shown in the Figure 5.1, Figure 5.2 and Figure 5.3.



A Laboratory For Teaching Object-Oriented Thinking

Kent Beck, Apple Computer, Inc.
Ward Cunningham, Wyatt Software Services, Inc.

From the OOPSLA'89 Conference Proceedings
October 1-6, 1989, New Orleans, Louisiana
And the special issue of SIGPLAN Notices
Volume 24, Number 10, October 1989

Contents

1. Problem
2. Perspective
3. CRC Cards
4. Experience
5. Conclusion
6. References
7. Appendix

Introduction

It is difficult to introduce both novice and experienced procedural programmers to the anthropomorphic perspective necessary for object-oriented design. We introduce CRC cards, which characterize objects by class name, responsibilities, and collaborators, as a way of giving learners a direct experience of objects. We have found this approach successful in teaching novice programmers the concepts of objects, and in introducing experienced programmers to complicated existing designs.

1. Problem

The most difficult problem in teaching object- oriented programming is getting the learner to give up the global knowledge of control that is possible with procedural programs, and rely on the local knowledge of objects to accomplish their tasks. Novice designs are littered with regressions to global thinking: gratuitous global variables, unnecessary pointers, and inappropriate reliance on the implementation of other objects.

Because learning about objects requires such a shift in overall approach, teaching objects reduces to teaching the design of objects. We focus on design whether we are teaching basic concepts to novices or the subtleties of a complicated design to experienced object programmers.

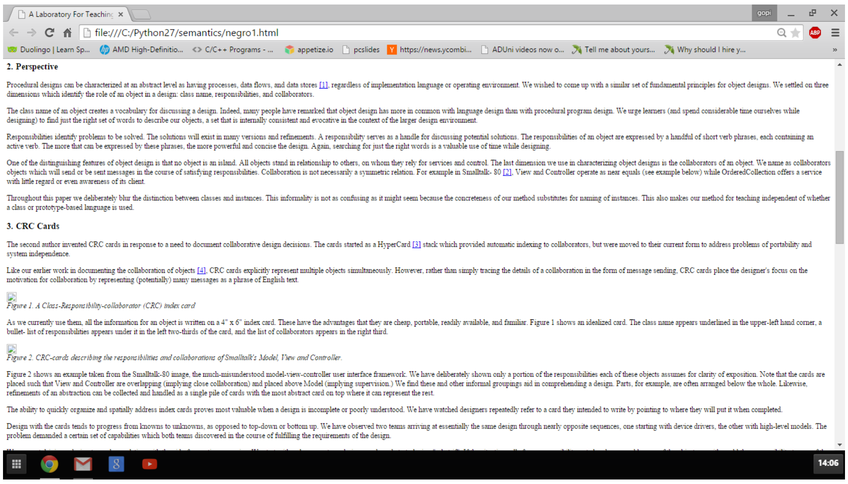
Rather than try to make object design as much like procedural design as possible, we have found that the most effective way of teaching the idiomatic way of thinking with objects is to immerse the learner in the "object-ness" of the material. To do this we must remove as much familiar material as possible, expecting that details such as syntax and programming environment operation will be picked up quickly enough once the fundamentals have been thoroughly understood.

It is in this context that we will describe our perspective on object design, its concrete manifestation, CRC (for Class, Responsibility, and Collaboration) cards, and our experience using these cards to teach both the fundamentals and subtleties of thinking with objects.

2. Perspective

Procedural designs can be characterized at an abstract level as having processes, data flows, and data stores [1], regardless of implementation language or operating environment. We wished to come up with a similar set of fundamental principles for object designs. We settled on three dimensions which identify the role of an object in a design: class name, responsibilities, and collaborators.

Figure 5.1: Input Data-HTML Page-Part 1



2. Perspective

Procedural designs can be characterized at an abstract level as having processes, data flows, and data stores [1], regardless of implementation language or operating environment. We wished to come up with a similar set of fundamental principles for object designs. We settled on three dimensions which identify the role of an object in a design: class name, responsibilities, and collaborators.

