AN ABSTRACT OF THE THESIS OF

Amirali Saeedi for the degree of Master of Science in

Industrial Engineering presented on May 25, 2010.

Title: A Computer-Assisted Qualitative Data Analysis Framework for the Engineering Management Domain

Abstract approved: ———————————————————————————

Toni L. Doolen

Qualitative data analysis (QDA) is a time consuming and, potentially unreliable research activity. In qualitative research, a number of tasks must be repeated for every new research case, even if each case is closely related or is in the same area of study.

Existing QDA applications provide users with a variety of tools and features that assist researchers in manipulating qualitative data. There is a great advantage in using these functions over completing these tasks manually. However, available QDA tools are not really more than a digital paper and pencil. In other words, existing tools are not equipped with any sort of automatic processing features.

A computer assisted framework was developed to help researchers in conducting qualitative data analysis. This framework leveraged the GATE platform, along with Natural Language Processing and Knowledge Extraction, to develop an automatic text annotation and summarization system. A performance model, developed from the literature on lean manufacturing implementation strategies was converted to an ontology. A lexicon database for lean implementation practices was also developed. A dataset from a previous research study focusing on lean implementation practices was used to conduct

this development and testing. A number of different summarization techniques were developed and tested. A customized sensitivity analysis method was developed and used to systematically perform summarization algorithms comparisons. For the best summarization algorithm, an average F-score of 0.6567 was recorded. This F-score was based on a recall of 0.85 and a precision of 0.55, demonstrating the feasibility of automatic processing on an unstructured qualitative dataset.

A Computer-Assisted Qualitative Data Analysis Framework for the Engineering Management Domain

by

Amirali Saeedi

A THESIS

submitted to

Oregon State University

in partial fulfillment of

the requirements for the

degree of

Master of Science

Presented May 25, 2010

Commencement June 2010

Master of Science thesis of Amirali Saeedi presented on May 25, 2010.

APPROVED:

Major Professor, representing Industrial Engineering

Head of the School of Mechanical, Industrial, and Manufacturing Engineering

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Amirali Saeedi, Author

ACKNOWLEDGEMENTS

I would like to specially thank my advisor, Dr. Toni L. Doolen, for guiding me through this research with excellent patience, and for the given encouragements.

I would also like to thank my committee members Dr. David Kim and Dr. David Porter for their suggestions, useful discussions, and reviewing this work.

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

Figure Page

LIST OF FIGURES (Continued)

Figure Page

LIST OF TABLES

Table Page

DEDICATION

To my beloved parents, and to my dearest Sarah.

A Computer-Assisted Qualitative Data Analysis Framework for the Engineering Management Domain

Chapter 1 - Introduction

While there is considerable literature devoted to the process of interview data coding and qualitative data analysis, there is no customized framework for qualitative studies in the area of engineering management. This thesis will develop and test a framework that can be used to assist researchers in performing qualitative data analysis. The datasets used for this research were adopted from a study related to engineering management.

1.1 Motivation

Attempts to populate knowledge by machine have gained some success. One of the major barriers in creating an automatic system to populate knowledge bases is the need for Natural Language Processing (NLP). Although advancements have been made in this area, NLP systems still cannot understand text as well as humans do.

Qualitative data analysis (QDA), and more specifically the coding process of QDA, is a time consuming and potentially unreliable research activity. These time consuming processes must be repeated for every new research case, even if each case is closely related or is in the same area of study.

This research uses NLP techniques to demonstrate that the knowledge created in previous research not only increases the precision of new discovered models, but can also help expedite the process of knowledge extraction and collection in new studies. This research focused on the use of these NLP techniques to propose a customized framework

for QDA. The framework was tested on a dataset focused on lean implementation practices.

1.2 Contribution

The subject of this thesis embraces a variety of topics from different areas of study such as engineering management, linguistics, statistics, and computer science. A vast amount of relevant literature exists in each of these areas. In this thesis, a selection of the most relevant literature streams is summarized to provide a sufficient level of understanding on each of the related areas. Although the focus of this thesis is not on lean manufacturing itself or the implementation of lean practices, an overview of the general lean literature is provided to enable an understanding of the challenges of research on the management of lean practices and enough explanation to understand the context of the dataset used for this study.

The literature about lean manufacturing principles (Womack et al., 1990; Krafcik, 1988; Alavi, 2003), existing practices (Holden, 2003), and implementation techniques (Filson & Lewis, 2000; Boyer & Sovilla, 2003) is extensive. A lean implementation performance model was created to analyze a dataset collected during a research project that was part of an NSF funded research study, led by two principal investigators, Doolen and Hacker (2004). The research model was developed based on previous studies and existing organizational performance models. The research model includes different factors e.g. External Factors, Internal Context, and Transformation Mechanisms. These factors incorporated a comprehensive set of management practices that were believed to be essential to a lean implementation. The research was conducted in two phases. In the first phase, a survey was developed and administrated to a number of electronic manufacturers in the states of Oregon, Washington, and California. In the second phase, more detailed on-site data collection was performed at these electronic manufacturers.

The second phase used employee interviews and shop floor observations to investigate the detailed management practices of three manufacturers.

The literature on QDA and coding procedures is also very well-developed and rich. A vast amount of research has been undertaken to classify and improve QDA (Seidel, 1998; Taylor-powell & Renner, 2003). A major part of the QDA literature is devoted to the process of coding as one the most challenging steps of the analysis (Gorden, 1992; Crittenden and Hill, 1971; Weston et al. 2001). Although much of the literature is focused on traditional methods of coding, a portion of the literature are studies in which QDA software has been used (Barry, 1998; Weitzman & Miles, 1995).

The literature on qualitative research makes a case for the need for using computer software to increase the performance and quality of research (Tesch, 1990). Currently, a number of applications are available in the market to help researchers organize, code, and complete searches within research documents. A few more advanced applications are able to manipulate a variety of data sources including text, pictures, audio, and video (Barry, 1998).

The literature, however, lacks research on the use of any form of intelligent automation for QDA. Existing advances in knowledge extraction with natural language processing (NLP) have the potential to be extremely helpful. The literature on QDA emphasizes the need for automation processes such as automatic text summarization (Jing, 2000; Amini et al., 2005), automatic coding (Shim at al., 1998; Raymond, 1992), automatic text categorizing (Stamatatos et al., 2000; Joachims et al., 1998), question answering systems (Ittycheriah et al., 2001; Ravichandran & Hovy, 20020), and etc.

Text mining and machine learning techniques can potentially be used to help researchers by taking care of the more repetitive and time consuming elements of the coding process. Application of automatic knowledge extraction has the potential to save time and effort. Although using computer software requires training and preparation,

computer software can increase the consistency of coding and subsequently decrease variance occurring as a result of human limitations, particularly fatigue.

1.3 Research Objectives

The purpose of this thesis is to explore whether a computer system can extract information from surveys and interview transcripts and further be used to answer research questions based on the extracted information. A dataset obtained from interviews and observations conducted at three different electronic manufacturers engaged in different phases of a lean manufacturing implementation was used for this study. More formally, this thesis investigated the feasibility of analyzing various transcripts of free-form text using machine, and further investigated if the analyses could be used to study lean implementation management practices.

To conduct the study, the overall process was further broken down into three sub steps. It is believed that by achieving all of these objectives and then putting them together, the overall objectives for the research can be achieved. These sub steps are summarized in Table 1.1.

Table 1.1: Research Objectives

1. Develop a lexicon database system for the engineering management domain
2. Use existing NLP techniques and the developed lexicon system, to perform automatic text annotation on documents related to engineering management
3. Design an automatic text summarization system to help the researchers in Qualitative Data Analysis

The first step initiates the design of an automated platform. In this study, a lexicon system is defined as a set of files containing lists of similar words (Bransford-Koons,

2005). The lexicon system was based on the Gazetteer module of General Architecture for Text Engineering (GATE) platform and was developed as a complementary part of the entire system. Details of the system and the development process are provided in the methodology section of this thesis.

Classification of new data, as well as existing information, is an important step in preparation of data for processing (Keim, 2002). To capture knowledge existing in text, the software should be able to detect a sense of the content. Unfortunately, at this stage of technology development, computer software cannot understand the content of the text. However, software can recognize word phrases and can linguistically analyze word phrases (Tan, 1999). Therefore, having an organized word list of dialogue related to lean manufacturing implementation practices can enable software to "sense" the content of relevant text.

The second objective helps to go one step further in the process of analyzing text. This step investigates the possibility of recognition and annotation of the important parts of text using computer software. This process is called automatic text annotation in the natural language processing literature (Nagao & Hasida, 1998; Erdmann et al., 2000). The lexicon system developed in step one is the basis for the automatic annotator unit. This unit uses simple distance patterns to detect sensitive and important parts of the text.

The outcome of the annotator unit can be used to build a summarization tool. This tool can help researchers shrink a set of documents down to automatically generated summaries containing only key portions of text. Descriptions for various text summarization techniques can be found in the literature (Amini et al., 2005; Mitra et al., 2002). Although accuracy in automatic summarization is challenging, the vast amount of data in qualitative analysis make even moderately accurate summarization tools useful and time saving (Taylor-Powell & Renner, 2003). The proposed platform allows researchers to identify existing relations and connections in the text. For this purpose, the developed categories from the first step along with the idea of word-distance in the

second step is used to index a corpus (a collection of text documents) and to discover potential conceptual relationships. For example, suppose we are trying to capture the importance of “employee training” as a preparation step for lean manufacturing implementation (Smeds, 1994). For the defined example, from a text mining perspective, we are interested in identifying instances of occurrence of these two words and any possible synonyms that appear relatively close to each other in the text. These two words should be recognized in a phrase with a minimum length of two words (at least having both of them in the phrase, i.e. employee training or training employees). Theoretically, there is no maximum limit for the number of words in the phrase. However, we were able to define practical limits to confine the length of phrases. Details of the deployed algorithm are provided in later chapters.

Ultimately the developed platform allows researchers to run structured questions on the text and test different hypotheses, based on the concepts under study. The application of this platform occurs when a researcher has developed a conceptual model (Seidel, 1998) about a system and would like to validate relationships hypothesized in the model using real data. As an example, suppose a researcher is reviewing interview transcripts. The researcher would like to know if evidence from the transcripts support the role of employee training on the success of a lean manufacturing implementation. At this time, this platform can search for such patterns and identify instances in the text that might support or refute the hypothesized relationships.

The framework created for this project was based on a publicly available system and incorporates various capabilities to accomplish these goals. Completing the steps from Table 1.1, resulted in a powerful framework for conducting qualitative research. The usability of this platform and approach were tested using an actual dataset. This framework can bring a new level of intelligence and automation into the conventional methods of QDA. It can save time and effort in the process and increase the reliability and repeatability of the analysis and subsequent conclusions drawn from the analysis.

1.4 Approach

In the current state of Machine Learning research, developing general purpose knowledge extraction systems is a challenge, and content-aware text processing tools are being actively investigated. However, if a study is restricted to a limited domain of knowledge, current capabilities of NLP can help develop customized knowledge extraction systems. Interview transcripts and employee surveys related to lean implementation practices were used as the test data. Once the expected knowledge is extracted from the qualitative data, querying and reasoning processes were used to test research hypotheses.

The data used to test the framework were obtained from a research project conducted in three organizations involved in the assembly of printed circuit boards. The goal of the research project was to investigate the management practices associated with the implementation of lean principles and tools to improve organizational performance. More specifically, the study was performed to evaluate the role of management practices on the lean implementation within these companies. The dataset included field notes, observations, and interviews. All data were transcribed into an electronic format. The available dataset provided a context-rich dataset needed to study the proposed method. Results and findings from previous research were used to validate and measure the performance of the framework.