The class name of an object creates a vocabulary for discussing a design. Indeed, many people have remarked that object design has more in common with language design than with procedural program design. We urge learners (and spend considerable time ourselves while designing) to find just the right set of words to describe our objects, a set that is internally consistent and evocative in the context of the larger design environment.

Responsibilities identify problems to be solved. The solutions will exist in many versions and refinements. A responsibility serves as a handle for discussing potential solutions. The responsibilities of an object are expressed by a handful of short verb phrases, each containing an active verb. The more that can be expressed by these phrases, the more powerful and concise the design. Again, searching for just the right words is a valuable use of time while designing.

One of the distinguishing features of object design is that no object is an island. All objects stand in relationship to others, on whom they rely for services and control. The last dimension we use in characterizing object designs is the collaborators of an object. We name as collaborators objects which will send or be sent messages in the course of satisfying responsibilities. Collaboration is not necessarily a symmetric relation. For example in Smalltalk- 80 [2], View and Controller operate as near equals (see example below) while OrderedCollection offers a service with little regard or even awareness of its client.

Throughout this paper we deliberately blur the distinction between classes and instances. This informality is not as confusing as it might seem because the concreteness of our method substitutes for naming of instances. This also makes our method for teaching independent of whether a class or prototype-based language is used.

3. CRC Cards

The second author invented CRC cards in response to a need to document collaborative design decisions. The cards started as a HyperCard [3] stack which provided automatic indexing to collaborators, but were moved to their current form to address problems of portability and system independence.

Like our earlier work in documenting the collaboration of objects [4], CRC cards explicitly represent multiple objects simultaneously. However, rather than simply tracing the details of a collaboration in the form of message sending, CRC cards place the designer's focus on the motivation for collaboration by representing (potentially) many messages as a phrase of English text.

Figure 1. A Class-Responsibility-collaborator (CRC) index card

As we currently use them, all the information for an object is written on a 4" x 6" index card. These have the advantages that they are cheap, portable, readily available, and familiar. Figure 1 shows an idealized card. The class name appears underlined in the upper-left hand corner, a bullet- list of responsibilities appears under it in the left two-thirds of the card, and the list of collaborators appears in the right third.

Figure 2. CRC-cards describing the responsibilities and collaborations of Smalltalk's Model, View and Controller.

Figure 2 shows an example taken from the Smalltalk-80 image, the much-misunderstood model-view-controller user interface framework. We have deliberately shown only a portion of the responsibilities each of these objects assumes for clarity of exposition. Note that the cards are placed such that View and Controller are overlapping (implying close collaboration) and placed above Model (implying supervision.) We find these and other informal groupings aid in comprehending a design. Parts, for example, are often arranged below the whole. Likewise, refinements of an abstraction can be collected and handled as a single pile of cards with the most abstract card on top where it can represent the rest.

The ability to quickly organize and spatially address index cards proves most valuable when a design is incomplete or poorly understood. We have watched designers repeatedly refer to a card they intended to write by pointing to where they will put it when completed.

Design with the cards tends to progress from knowns to unknowns, as opposed to top-down or bottom up. We have observed two teams arriving at essentially the same design through nearly opposite sequences, one starting with device drivers, the other with high-level models. The problem demanded a certain set of capabilities which both teams discovered in the course of fulfilling the requirements of the design.

Figure 5.2: Input Data-HTML Page-Part 2

The output, i.e. the summary of the given HTML is shown Figure 5.4 and Figure 5.5.