The lean implementation performance model (Doolen & Worley, 2004) used to analyze the data, is presented in the literature review chapter of this thesis. The model contains four components, each of which is defined with a set of variables taken from the literature on strategic planning, process improvement, and performance measurement. Four comprehensive frameworks from the literature were examined to create the model. The performance model and defined variable sets were created through the integration of these frameworks. A more in-depth discussion on these frameworks are provided in the literature review section.

The overall outline of the framework is designed on the basis of the classic QDA methodology. Thus, the major contribution of this research is the development of tools to enable the coding process of QDA. For this thesis, a variety of tools and concepts from different areas of Natural Language Processing, knowledge extraction, and text mining were examined and deployed into the framework. The framework was built using a General Architecture for Text Engineering (GATE) platform developed at the University of Sheffield. This publicly available platform has been used successfully in a large number of projects (http://www.gate.co.uk/). GATE provides a wide range of tools for many different tasks. Before using the GATE platform, the system had to be configured and necessary components had to be developed or customized.

1.5 Findings

In this study a collection of words and phrases that appear frequently in a set of research documents, obtained as part of a study of lean implementation practices, was identified. As a result, this study produced a lexicon database that covers major concepts related to a lean manufacturing implementation. The results confirmed that the process of developing the lexicon database directly affects the quality of computer-generated annotations. There was some evidence that the developed lexicon might be more applicable to documents resulting from interviews with managers than from line employees.

Since computer software cannot understand the content of text, computer generated annotations are still not very accurate. However, this study showed that researchers can save time by starting with a preliminary coded version of documents, instead of beginning from scratch. The time difference becomes obvious once the time needed to code a dataset manually is compared to the time needed for development of a lexicon database for the study domain. When a segment of a text is connected to a specific concept in the study model, the researcher can more efficiently analyze that segment.

Performance measures obtained for the summarizer system showed the importance of employing a reliable summarization algorithm. Evidence was found that very small changes in the summarization algorithm and its parameters can significantly impact the quality of the automated summaries generated by the tool.

1.6 Conclusions

Current qualitative data analysis processes can be improved in different ways. Existing computer software packages offer many tools that make conducting qualitative analysis of datasets easier and faster. Researchers can take advantage of available computer software packages for storing and organizing, searching, editing, and coding documents. Moreover, achievements in Natural Language Processing (NLP) and knowledge extraction domains can replace some of the manual processes, which are part of qualitative data analysis. The framework presented in this thesis demonstrated the use of automatic annotation and automatic summarization systems on an engineering management qualitative research dataset. The presented framework can process large datasets in a short duration, and therefore, can result in savings of both time and human effort.

Chapter 2 - Literature Review

In this chapter, a comprehensive literature review of various domains related to this research study is presented. First, lean manufacturing and management practices related to lean implementations are reviewed. A number of related topics from the computer science domain are reviewed next. Computer science research offers tools and techniques for automating tasks that are usually conducted manually. A summary of natural language processing and knowledge discovery is presented. Qualitative data analysis (QDA) is introduced and a summary of available QDA software is provided. Later, classical research procedures for qualitative studies are described, and relevant work completed in other domains are introduced.

2.1 Lean Manufacturing

In 1950, Eiji Toyoda and Taiichi Ohno created the Toyota Production System (TPS) (Womack et al., 1990). This system was based on Ford's standardized assembly line, but TPS added the team concept. In TPS, workers were trained not only on shop floor tasks, but in a variety of other skills. They were taught to be responsible for standardizing their work and for continuous improvement (Krafcik, 1998). Krafcik used the term lean production system in 1988 during a comprehensive research on automobile assembly plants. Womack et al. (1990) further elaborated on the Toyota Production System in the book "The Machine that Changed the World".

Lean manufacturing is defined as the systematic removal of waste by all members of the organization from all areas of the value stream (Womack & Jones, 1996). Womack and Jones (1996) define value stream as all the activities that contribute to the transformation of a product from raw material to finished product including design, order taking, and physical manufacturing.

Numerous advantages of using lean manufacturing are described in the literature. In general, lean manufacturing reduces the cost of doing business by keeping the inventory at the minimum level and by providing reduced cycle times (Allen, 2000). Lean manufacturing benefits documented in the literature include reduced build time, increased quality, increased customer satisfaction, reduced inventory, increased safety in the workplace, and reduced setup time.

Every organization will have a unique implementation of lean practices. Thus, there is no general formulation for lean implementation (Bamber and Dale, 2000). A systematic approach for introducing lean practices into the organization is important to the success of the implementation. The process of implementing lean practices is very important, and the implementation methodology that is used affects the outcome of the process (Allen, 2000). The variables that will affect the success of a lean manufacturing implementation are important to understand. It is also important to realize how the organization may be affected by a lean manufacturing implementation.

To study the relationship between lean implementation and organizational outcomes, a performance model was proposed. The performance model captured for this study was developed for a research project that generated the data used for this study (Doolen & Worley, 2004). The overall research project was conducted in two phases. In the first phase, a survey was created to measure the degree to which electronic manufacturers had adopted lean manufacturing practices. A comprehensive set of lean manufacturing practices were identified from the literature, and survey items were developed from this list. The survey was mailed to three different types of manufacturers, including semiconductor and wafer manufacturers, electronic equipment manufacturers, and printed circuit board manufacturers and assembly operations. All manufacturers were located in the states of Washington, Oregon, and California.

In the second phase, data collection concentrated on identifying the specific issues faced by organizations engaged in implementing lean manufacturing tools and

techniques. Data were collected from field observations and interviews in three different electronic manufacturing organizations. Field notes and interviews were transcribed into an electronic format and then coded linking data from the notes to the proposed performance model developed by the researchers.

The development of the research model was informed by previous research in the areas of strategic planning, process improvement, and performance measurement. Figure 2.1 displays the performance model created for the research project. This model incorporates both components and the structural relationship between the components.

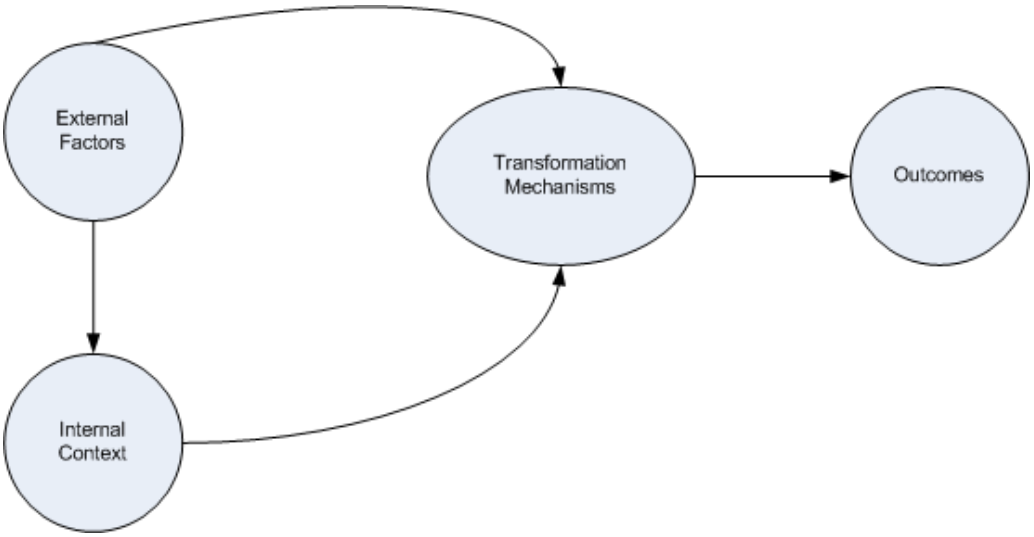


Figure 2.1: Lean Implementation Performance Model (Worley & Doolen, 2006)

External factors, internal context, transformation mechanisms, and outcomes were the four model components defined. Model components were further specified through a defined set of variables. These variables and the relationships between variables and the performance model are illustrated in Figure 2.2.

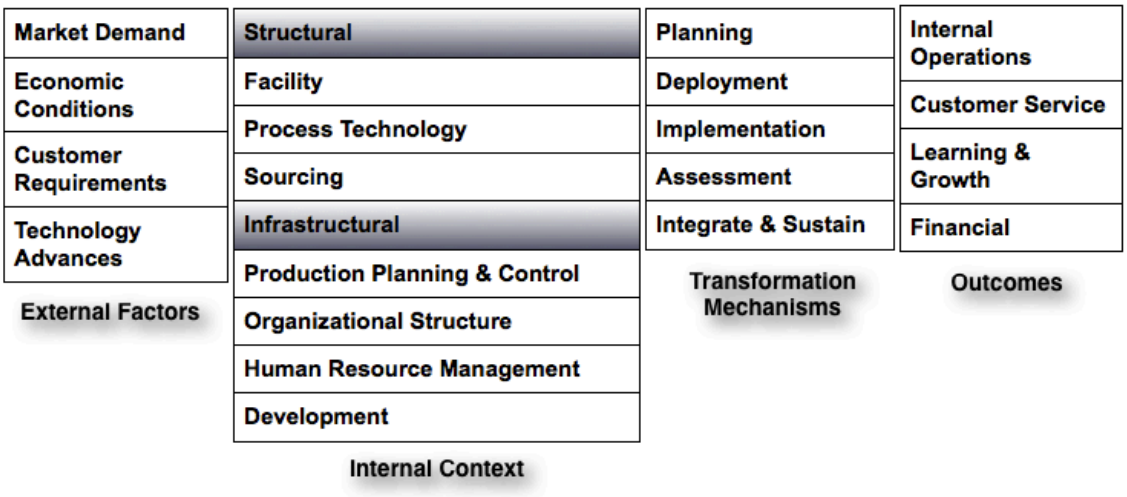


Figure 2.2: Performance Model Defined Variables

The external factors and internal context components were developed based on two existing manufacturing strategy frameworks. Structural and infrastructural categories were defined by Hayes and Wheelwright (1984) for evaluating manufacturing strategy. Structural categories included capacity, facilities, technology, and vertical integration. The four infrastructural categories were workforce, quality, production, planning / materials, and organization. In addition, Hayes et al. discussed the importance of external factors on manufacturing strategies (Hayes et al., 1988).

Miltenburg (1995) created another framework for manufacturing strategy. In this framework, the linkage between “manufacturing levers” and manufacturing performance was described. The Miltenburg framework includes human resources, organization structure and controls, production planning and control, sourcing, process technology, and facilities as manufacturing levers. This framework also included delivery, cost, quality, performance, flexibility, and innovativeness as outcomes.

The Deming cycle (Plan, Do, Study, Act) (1986) was used to define the transformation mechanisms. Finally, the balance scorecard concept was used to define the performance outcome component of the model (Kaplan & Norton, 1992).

The next few sections present literature summaries on topics related to computer science domain. In the next section, a summary of Natural Language Processing literature is presented.

2.2 Natural Language Processing

Natural Language Processing (NLP) is a joint field of computer science and linguistics that studies the interactions between computers and human natural languages. The term natural language is used to distinguish human languages such as English, Spanish, or Persian from computer languages such as C++, Java, or HTML.

NLP usually refers to computer systems that analyze, attempt to understand, or produce one or more human languages (Allen, 2003). NLP systems are able to work with a variety of different input and output formats. The task of an NLP system might be to translate between two languages, to comprehend and represent the content of text, or to generate summaries. NLP systems are also able to model language computationally and are used in many different systems such as text to speech (text to speech is computer software that produces human speech artificially), cooperative interfaces, multilingual interfaces, machine translation, and message understanding systems (Joshi 1991). When human natural language is analyzed and understood by a computer system, it is possible to employ several knowledge discovery techniques to extract useful information.

2.3 Knowledge Discovery

Different definitions of the term knowledge discovery in database (KDD) can be found in the literature. According to Fayyad et al. (1996) KDD is defined as “the non-trivial

process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data." (p. 24).

The purpose of analysis in KDD is to discover patterns, hidden information and connections that exist in data. The initial data can be in a variety of forms including interviews, observations, managerial reports, technical documents, etc. The Cross Industry Standard Process for Data Mining (Crisp DM, 1999) defines several steps to extract useful patterns from a dataset. The main steps are as following: (1) business understanding, (2) data understanding, (3) data preparation, (4) modeling, (5) evaluation, and (6) deployment. A schematic of the KDD is shown in Figure 2.3. The data preparation step affect the selection of a suitable data mining algorithm. In the next section, a summary of Machine Learning literature is provided. Machine Learning tools and techniques can automate the knowledge extraction process.

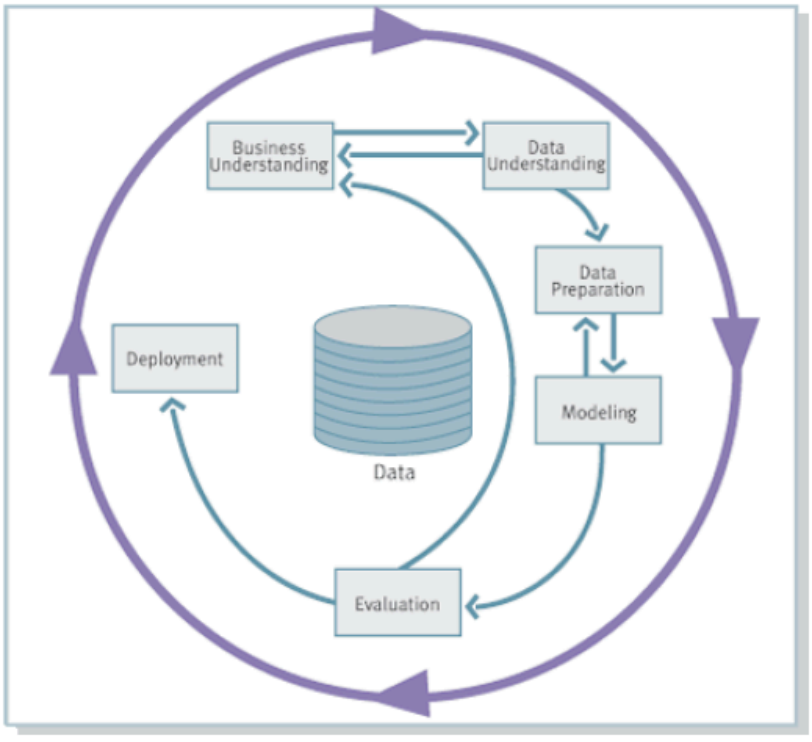


Figure 2.3: Main Steps of KDD (Hotho et al., 2005)

2.3.1 Machine Learning and Data Mining

Machine Learning (ML) lies in the area of artificial intelligence and is focused on techniques for computer software to learn from analyzing data. Mitchell (1997) presents a few common machine learning methods that can be used in characterizing the ML literature. These methods include classification, supervised and unsupervised learning, transduction, etc.

ML is a very active research field in the artificial intelligence domain. ML helps find solutions to many problems in computer vision, speech recognition, and robotics. The application of ML methods to large databases is called data mining.

Research activities in the area of data mining are very broad. Data mining continues to be an active research area in many disciplines. Data mining is often used as synonym for KDD, which means that data mining is the process of knowledge extraction from data sets. Some researchers use data mining as part of the knowledge extraction process in KDD, which deals with pattern recognition, search algorithms, and methods (Hotho et al., 2005). Other researchers prefer to refer to data mining as a separate field that deals with searching for valuable and interesting information in large quantities of data (Kumar and Joshi, 2003). In the next section, another division of data mining is introduced.

2.3.2 Text Mining

Text mining lies in the area of KDD. Usually text mining refers to knowledge discovery from text or applying data mining techniques customized for text. The term text mining was coined by Feldman et al. (1995). The term was used to describe computer supported analysis of text. Text mining is a newer area of research and incorporates a wide range of techniques from information extraction and natural language processing (NLP) and combines them with methods and algorithms from KDD, data mining, and machine learning. In other words, in text mining, the same analysis procedures as used in KDD are

applied. However, instead of having organized data, text mining is focused on documents that contain unstructured data. As a result of dealing with unstructured text, several questions about method selection and modifications to the models and algorithms are needed.

Text mining research is conducted in three different areas. Information extraction is about finding facts from text. Text data mining, which can be defined similar to data mining, is using methods and algorithms to find patterns in text. In this case pre-processing of data is very important and has a significant effect on the results. Modifications must be made to prepare data for the application of existing data mining methods and algorithms. In the final perspective, text mining is performed by following the steps in the knowledge discovery model (Crisp, 1999). In general, the knowledge discovery process in text can be defined as the extraction of not yet discovered information in a large collection of text (Hearst, 1999).

Techniques available from the knowledge discovery domain, and more specifically from text mining, can be helpful in study areas dealing with large amounts of data. Qualitative data analysis is one of those areas that can benefit from automatic knowledge extraction processes.

2.4 Qualitative Data Analysis (QDA)

Taylor-Powell and Renner (2003) introduced a general framework for conducting qualitative research. This framework is presented in Table 2.1.

Table 2.1: A QDA Framework proposed by Taylor-Powell & Renner, (2003)

1	Get to know the Data: read and re-read your data
2	Hypothesize the research model
3	Focus the analysis: review the purpose of the evaluation, Identify key questions
4	Categorize information: identify themes and patterns - Coding the data
5	Identify patterns and connections within and between categories
6	Interpretation: Bring all together

Auerbach and Silverstein (2003) present another framework for qualitative research using grounded theory. This framework is summarized in Table 2.2. Grounded theory enables a researcher to begin a research study without having to test a hypothesis. Rather, hypotheses emerge by listening to what participants say. Because the method involves developing hypotheses after data are collected, it is called hypothesis-generating research rather than hypothesis-testing research. The grounded theory method uses two basic principles: (1) questioning rather than measuring, and (2) generating hypotheses using theoretical coding. (Auerbach & Silverstein, 2003)

Table 2.2: The Hypothesis-Generation Research Design Recipe (Auerbach & Silverstein, 2003)

1	Conduct a literature review and identify your research issues
2	Define the research concerns
3	Create a narrative interview
4	Choose an original research sample based on your research concerns. Choose subsequent samples to expand and refine your theory
5	Decide on sample size

Qualitative or narrative data is a form of data that are derived from different forms and formats: including free-form text, open-ended questions, surveys, interviews, artifacts, logs, diaries, etc. Qualitative data deals with descriptions and can only be observed and can not be measured. In other words, qualitative data or narrative data consist of words and observations. Like all data, qualitative data need interpretation and analysis to bring order and understanding. Such interpretation and analysis requires both a systematic approach and creativity.

The qualitative approach to research leads to studies that are quite different from those designed using more traditional scientific approaches. The traditional approach, often referred to as quantitative approach, typically leads to hypothesis testing research, whereas the qualitative approach often lends itself to hypothesis generating research (Auerbach & Silverstein, 2003). Unlike quantitative analysis, researchers do not seek precise measurement and analysis of target concepts, but instead researchers need to become immersed in the subject matter. Seidel (1998) defines qualitative data analysis (QDA) as a process of noticing, collecting, and thinking about interesting things. Sullivian (2004) defines QDA as a broad set of practices for dealing with data that does not lend itself to numeric representations. QDA techniques require that a researcher spend

vast amounts of time reading and evaluating text from transcripts. This step of QDA is important as it helps researchers become familiar with the data and allows the researcher to begin building hypotheses.

QDA begins with making observations and producing records. Records are documents that can be created from field notes, interviews, surveys, etc. The process by which a researcher identifies interesting things in records and begins to shape the analysis produces a coding schema. This stage is also referred to as “getting to know the data” in the literature (Taylor-Powell and Renner, 2003). In QDA, the available data must be completely understood, which requires the researcher to read the text many times. Also, just having data does not mean that data are of high quality. The analysis continues by collecting and sorting instances of “things”. This stage is very similar to working on a jigsaw puzzle and sorting the pieces of the puzzle into different groups to help recognize the overall shape. In QDA, when a piece of data is identified, it is “noticed” and “coded” and when the pieces are sorted, they are “collected”. Coding is assigning abbreviated codes of a few letters, words, or symbols and placing them next to the themes and ideas found in the text (Taylor-Powell and Renner, 2003).

Data can be coded with two different types of coding schemas. Objectivist codes are replacing parts of the text with well produced codes that accurately represent that part of the text. In contrast, heuristic coding schema flag and sign post important parts of the text (Seidel, 1998).

In the next stage, researchers begin to make sense out of each collection. Researchers look for patterns and relationships existing within and across the collections. Assessing the relative importance of different cases and highlights can be very helpful to analysis at this stage.

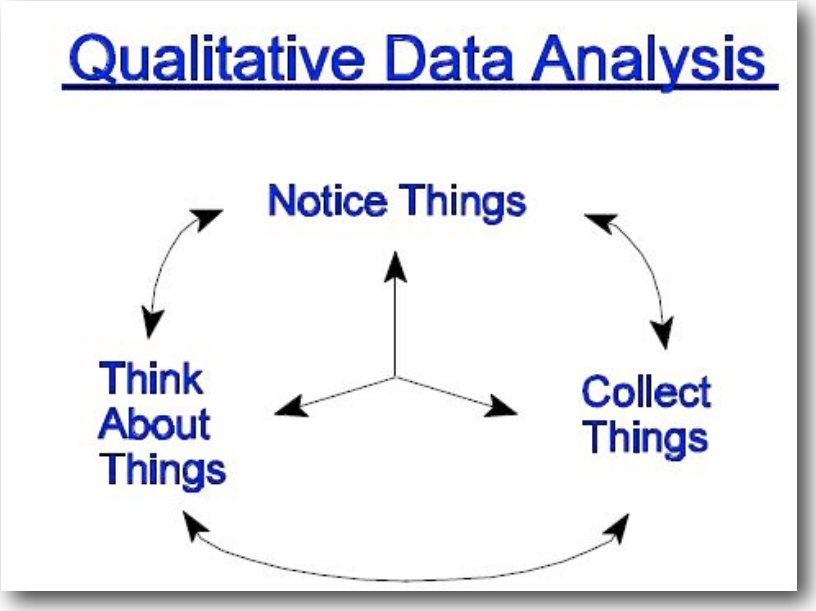


Figure 2.4: Schematic Qualitative Data Analysis Lifecycle (Seidel, 1998)

The “thinking” stage, is focused on making general discoveries from the data. Recognizing relationships and connections can help explain findings. To justify these findings in a more systematic manner, hypothesis testing can be carried out. Hypothesis testing research investigates a phenomenon in terms of a relationship between an independent and dependent variable, both of which are measurable numerically. This relationship is called a hypothesis. The aim of the research is to test whether the hypothesized relationship is actually true, using statistical methods. Difficulties in using hypothesis testing research in social systems arise since often a researcher does not know enough to state meaningful hypotheses or because the researcher does not know enough to select meaningful independent and dependent variables.

In qualitative data analysis, computer software can be used to speed up the study and to help the researchers. In the next section, a summary of available software packages for QDA and a comparison of QDA software is presented.

2.4.1 Software for QDA

Choosing between QDA software packages and finding the most appropriate package can be very challenging. To select the most appropriate application, the researcher must answer a number of questions. For example one question is what are the pros and cons of using specific software. Another question is how the software affects the process and end product of the research. Finally, the researcher must determine whether or not a particular package will suit all research objectives.