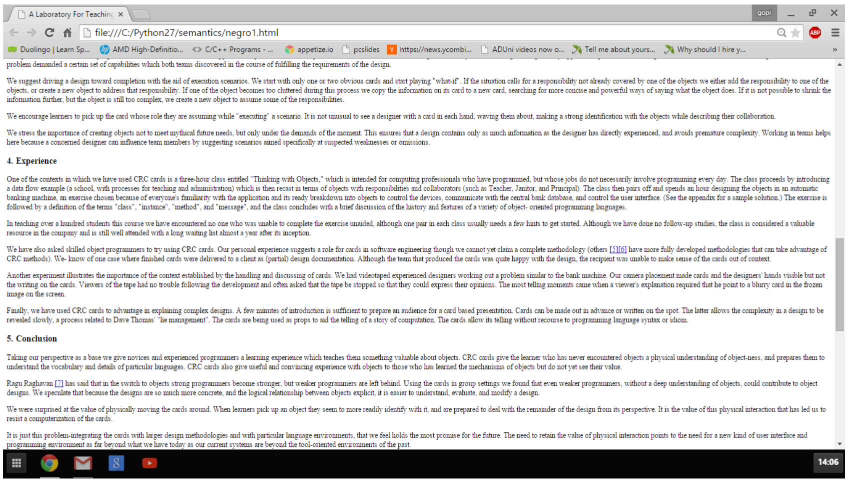
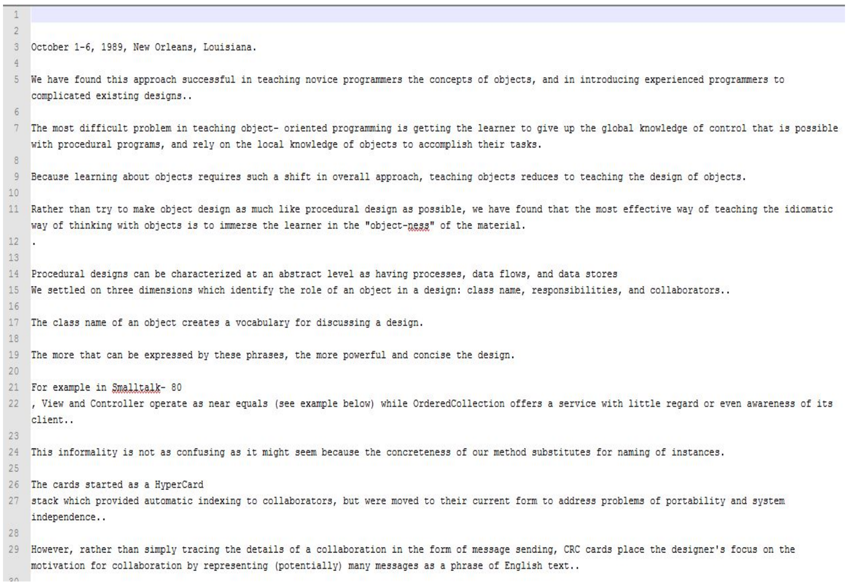


Figure 5.3: Input Data-HTML Page-Part 3



October 1-6, 1989, New Orleans, Louisiana.

We have found this approach successful in teaching novice programmers the concepts of objects, and in introducing experienced programmers to complicated existing designs..

The most difficult problem in teaching object- oriented programming is getting the learner to give up the global knowledge of control that is possible with procedural programs, and rely on the local knowledge of objects to accomplish their tasks.

Because learning about objects requires such a shift in overall approach, teaching objects reduces to teaching the design of objects.

Rather than try to make object design as much like procedural design as possible, we have found that the most effective way of teaching the idiomatic way of thinking with objects is to immerse the learner in the "object-ness" of the material.
.

Procedural designs can be characterized at an abstract level as having processes, data flows, and data stores
We settled on three dimensions which identify the role of an object in a design: class name, responsibilities, and collaborators..

The class name of an object creates a vocabulary for discussing a design.

The more that can be expressed by these phrases, the more powerful and concise the design.

For example in Smalltalk- 80
, View and Controller operate as near equals (see example below) while OrderedCollection offers a service with little regard or even awareness of its client..

This informality is not as confusing as it might seem because the concreteness of our method substitutes for naming of instances.

The cards started as a HyperCard
stack which provided automatic indexing to collaborators, but were moved to their current form to address problems of portability and system independence..

However, rather than simply tracing the details of a collaboration in the form of message sending, CRC cards place the designer's focus on the motivation for collaboration by representing (potentially) many messages as a phrase of English text..