American Evaluation Association (AEA) has published a comprehensive list of QDA software packages. The list contains product names, developer company contact information, supported operating systems, and license pricing information (2010, January 26). Table 2.3 contains a subset of products listed in the AEA catalogue. The table captures the information for those applications that are used more often. All of the products listed are able to manipulate multiple types of qualitative data including text, audio, pictures, and video.

Table 2.3: An Overview of Existing Qualitative Data Analysis Software Packages

Product
Product: AnSWR **Developer:** Centers for Disease Control: Information Systems Support Services (CISSS) Division of HIV/AIDS Prevention, Epidemology Branch **Developer Site:** http://www.cdc.gov/hiv/software/answr.htm **Platform:** Windows 95,98, NT, 2000, ME, or XP **Price:** Freeware Software
Product: Atlas.ti **Developer:** Thomas Muhr **Developer Site:** http://www.atlasti.com/index.php **Platform:** Windows 95, Windows 98, Windows NT 4.0 **Price:** $395.00 (Educational-Single Copy); $715.00 (Standard- Single Copy); $1,580.00 (Educational-5 Pack); $2,860.00 (Standard-5 Pack); $1,440.00 (Educational Workstation-Qty. 5); $2,640.00 (Standard Workstation-Qty. 5)
Product: ELAN **Developer:** Max Planck Institute for Psycholinguistics Developer Site: http://www.lat-mpi.eu/tools/tools/elan **Platform:** Windows 2000/XP/Vista, MacOS X, Linux **Price:** Freeware
Product: Ethnograph **Developer:** Qualis Research Associates **Developer Site:** http://www.qualisresearch.com **Platform:** Windows 3.1, 95, 98/PC **Price:** $295 (single copy); $200 (Student copy)
Product: HyperRESEARCH 2.8.2 **Developer:** ResearchWare Inc **Developer Site:** http://www.researchware.com/ **Platform:** MacOS, Windows **Price:** $370 USD (Student discounts and institutional licensing is available. See http://www.researchware.com/hr-offer.htm for Student pricing. Contact support@researchware.com for Institutional licensing.)

Product
Product: MaxQDA **Developer:** MaxQDA **Developer Site:** http://www.maxqda.com/ **Platform:** Windows/PC **Price:** $701
Product: NVivo 8 **Developer:** QSR International Pty Ltd (QSR) **Developer Site:** http://www.qsrinternational.com **Platform:** Microsoft Windows 2000 Professional, MIcrosoft Windows XP SP2+, Microsoft Vista **Price:** Contact QSR
Product: XSight **Developer:** QSR International Pty Ltd **Developer Site:** http://www.qsrinternational.com **Platform:** Microsoft Windows 2000, Windows XP, Windows Vista **Price:** Please contact QSR

Weitzman and Miles (1995) conducted a comprehensive review of existing QDA applications. The research identified QDA software that supported theory building functionality. Atlas.ti and Nudist (the developer company has changed the name of this product to NVivo recently) was rated at the head of the field (Lewis, 2004).

> ATLAS.ti and NVivo are among the best available and potentially most useful qualitative data analysis (QDA) tools. Both products enable the researcher to associate codes or labels with chunks of text, sounds, pictures, or video; to search these codes for patterns; and to construct classifications of codes that reflect testable models of the conceptual structure of the underlying data. Both are tremendously flexible programs that can be readily applied in a wide range of applications (p. 439).

There is no clear winner between Atlas.ti and NVivo. To make a choice between these applications, strengths and weaknesses of both applications should be weighted and matched against research needs.

Using software in conducting qualitative research can speed up the process. Coding documents with the assistance of computer applications can be easier and faster than coding by hand. Existing QDA applications provide users with a variety of tools and features such as search options, easy ways to copy and paste pieces of the text, etc. These functions assist researchers in manipulating qualitative data, and there is a great advantage in using these functions over completing these tasks manually. However, currently available QDA tools are not really more than a digital paper and pencil. In other words, existing tools are not equipped with any sort of automatic processing features. In the next section, the classical procedure of QDA is described, and the challenges of conducting a qualitative study are summarized.

2.4.2 Classic Procedure of Qualitative Data Analysis

In this section, the classical procedure of conducting a qualitative study is briefly described. A summary of this procedure is presented in Figure 2.4.

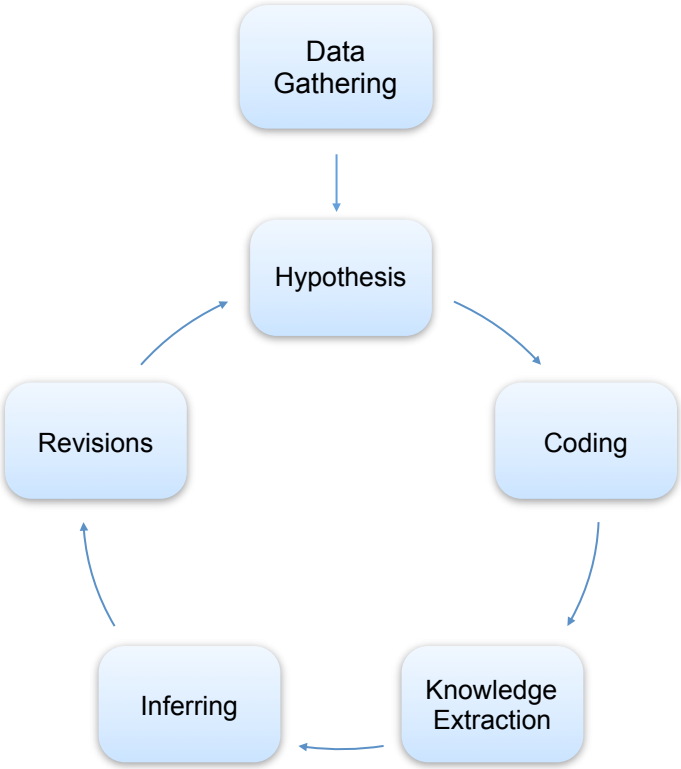


Figure 2.5: Classic Procedure of Qualitative Data Analysis

First, a comprehensive study of the relevant literature is performed. Researchers must achieve a high level of understanding of the system under study. This stage may include physically attending the system's site to perform direct observations and data gathering. It is important to prepare for systematic data gathering before conducting any data collection.

Based on the literature and available data, a model will be hypothesized and using the model, a coding schema will be developed. This coding schema is the major tool for QDA and is used to add semantics to data. Linguistic characteristics of the words that describe the relationships between words and their meaning, must also be considered. Analysis starts with manual coding of the data (See Appendix A). The coding of the data can be done either by hand or by using computer software.

Because qualitative research relies on data collection methods, it is subjected to biases such as researcher bias or over-reliance on one source (Taylor-Powell & Renner, 2003). To negate the possible effects of these biases the researcher must develop a systematic approach for data collection and analysis. Also, to minimize bias effects, manual coding is usually performed by a group of experts in the study domain. These experts read all the available data a number of times to get a thorough understanding of the data.

The model developed from the literature is next used to tag data using the developed schema. Tagging is the process of coding segments of text with predefined categories (that are derived from the coding schema). The segments can be single words, phrases, sentences or entire paragraphs. Coded segments can overlap as well. The coding process is an iterative process. This means that a researcher must go back and forth between documents and within sections of documents several times to code segments and to keep coding of all segments congruous. Sometimes, it is necessary to make changes in the coding schema itself. This overall process is time consuming. In the final step, the researcher puts all of the coded pieces together, trying to discover relationships. Facts and evidence are extracted to provide justification for behaviors and relationships observed in the system under study.

Researchers must read a large amount of data in qualitative studies. Therefore, a reduction in the amount of data, without losing essential information can be helpful. An overview of automatic summary generation techniques is presented next.

2.5 Automatic Text Summarization

The amount of available information has grown rapidly, and there is an increasing demand for automatic methods for text summarization (Mitra et al., 2002). Automatic summarization is defined as “reducing the size of a text while preserving its information

content" (Mani & Bloedorn, 1998). Summary of a text allows readers to get a sense of the content without reading all the sentences in the full text. Text summary allows readers to find what they need more quickly by assimilating only essential information from many texts with less effort (Yu et al., 2007). Although automatic summarizations do not guarantee great narrative coherence, they do provide approximate content of a text for review and judgement (Neto et al., 2002).

Automatic text summarization, which is also called automatic summary generation or automatic sentence reduction in the literature, is a popular topic in computer science and machine learning domains. Automatic text summarization is classified under Automatic Text Processing, which is an active research field. A large number of different techniques for generating automatic summary extractions have been introduced in the literature (Jing, 2000; Amini et al., 2005; Neto et al., 2002).

In general, summary generation is a complex task and needs deep natural language processing capabilities (Mitra et al., 2002). The most common approach to performing text summarization is extraction (Afantenos, 2005). An extractive summary is a subset of the sentences of the original text (Neto et al., 2002), and the extractive summarization approach extracts sentences or parts of sentences from text (Yu et al., 2007; Sparck-Jones, 1999).

The extractive summarization approach is designed based on lexical chains. Lexical chains or lexicons are sequences of related words in which lexical cohesion occurs between them. The lexicons are used to relate different parts of a text for the purpose of extracting important sentences (Yu et al., 2007).

Edmundson (1998) has proposed four methods for determining important sentences of a text: the presence of high frequency content words or keywords, pragmatic words (cue words), title and heading words, and structural indicators or sentence location. In the first method, lexicons are identified from sets of words that are semantically related (Yu et al., 2007). This method is very efficient, because there is no need for vast knowledge

bases or deep semantic analysis. All that is required for identifying the lexicons is a generic knowledge database that contains nouns and their associations. For English text summarization, a considerable number of lexical resources are available (for example see WordNet at: http://wordnet.princeton.edu/). In the next section, similar research conducted in different areas of study are summarized. The motivation behind these works is to automate time consuming processes that were completed manually.

2.6 Similar Research

As a part of the literature review for this thesis, a survey to identify similar prior work and research was performed. This thesis used a dataset from a study of engineering management and lean manufacturing, QDA, NLP, and text mining. However, the approach used in this study and the developed framework are very unique since no previous research was identified where NLP and text mining techniques were used in conducting qualitative research on data from the engineering management domain. In this section, previous studies where NLP and text mining have been used in other domains are summarized.

Machine learning and data mining techniques are heavily used in other knowledge domains. Bioinformatics is a good example of a domain that employs a wide range of knowledge extraction and creation techniques to study and analyze enormous data sets. Practices such as data mining and text mining have been used in studies on decryption and pattern recognition of genes and protein classifications (Cohen & Hersh, 2005). NLP and machine learning techniques have been applied to analyze patient data and human phenomena (van Driel et al., 2006) and are common in the bioinformatics literature.

The most relevant work found in a review of the literature was a PhD dissertation titled ‘Dynamic Semantic Annotation of California Case Law’ by Bransford-Koons (2005). Bransford-Koons worked on a computer system, named Amanuensis, to extract

information from legal court opinions and to answer questions about these opinions. The Amanuensis system used the General Architecture for Text Engineering (GATE) framework to extract knowledge from California appellate and Supreme court text containing opinions. The Amanuensis system used OpenCyc knowledge repository and reasoning system to create a question answering system suited for the processed court opinions.

There are several characteristics of the court text opinions analyzed in Bransford-Koons's study that supported the automated analysis. First, words and phrases used in opinions are common and well-defined in the practice of law. Second, court opinions are structured and published using a standard format, which makes court opinions well-suited for machine processing. Bransford-Koons was able to use these features to build additional knowledge from the opinions.

There is a large amount of research in the literature on automatic text summarization. Each of which uses different methods for text summarization (Neto et al., 2002; Jing, 2000; Massih et al., 2005). The majority of previous research classified in this category is on the topics related to the computer science domain. Of the works reviewed, the research study conducted by Yu et al. (2007) is most relevant to this study. There are two major types of summarization evaluation methods: intrinsic evaluation and extrinsic evaluation. Intrinsic evaluation compares automatically generated summaries with ideal reference summaries. Extrinsic evaluation measures the performance of automatically generated summaries in a particular task such as classification. In Yu et al.'s work, a human summary is constructed manually and used as an ideal reference to evaluate system performance. In the evaluation, the generated summary is compared to the summaries produced using a computer system. Machine learning measures, such as precision and recall, are then used to evaluate the performance.

In this chapter, a summary of literature from a number of different domains was provided. The literature summary on lean manufacturing and QDA provides readers with

background on the performance model leveraged for this study. The rest of the literature presented in this chapter summarized achievements in NLP, Machine Learning, and Data Mining, that were applicable to this research. The tools and techniques introduced in these areas were used in the automation of manual processes completed in a qualitative study.

Chapter 3 - Research Methods

In this chapter, the methodology used in this study is described. First, the design of the study is presented. The GATE platform and its modules are introduced. Next, performance evaluation techniques used in this study are described. Last, the steps of the proposed methodology are discussed in detail.

3.1 Study Design

The corpus provided for this work contained transcribed interviews saved as text files in standard Microsoft Word document (DOC) format. These data were in three different sets, each pertaining to an independent organization. For each of the three organizations, there were about twenty interviews, each from an independent source. The files contained answers to more than twenty open-ended questions. Examples of several interview data files are provided in Appendix B. The collected documents were stored on the local hard disk, and a preliminary transformation was performed to convert all the files into XML and RTF formats. This transformation was necessary for the further analysis completed in this study. No extra cleaning or editing was performed on the data. At the first stage, both questions and answers were kept in the data set to evaluate the performance of different modules.

Unlike Bransoford-Koons's (2005) research using California court cases, the available dataset was not structured in a consistent manner. In other words, the data files used for this research did not use any pre-set format, and the text was free form i.e. nothing had a specific place. This lack of formatting and structure is what distinguishes the current research from previous research. Returning to the qualitative research framework proposed for this research, these differences can be summarized as:

- Lexical ambiguity in documents
- Large vocabulary used in interviews and data sources
- Not having a good idea of what potential users of this system would like to know about the data

3.2 GATE

The GATE (General Architecture for Text Engineering) platform is a collection of tools produced at the University of Sheffield in England. GATE is a complex set of utilities developed for language processing. The GATE developer's website introduces GATE as follows:

> GATE comprises an architecture, framework (or SDK) and graphical development environment, and has been built over the past ten years in the Sheffield NLP group. The system has been used for many language processing projects; in particular for Information Extraction in many languages. The system supports the full lifecycle of language processing components, from corpus collection and annotation through system evaluation. GATE is funded by the EPSRC and the EU.

The GATE framework provides language engineers with a collection of software and language engineering tools that enable the design of very flexible analysis systems without the need for any programming (Cunningham et al., 2002). GATE uses a graphical user interface that lets users interact with all existing tools and that allows users to customize the tools. In addition to the user interface, an API library is embedded allowing the user to easily communicate with the framework from other applications. An application programming interface (API) is an interface implemented by a software

program to enable interaction with other software, much the same way that a user interface facilitates interaction between humans and computers.

The GATE framework is developed with Java, and it is freely available at http://gate.ac.uk. The GATE framework is covered by the GNU Library License (http://www.gnu.org/copyleft/lgpl.html). The GATE framework has powerful tools for annotating documents (Kentne & Maynar, 2005). Annotation can be done in both manual and automatic modes. Thus, the existing annotation tools along with GATE's GUI can replace software like Nvivo and Atlas.ti. An advantage of using GATE for manual annotation, over other traditional software packages, is that GATE can be used without any cost to the users.

Document annotations in GATE work similarly to mark-up languages like HTML and XML. The annotation can be used to represent codes used in qualitative data analysis, and as a result, can enrich documents by adding concepts through the annotation process. The added annotations will be used for further analysis and document processing.

The GATE architecture has two fundamental classes of resources: language resources and processing resources. A language resource can be made of an individual text file, loaded as a GATE document, or a collection of texts, loaded as a GATE corpus. When a document is loaded into GATE, the text is separated from its markup using standoff annotations. A processing resource is a processing module such as a tokenizer or a name entity recognizer. Users can add new language or processing resources by selecting the language resources or processing resources menus in the left panel of the GATE platform.

The GATE framework consists of a set of components that complete all necessary processing of documents. These components are aligned in a specific layout called pipelines. This idea is very similar to the UNIX command piping feature. A collection of processing resources (components) organized into a pipeline are called an application. An application can be saved using a specific name and later invoked for repetitive tasks. Different types of pipeline are implemented in GATE, however normal pipelines and

corpus pipelines are used the most. Normal pipelines are applications that process single documents. Corpus pipelines are applications that can process an entire corpus. An application takes either a document or corpus as an input.

ANNIE is the predefined corpus pipeline in GATE. It is a simple application containing all basic tools of text processing. The collected interviews were converted to be prepared for processing using the ANNIE information extraction system. ANNIE consists of sets of different plugins including general purpose lexicons and rules to syntactically and semantically tag a corpus, in adding a sentence detector module, a part of speech tagger, and etc. Additional modules, rules, and lexicons were created in this study to identify segments of text related to lean manufacturing implementation domain and to complete automatic tagging and text summarization. In Figure 3.1, a processed document in the GATE graphical user interface and two specific annotations of interest are illustrated. These annotations were generated after supplement of Gazetteer lists and rules were added to the system.

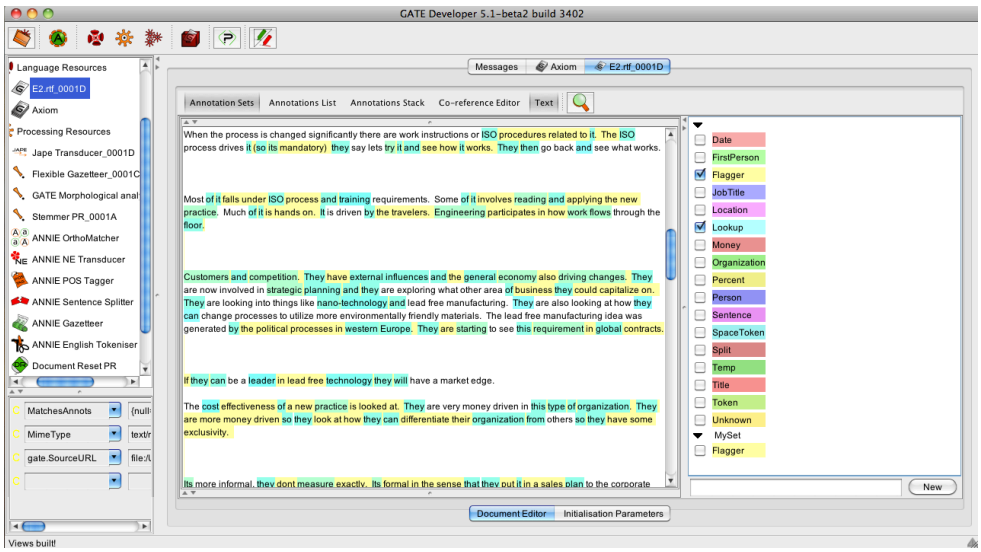


Figure 3.1: GATE Graphical User Interface (GUI)

The output from ANNIE is passed to another set of modules, which uses tags to populate and generate more semantics based on the context. As a simple example, assume one of the employees interviewed talks about improving measurement techniques in the quality control department. Based on the conceptual model, instances of “Assessments” as a system outcome are of interest. Thus, measurement can potentially be an instance of assessment and the initial processing through the ANNIE will mark the phrase “measurement” as an “Assessment” related item. The tags generated by ANNIE would then be used to create instances in the knowledge base for an “Assessment” instance which will be grouped to the Outcome node of the model. Once a knowledge base is created, these tags can be used to highlight specific parts of the text, to ask questions about the content of the document and to discover new relations. In the next few sections, the GATE platform and tools available in this platform are introduced in detail. ANNIE components will be described in more detail next.

3.2.1 A Nearly-New Information Extraction system (ANNIE)

The GATE framework is enhanced with the ANNIE information extraction system. This system is a pipeline of utilities necessary for basic language processing and information extraction that is provided by GATE platform developers. The default and preloaded tools in ANNIE’s pipeline are document resetter, English tokeniser, Gazetteer, sentence splitter, and the part of speech tagger. Additional processing resources can also be added to the pipeline manually. An overview of the ANNIE system is summarized in Figure 3.2.

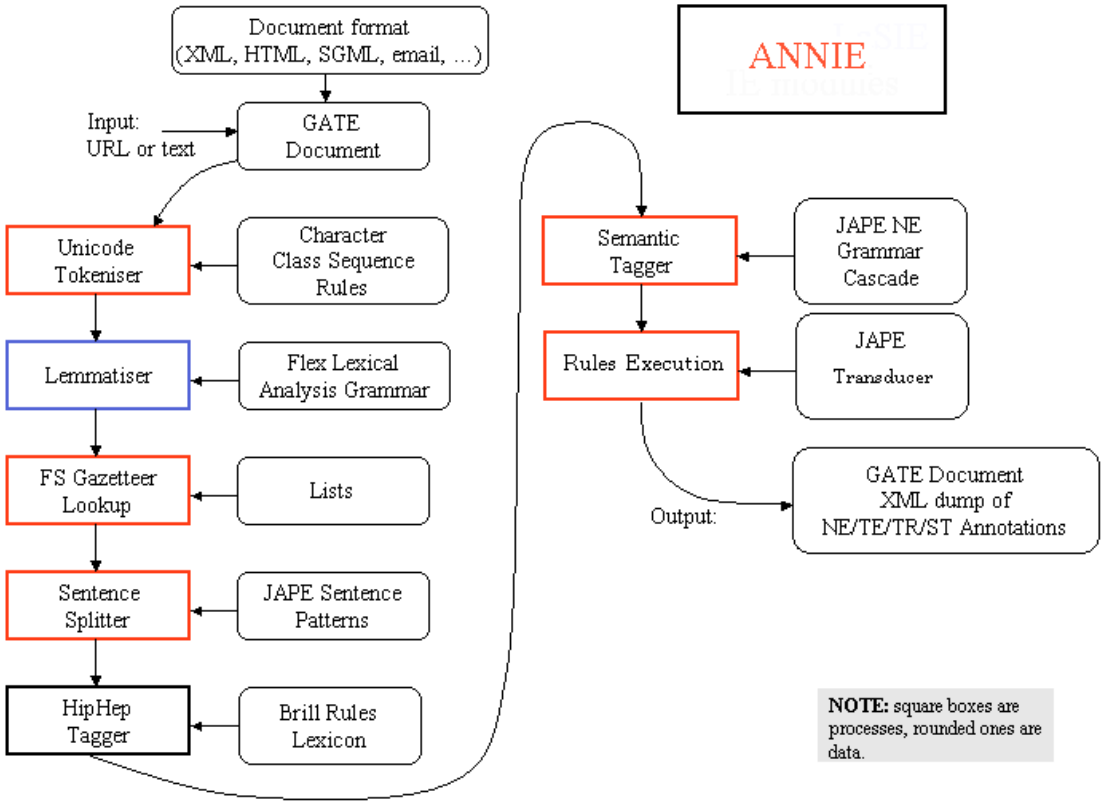


Figure 3.2: A Schematic Representation of the ANNIE System Structure

The document resetter component removes all existing annotations, resets the document to its original state, and prepares the document for a fresh run. The tokeniser component performs the process of converting a sequence of characters into a sequence of tokens such as numbers, punctuation and words of different types. A token is defined as a categorized block of text. The Gazetteer module is the lexicon system and was an important component in this study. Sentence splitter recognizes the boundaries of sentences. Regardless of the domain, this tool segments the text into sentences. Part of speech tagger uses a set of default lexicon and rules to produce part-of-speech tags as annotations on each word or symbol. Part of speech tagger is language dependent but domain and application independent.