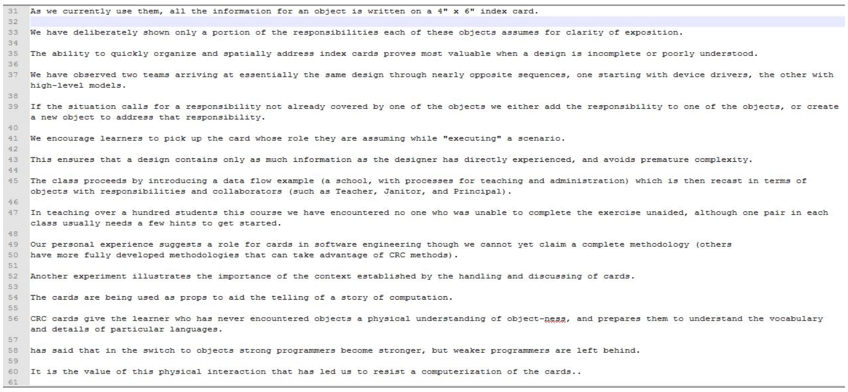
Figure 5.4: Output - Summary of the Technical Paper - Part 1

6 Conclusion & Future Work

An application has been created using which a user can get the summary of any technical paper written in HTML so that it becomes easy for the user to compare between many such papers for similarities.

6.1 *Future Work*

Future work includes functionalities that can compare the summaries of any two technical papers automatically and tell the user whether those two papers are similar or not.



```
As we currently use them, all the information for an object is written on a 4" x 6" index card.

We have deliberately shown only a portion of the responsibilities each of these objects assumes for clarity of exposition.

The ability to quickly organize and spatially address index cards proves most valuable when a design is incomplete or poorly understood.

We have observed two teams arriving at essentially the same design through nearly opposite sequences, one starting with device drivers, the other with
high-level models.

If the situation calls for a responsibility not already covered by one of the objects we either add the responsibility to one of the objects, or create
a new object to address that responsibility.

We encourage learners to pick up the card whose role they are assuming while "executing" a scenario.

This ensures that a design contains only as much information as the designer has directly experienced, and avoids premature complexity.

The class proceeds by introducing a data flow example (a school, with processes for teaching and administration) which is then recast in terms of
objects with responsibilities and collaborators (such as Teacher, Janitor, and Principal).

In teaching over a hundred students this course we have encountered no one who was unable to complete the exercise unaided, although one pair in each
class usually needs a few hints to get started.

Our personal experience suggests a role for cards in software engineering though we cannot yet claim a complete methodology (others
have more fully developed methodologies that can take advantage of CRC methods).

Another experiment illustrates the importance of the context established by the handling and discussing of cards.

The cards are being used as props to aid the telling of a story of computation.

CRC cards give the learner who has never encountered objects a physical understanding of object-ness, and prepares them to understand the vocabulary
and details of particular languages.

has said that in the switch to objects strong programmers become stronger, but weaker programmers are left behind.

It is the value of this physical interaction that has led us to resist a computerization of the cards..
```

Figure 5.5: Output - Summary of the Technical Paper - Part 2

References

[1] Dana Angluin. Inference of reversible languages. *Journal of the ACM (JACM)*, 29(3):741–765, 1982.

[2] Valter Crescenzi, Giansalvatore Mecca, Paolo Merialdo, et al. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, volume 1, pages 109–118, 2001.

[3] Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. Multi-document summarization by sentence extraction. In *Proceedings of the 2000 NAACL-ANLPWorkshop on Automatic summarization-Volume 4*, pages 40–48. Association for Computational Linguistics, 2000.

[4] Mandar Mitray, Amit Singhalz, and Chris Buckleyyy. Automatic text summarization by paragraph extraction. *Compare*, 22215(22215):26, 1997.

[5] Wikipedia. Automatic summarization — wikipedia, the free encyclopedia, 2015. [Online; accessed 8-May-2015].