For this research, several other tools were added to the default pipeline and some tools were modified. The tools that were added included the NE Transducer, which contains more rules for relationship extractions, OrthoMatcher, which lets the Gazetteer recognize similar words starting with upper or lower case letters, and the stemmer and morphological analyzer modules, which identify the root of each word (for example ‘study’ for ‘studying’ and ‘studied’ and ‘book’ for ‘books, and ‘booked’). The flexible Gazetteer module, which allows re-annotation of the document based on word roots and the uppercase-lowercase recognitions was also added. This module re-runs the Gazetteer list over specified annotation sets, which are the outcome of the stemmer and morphological analyzer.

The next significant module of the customized pipeline is the JAPE transducer. The JAPE transducer is a semantic tagger, which uses GATE’s Java Annotation Pattern Engine (JAPE). The ANNIE and JAPE modules were customized by creating new gazettes and JAPE rules. The application then applied the JAPE rules on the transcripts from the targeted dataset.

The tokenization process itself is a topic studied in computer science and more specifically in the NLP domain. However, it was necessary to understand this module enough to be able to use it in this project. The ANNIE tokeniser splits the document into smaller entities such as words, numbers and punctuations. A complete definition of recognized entities is provided in GATE’s user manual, portions of which are summarized in Table 3.1.

Table 3.1: GATE Platform Recognized Linguistic Entities

Entity Name	Description
Words	Any set of contiguous upper or lowercase letters, including a hyphen (but no other forms of punctuation).
Numbers	A number is defined as any combination of consecutive digits.
Punctuations	Three types of punctuation are defined: start_punctuation (e.g. '('), end_punctuation (e.g. ')'), and other punctuation (e.g. ':'). Each punctuation symbol is a separate token.
Symbol	Two types of symbol are defined: currency symbol (e.g. '$', '£') and symbol (e.g. '&', '^'). These are represented by any number of consecutive currency or other symbols (respectively).
Space Token	White spaces are divided into two types of SpaceToken - space and control - according to whether they are pure space characters or control characters. Any contiguous (and homogeneous) set of space or control characters is defined as a SpaceToken.

The tokeniser module identifies the beginning and the end of each token and assigns the token type as an annotation. The results of the tokenization process are shown in Figure 3.3.

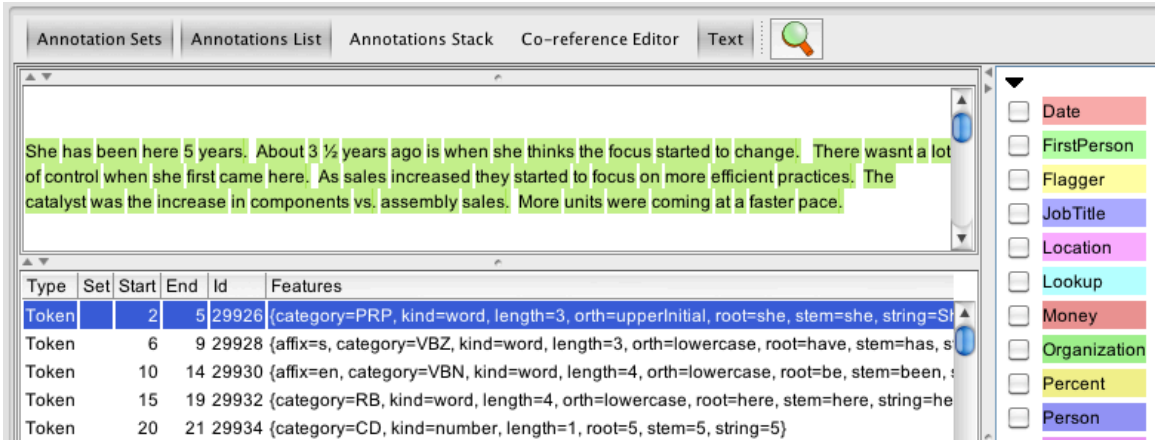


Figure 3.3: The Document Editor Module's Interface

The way that GATE's tokeniser annotates a phrase in a document, is shown in Figure 3.4. In this specific example, the word "She" is a word token that starts at node 2 and ends at node 5. The digit "5" is a number token which is located at node 20.

```
<GateDocumentFeatures>
<Feature>
        <Name className="java.lang.String">MimeType</Name>
        >Value className="java.lang.String">file:/Comp1/E2.rtf>/Value>
</Feature>
</GateDocumentFeatures>
<!--The Document content area with serialized nodes -->

<TextWithNodes><Node id="0" />
<Node id="1" />
<Node id="2" />She<Node id="5" /> <Node id="6" />has<Node id="9" /> <Node
id="10" />been<Node id="14" /> <Node id="15" />here<Node id="19" />
```

Figure 3.4: An Example of XML Annotation Tags

The tokeniser module can take care of simple tasks related to text processing. More complicated NLP capability can be added through JAPE. Although the tokeniser can be enhanced for more functionality, the current state of this module was adequate for this project.

In the GATE framework, the Gazetteer module is the central lexicon system. The Gazetteer can be easily modified by adding new Gazetteer lists. Gazetteer lists are added or modified by the user. In this project, Gazetteer list preparation was a critical step and is described in detail.

3.2.2 JAPE (JAVA Annotation Patterns Engine)

JAPE is a system for writing annotation rules that look for patterns in the text or in annotations created manually or by other components. JAPE also adds new annotations, if patterns are discovered. JAPE grammar consists of a set of phases, each of which

consists of a set of patterns and rules. The phases run in order and start a cascade of finite state transducers over annotations (Cunningham et al., 2000). JAPE rules consist of a left hand side (LHS) and a right hand side (RHS). The LHS of the rules consists of an annotation pattern that can also contain regular expression operators (e.g. *, ?, +). The RHS of the rule contains annotation manipulation statements. In other words, patterns detected by the LHS rules will be passed to the RHS, and designated labeling activities will be performed over the detected pattern. Examples are provided next.

3.2.3 Tokeniser

The tokeniser is the first resource used to process a document. The Tokeniser splits the entire document into very simple tokens such as different types of words, numbers, and punctuation. This component can distinguish between different punctuation and uppercase and lowercase words and is capable of doing deep linguistical analysis.

The Tokeniser component uses the JAPE grammar for creating rules. JAPE's rule structure stays valid for tokeniser rules. In a tokeniser rule, the RHS describes the annotations to be added to the annotation set. The LHS is separated from the RHS by '>'. A number of different operators can be used on the LHS. Four common operators are defined in Figure 3.5.

```
| (or)
* (0 or more occurrences)
? (0 or 1 occurrences)
+ (1 or more occurrences)
```

Figure 3.5: JAPE Grammar Operators

The RHS uses a semicolon as a separator and has the following format:

{LHS} > {Annotation type};{attribute1}={value1};...;{attribute n}={value n}

An example of a tokeniser rule is provided to demonstrate how to use the tokeniser. The following tokeniser rule detects words starting with a single capital letter:

"UPPERCASE_LETTER" "LOWERCASE_LETTER" >*
Token;orth=upperInitial;kind=word;

Based on the LHS, a word starting with an uppercase letter, followed by zero, one, or more lower case letters, can be matched by this rule. Regarding the RHS, the detected word (i.e. string of characters) will be annotated as “Token”. Two attributes are added to the token annotation set. The attribute “orth” (orthography) gets the value “upperinitial” and the attribute “kind” gets the value “word.”

3.2.4 Gazetteer

The Gazetteer lookup module identifies words or phrases in the document that are related to particular entity types by looking in a set of pre-stored lists defined by the user. The terminology for this specific project is included in a Gazetteer list and contains names or phrases that represent ideas taken from the research model. This list was used to identify the occurrences of instances of model concepts or specific codes. The results are produced and saved as a Lookup annotation. For example, words “hunger” and “food” have a strong logical connection to each other. Human minds think about food when the word hunger is present. This happens because humans look for food when a feeling of hunger arises. This relationship might not be always true. However, this relationship will hold true most of the times and can provide a clue to be used for automatic analysis and pattern recognition on a piece of text. The Gazetteer list in this framework works in a same fashion.

In the GATE framework, Gazetteer lists are stored as plain text files, with one entry per line. Each list contains a set of words, such as names of cities, organizations, etc. Figure 3.6 shows an example of a small section of the list for cities:

Figure 3.6: An Example Segment of Gazetteer City List

GATE developers have loaded the Gazetteer module with a large number of predefined datasets containing lists of countries and their major cities, airports, first and last names, a fair number of companies and organizations, etc. This predefined dataset can be very helpful for users creating new datasets. The most important advantage of having this predefined dataset is that the Gazetteer module can use the user-added lists along with preloaded lists to map meaningful text to each other, to get a better sense of data, and to add more content to the text. This also makes the GATE platform better able to recognize name entities, split text and perform morphological analysis or semantic annotations.

Gazetteer list collections are accessible by going to the path “GATE\plugins\ANNIE \resources\gazetteer”. Each list is stored as a file with an “.lst” extension. Any list collection for the Gazetteer must also have an index file. Based on the internal mechanisms of the Gazetteer plugin, the default name of this index file must be exactly ‘lists.def’. For each list in the Gazetteer a major type and an optional minor type must be assigned. An example of an index file is provided in Figure 3.7. Each index item consists of three segments, separated by two colons. The first segment contains the list name. The second segment shows the major type and the third segment shows the minor type

associated with the list. These lists are later compiled into finite state machines. Any tokens in the text that are matched by these machines will be annotated with features specified with the major and minor types.

```
country.lst:location:country
currency_unit.lst:currency_unit:post_amount
date_unit.lst:date_unit
time.lst:time:absolute
day.lst:date:day
```

Figure 3.7: Gazetteer List Index File Example

Another way for accessing the Gazetteer list collection is using the built-in Gazetteer editor. It can be accessed by double clicking on the ANNIE Gazetteer module in the GATE. This will bring up the Gazetteer plugin window in the main frame. The Gazetteer window is divided into two columns. The first column, which has the label “Linear Definition” contains list name, major type, and minor type, separated by a colon. The Gazetteer module window can be seen in Figure 3.8. Command buttons located at the top of the page can be used for adding new lists to the collection. Adding words and phrases can be done by first selecting a specific list in the left column and then typing directly into the right column of the module. Each word or phrase must be typed in a single line.

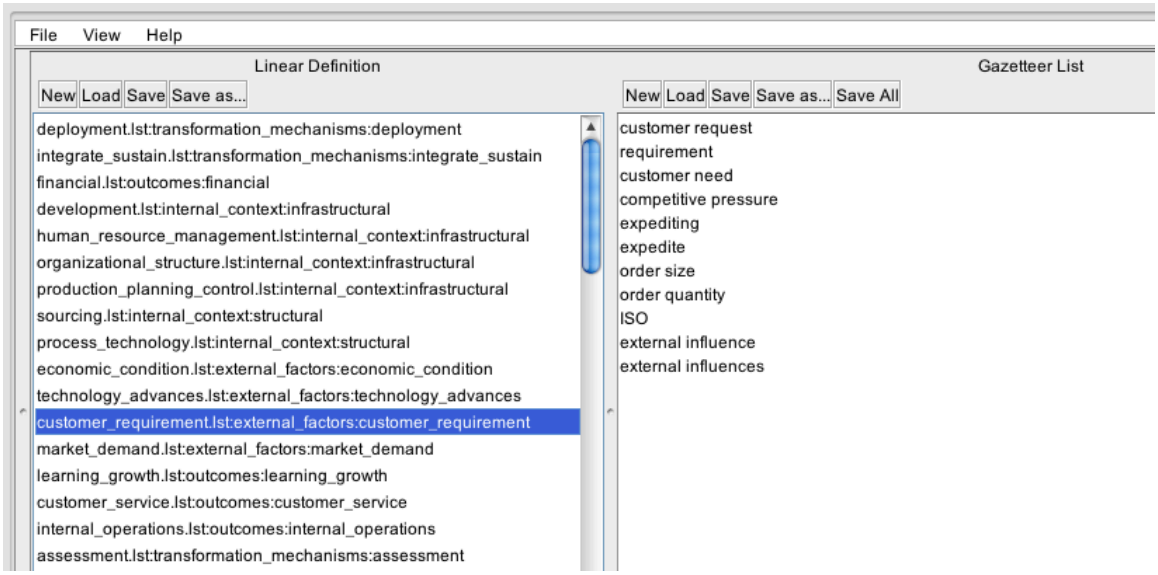


Figure 3.8: Gazetteer Module Interface

3.2.5 JAPE Grammar

JAPE grammar contains rules that act on annotations assigned manually or systematically in earlier phases to produce annotated entities. The general structure of JAPE rules also applies to JAPE grammars. The rules consist of LHS and RHS parts, separated by " >". The LHS performs pattern detection, and the RHS describes the annotation to be assigned. On the LHS, the patterns are described using existing annotations already assigned by tokeniser and Gazetteer components. In a JAPE grammar rule, a pattern can be defined in three formats:

- specifying a text string, e.g. {Token.string == "budget"}
- specifying the attributes and corresponding values for an annotation, e.g. {Token.kind == word}
- specifying a pattern by using defined annotation types in Gazetteer, e.g. {Lookup.majorType = "transformation_mechanisms"}

Grammar rules can become very complicated. Thus, macros can also be defined and used in the LHS of the rules. This means that instead of repeating patterns and information (patterns that are going to be used frequently) in the rules, the patterns can be defined in a macro and later be called in any other rule. A macro can also be called within other macros. The same operators used in tokeniser rules (e.g. *, ?, +, |) can be used in grammar rules.

Pattern definition statements are followed by a label for annotation. This label works as a local name for defined patterns and makes it possible to manipulate more than one pattern in a single rule. The RHS of the grammar rule contains information about the annotation. By addressing the label used in the RHS, the connection between the annotation and annotation information will be established. A new annotation will be added to the identified annotation using the name followed by the label in the RHS. Finally, attributes and corresponding values will be added to the new annotation. A set of JAPE grammar rule examples are provided next to demonstrate the use of JAPE language.

In the example below, a rule with the name “OutcomeDetectionRule” is shown. The pattern will get a new annotation with the name “outcomes” which contains two attributes. The first attribute, “kind”, is assigned the value of “Node”. The second attribute is the rule name, which has the name of the rule applied on the annotation as the value. The purpose of having the attribute “rule” is to identify annotation corresponding to grammar rules.

```
Rule: OutcomeDetectionRule
(
{Lookup.majorType == outcomes}
)
  :outcomesTag -->
```

:outcomesTag.outcomes = {kind="Node", rule=OutcomeDetectionRule}

Instead of defining a pattern inside a rule, one can define a macro and call the macro in the rule as shown in the following example:

```
Macro: OutcomeMacro
({Lookup.majorType == outcomes})

Rule: OutcomeDetectionRule
(
(OutcomeMacro)
)
 :outcomesTag -->
        :outcomesTag.outcomes = {kind="Node", rule=OutcomeDetectionRule}
```

Simple grammar rules do not usually contain Gazetteer lookups, making them understandable and straightforward. Using Gazetteer lookups in grammar rules can increase the potential for ambiguity. This type of rule that contains lookups, can cover a wider range of possibilities and can increase the need for more rules to describe all situations (Cunningham et al., 2000). This imposes a need for prioritization and ordering of rules.

For example suppose there is a need to develop a rule that captures IP addresses from the text. Because any IP address follows a specific format (four series of numbers separated by dots), any IP address can be represented with a simple format:

```
Rule: IPaddress
(       {Token.kind == number}
        {Token.string == "."}
        {Token.kind == number}
        {Token.string == "."}
        {Token.kind == number}
        {Token.string == "."}
        {Token.kind == number}
```

):ipAddress -->

:ipAddress.Ip = {kind = "ipAddress", rule = "IPaddress"}

Now consider another case where there is a need to capture date and time in the text. There is no standard representation for dates and times. Date's information, for example, can be shown in the following formats:

Wed, 10/7/00

Wed, 10/July/00

Wed, 10 July, 2000

Wed 10th of July, 2000

Wed, July 10th, 2000

Wed 10 July 2000

This type of ambiguity can be addressed in two ways. The first method is by using the context of the text. While the software does not have any sense about a text's meaning and the context of the content, a sequence of the words can provide context information. The following example demonstrates a simple use of context.

```
Rule: YearContext1
Priority: 40
({Token.string == "in"}|
 {Token.string == "by"}
)
(YEAR)
:date -->
 :date.TempDate = {rule = "YearContext1"}
```

Annotations located in the pattern are inserted between two round brackets. In the rule definition, the context of interest is described in exactly the same way as the pattern to be matched. The rule indicates that when an annotation of "YEAR" comes exactly after "in" or "by", it should be annotated with "Timex" and two attributes of kind and rule with their corresponding values should be added (assume an appropriate macro for

YEAR is defined before this rule). Thus, in this rule the interest is not in the annotation YEAR itself. Rather the interest is in those parts of the text that really indicate a date.

A second example built based on the context of the text is shown next. This rule captures only those "phone numbers" that are located between "<" and ">" signs.

```
Rule: Telephone
({Token.string == "<"})
(
  (TELEPHONE)
)
:telephone
({Token.string == ">"})
-->
        :telephone.Number= {kind = "phonenumber", rule = "Telephone"}
```

A second method for handling ambiguity is using control commands and priorities. Each grammar can have two possible control styles. These control styles are specified at the beginning of the grammar. When the "Brill" style is used, every rule that matches will be fired (in JAPE terminology, firing a rule means to execute a grammar rule when the requirements of that rule are satisfied). This means that a segment of text can be annotated many times with different rules and there is no need for any prioritization. When the "Applet" style is used, only one rule can be fired on a specific region of the text. Thus, there is a need for prioritizing rules. A general rule prioritization algorithm is defined as:

1. If the matching region of several rules overlap in a text document, the one covering the longest region is fired.
2. If more than one rule matches the same region of the text, the one with higher priority is fired.

3. If more than one rule with the same priority matches, the one defined earlier in grammar is fired (rules listed earlier in the grammar file, have higher priority).

Priority is an integer number and can be assigned to any rule. The higher the number, the greater the priority. If a priority is not defined for a rule, a default priority, which is -1, is automatically assigned. The following example illustrates how prioritization works. This example contains two rules which can capture the same text region.

```
Rule: Location
Priority: 25

(
 ({Lookup.majorType == loc_key, Lookup.minorType == pre}
  {SpaceToken})?
 {Lookup.majorType == location}
 ({SpaceToken}
  {Lookup.majorType == loc_key, Lookup.minorType == post})?
)
:locName -->
  :locName.Location = {kind = "location", rule = "Location"}

Rule: GazLocation
Priority: 20
  (
({Lookup.majorType == location}):location
  )
  --> :location.Name = {kind = "location", rule = GazLocation}
```

Assume the phrase “Everest mountain” is in the text. Also, assume “Everest” is defined as a “location” and “mountain” defined as “loc_key” of type “post” (which means that this is a key location, which usually comes right after a name or description) in the Gazetteer. Thus, the first rule will capture the entire phrase but the second rule only

captures the location annotation for “Everest”. Based on the rule manipulation structure, only the first rule will be fired, because it applies to a longer region of the text. Now assume that only “Everest” is in the text. In this case also, both rules will match the same region of the text. However, because the first rule has the higher priority, it will be fired. The controls and prioritization only operate within a single grammar. If there is a need for global priorities, all the rules must be defined in a single grammar file.

3.3 Annotation Evaluation

It is very important to analyze and measure the output performance of language processing tools. The processing environment should be equipped with tools to evaluate the results and specify the level of accuracy and reliability of the outcome. The GATE platform is equipped with an evaluation tool called AnnotationDiff which enables automated performance measurements. This tool is provided by GATE developers and uses the embedded graphical features of the GATE framework to present visual representations of the result. The graphical interface of the AnnotationDiff tool is illustrated in Figure 3.9.

GATE’s AnnotationDiff tool makes it possible to compare two different annotation sets in a document. With this tool, one can compare either system-generated annotations with a manually entered reference or two different versions of system-generated annotations. One set is considered as the key set and the other annotation set is assumed to be the response set. AnnotationDiff uses different colors to distinguish between two sets using metrics such as correct, partially correct, missing, or false positive.

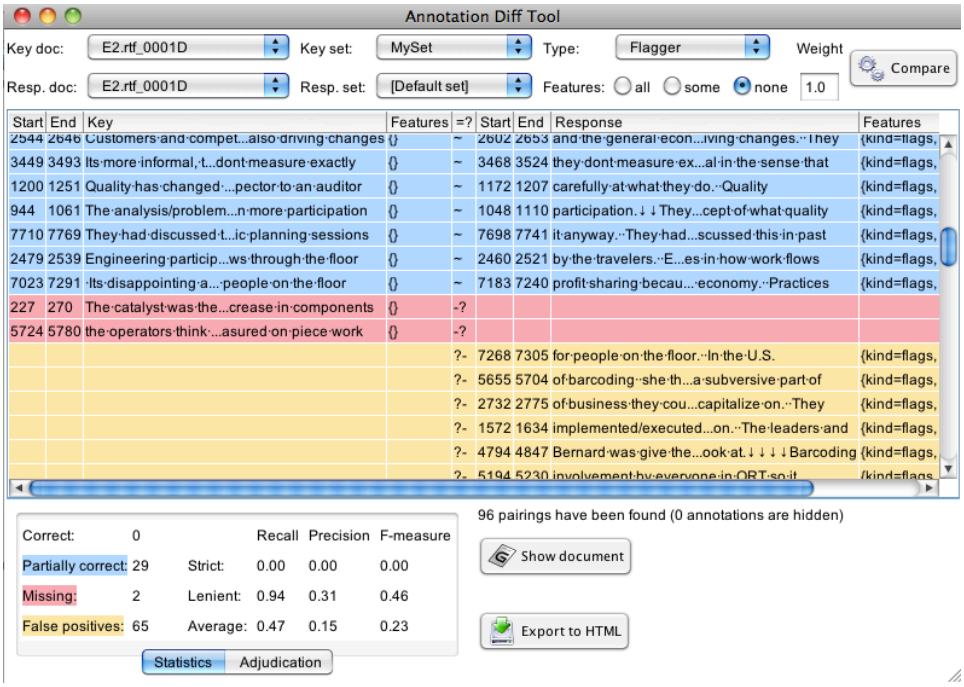


Figure 3.9: Annotation Diff Tool Interface

The AnnotationDiff tool calculates precision, recall, and F-measure as the performance measurements for each type of annotations. Many different performance measures have been proposed for performance evaluation of information retrieval systems. Most of the performance measures require both a document set and a query for measurement. Precision and recall have been long used as performance measurements for information retrieval and extraction (Hirschman et al., 1999). Due to the high correlation between these measures and human judgments, precision and recall are very useful performance measures and are widely used for evaluation of NLP systems (Melamed et al., 2003). The simple logic behind these measures is dividing a set of documents (or data) into relevant and irrelevant results to a particular query.

Precision and recall are defined in terms of a set of retrieved documents from a query and a set of relevant documents to a specific topic. In information retrieval, precision is defined as the number of relevant, retrieved documents divided by the number of

retrieved documents. For example, for a text search on a set of documents or in a database, precision is defined as the number of correct results divided by the number of all returned results.

$$precision = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{retrieved\ documents\}|} \quad (1)$$

Recall is defined as the fraction of relevant documents that are retrieved from executing a query. For example, recall for a text search is the number of correctly retrieved documents divided by the number of all existing, relevant documents.

$$recall = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{relevant\ documents\}|} \quad (2)$$

Different terms have been used to define precision and recall. Precision for a specific class is defined as the number of true positives (i.e. the number of items correctly labeled as belonging to the class) divided by the total number of items labeled as belonging to that class (i.e. the summation of true positives and false positives, which are items that have been incorrectly labeled as belonging to the class). In this context, recall can be defined as the number of true positives divided by the total number of items that actually belong to the class (i.e. the sum of the true positives and false negatives which are items which were not labeled as belonging to the class but should have been).

A perfect precision of 1.0 means that all retrieved documents were relevant, but does not say if all relevant documents were retrieved. A perfect recall of 1.0 means that all relevant documents were retrieved, but says nothing about the number of irrelevant retrieved documents. In practice, there is a reverse relationship between precision and recall, where an increase in one metric results in a reduction of the second metric. For example, an information retrieval system can often increase recall by retrieving more documents, but this is done at the cost of increasing the number of irrelevant, retrieved documents which decreases precision.

Since it is easier to work with a single measure of performance, precision and recall are used together and define a new measure called F-measure (or F-score). F-measure is defined as the weighted harmonic average of precision and recall. The equation for the weighted harmonic mean of precision and recall, which is known as traditional F-measure is shown in equation 3:

$$F - measure = \frac{2precision \times recall}{precision + recall} \tag{3}$$

In statistics, F-measure is used as a measure of a test's accuracy, where an F-measure reaches its best value at 1 and worst score at zero. According to the definitions of precision and recall, higher values of F-measure are harder to achieve. In the field of information retrieval, F-measure is used for measuring the performance of data classification or search and querying techniques. AnnotationDiff tool calculates performance measures in two ways. The Strict measure considers all partially correct responses as incorrect. The Lenient measure considers all partially correct response correct. AnnotationDiff shows the performance calculated in both ways as well as an average based on a half weight on each.

3.4 Research Methodology

In this research a different framework for QDA is proposed. This framework was developed specifically for a study of management practices associated with lean manufacturing implementations. However, the process used to develop the framework can be applied to other domains. The framework offers a set of tools to help researchers with viewing and editing documents, manual and automatic text annotation, and with automatic text summarization.

Text mining is one way to approach QDA. One of the earliest steps in text mining is reviewing sample text to gain some insight into the breadth of topics, variations in terms, levels of grammatical structures, use of domain-specific terminology and abbreviations, and the amount of noise, or dirty data, in the text. QDA can make this typically ad hoc process more systematic.

The proposed framework, takes advantages of text mining and NLP techniques to help researchers perform QDA easier and faster. The use of the GATE platform can bring additional intelligence to classic QDA and can save time and effort. The keywords method introduced by Edmundson (1998) was used to identify important sentences of the text, and available tools in the GATE platform were used to develop an automatic summarization system. A Gazetteer module was created to collect keywords and lexicons. This work draws heavily on the disciplines of information management, computer science and statistics and also reaches into the tangentially related areas of operations research (Sullivian, 2004).

The raw data used to develop and test in this framework were also used for a classical QDA study. This study focused on those steps of the research project, which occurred after data collection was completed. The developed framework utilized text mining and NLP features, and a series of setups and configurations were performed. The details of this process are summarized separately.

3.5 Performance Model Ontology

The performance model used for conducting the qualitative analysis for this research was introduced in the literature review section. This model was developed for a set of case studies conducted to characterize management practices used in lean implementations (Doolen & Worley, 2004). The performance model is summarized in Figure 2.1 and the structure used to operationalize each of the model variables is shown in Figure 2.2. In this research, a taxonomical hierarchy of the elements and variables was developed and is presented in Figure 3.10. The structural and infrastructural sub-elements of internal context were further defined using an extra level. This additional layer of variable subelements are presented in Figure 3.11.

Figure 3.10: Performance Model Ontology

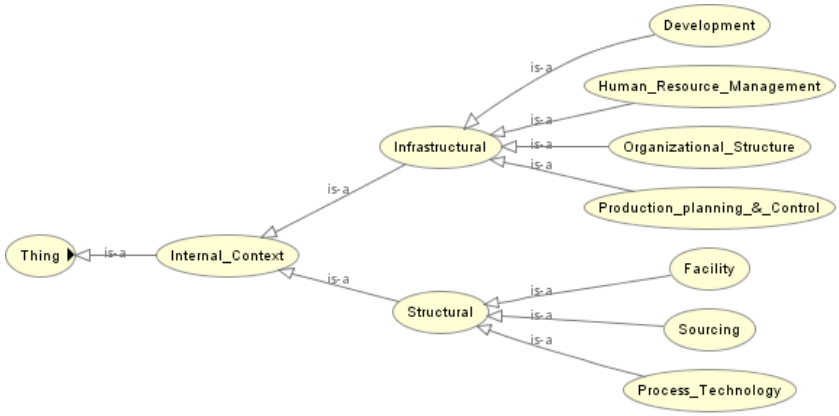


Figure 3.11: "Internal Context" Sub-Elements

The representation used for the performance model structure in Figure 3.10 is called an ontology. In information science, an ontology is a knowledge base that is created to link semantically related terms to each other (Hand et al., 2001). An ontology is a model for describing a domain and consists of a set of types, attributes, and relationship types. Ontologies define a common vocabulary for researchers to share information in a domain. Modern knowledge extraction techniques use ontologies to systematically annotate documents using a classified hierarchy (Borst, 1997).

Standardized ontologies have been developed in many disciplines. Domain experts use these ontologies to annotate and share information in their fields (Noy & McGuinness, 2001). While broad, general-purpose ontologies such as UNSPSC (developed by United Nation Development program and Dun & Bradstreet) for products and services terminology have been developed (www.unspsc.org), other special-purpose ontologies generated for specific domains have also been created. For example, in medicine, SNOMED contains large standardized and structured vocabularies (Price & Spackman, 2000). Having an ontology for a study domain is advantageous for researchers in that domain. The ontology gives researchers a common and standard vocabulary, which makes communications and sharing data easier and faster.

In this research, the Protégé platform (http://protege.stanford.edu/ Accessed on Monday, February 15, 2010) developed at Stanford University was used to build the ontology presented in Figure 3.10. Protégé is a free, open source ontology editor and knowledge base framework. A comprehensive tutorial for Protégé can be found at http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html.

Each node in the model represents a class of objects. Arrows in the model show a parent-child relationship. These arrows can be interpreted as "is-a" or "kind-of" relations. As examples, "Market Demand" is an "External Factor" and "Planning" is a kind of "Transformation Mechanism." In the hierarchy, classes that are a direct subclass of the same class are siblings. For example, "Assessment" and "Deployment" are siblings, since they are direct subclasses of "Transformation Mechanisms." For this study a top-down development process was used to define the classes and the class hierarchy (Noy & McGuinness, 2001).

Using the structure defined by the ontology, a comprehensive Gazetteer list was developed. The Gazetteer list was compiled from a set of independent text files. Each file contained a set of words or phrases representing an instance or a concept that belongs to one of the model variables. The Gazetteer list development process consisted of several steps, each of which added new instances to the list. These steps are described next.

3.6 Gazetteer List Preparation

In this section, the steps completed to develop a Gazetteer lists are presented. Gazetteer lists work as lexicons and are used to automatically annotate text, and to identify important parts of text.

3.6.1 Word Frequency Technique

A word frequency technique was used to identify the most common words in the documents based on frequency. Word frequency analysis ranks the most frequently used words used in any given body of text, usually ignoring common words. This analysis can help discover more important instances and concepts within a document or a set of documents. Word frequency is a good starting point for discovering words to use as instances for model variables and for developing Gazetteer lists.

Many software packages, including Microsoft Word, have word frequency counter tools. A number of free tools also exist online that specifically perform word frequency analysis. Hermetic Word Frequency Counter is one such free tool (http://www.hermetic.ch/wfc/wfc.htm). This software can scan multiple text files simultaneously and provide a count of the number of occurrences of different words.

3.6.2 Brainstorming

Brainstorming is a group, idea-generation technique that has been used for many years to generate a large number of ideas for a problem (Gallupe et al., 1992). The idea was first popularized by Faickney (1953) in Applied Imagination. In the literature, brainstorming is recognized as an effective tool for increasing creativity in organizations (Sutton & Hargadon, 1996).

A brainstorming process was performed for this research to add new words and phrases to the Gazetteer list collection. Similar to the word frequency technique, brainstorming can help to initiate Gazetteer collection development by shaping the preliminary structure of the lists. The brainstorming phase increased the collection size by adding a number of key terms and relevant words from both the engineering management and the lean manufacturing domains.

3.6.3 Word Lists

Word lists were created by combining the results of the word frequency analysis and brainstorming phases and by conducting another document review. In this phase all transcripts and artifacts were carefully reviewed. In this round of word selection, the document collection was skimmed, and additional related key words and phrases were identified.

A form was created with Microsoft Word. The form contained two-columns and was used to categorize extracted words and phrases. Figure 3.12 shows an example of this list. The first column contains the words, and the second column was left blank for the researcher to identify an appropriate category for each specific word.

An iterative process was performed to assign one or more categories to each word or phrase. Assigned categories were the lowest level classes depicted in the ontology model.

Word	Category
quality, quality problem	Internal operations
cycle time	Internal operations
practices	Implementation
limit on resources	Financial
error reduction, reduction in error	Internal operations
DPMO	Internal operations
time issue	Internal operations
operator	Implementation
quality issue, minor issue	Internal operations
assemble assembly	Process technolog

Figure 3.12: An Example of the Word Category List

A review of the list was performed by two researchers. The process was completed over a one month period. Each week the upgraded collection was entered into the

Gazetteer module for a test run. At this stage, no statistical analyses were performed to quantify Gazetteer module performance. The annotation outcomes were examined for a very small subset of the corpus to identify additional missing keywords, and necessary modifications were performed on the Gazetteer list. Through this review, new words and phrases were identified and added. This iterative process was repeated until the researchers agreed that the state of the collection reflected the key concepts of the domain. The completed Gazetteer list collections are provided in Appendix C.

3.7 Application Structure

To process text using the GATE platform, applications must be developed. In GATE, the term application has a specific meaning. In the GATE platform, an application is defined as a collection of processing resources organized into a pipeline. For this study a customized application pipeline was developed to offer all necessary processes for this research. This customized pipeline was created using the default ANNIE application. To initiate this application, a user can either load the ANNIE system with the default pipeline from the file menu (e.g. using the green circular icon in the toolbar having an 'A' in the center. See Figure 3.13) or create a new corpus application pipeline by right clicking on the "Applications" in the left menu and then manually adding the processing resources listed in Figure 3.14 one by one.

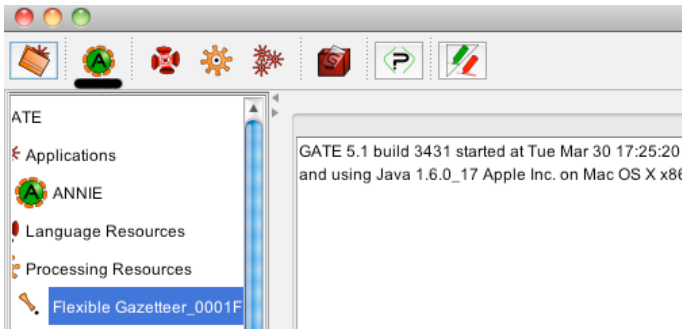


Figure 3.13: ANNIE Load Button

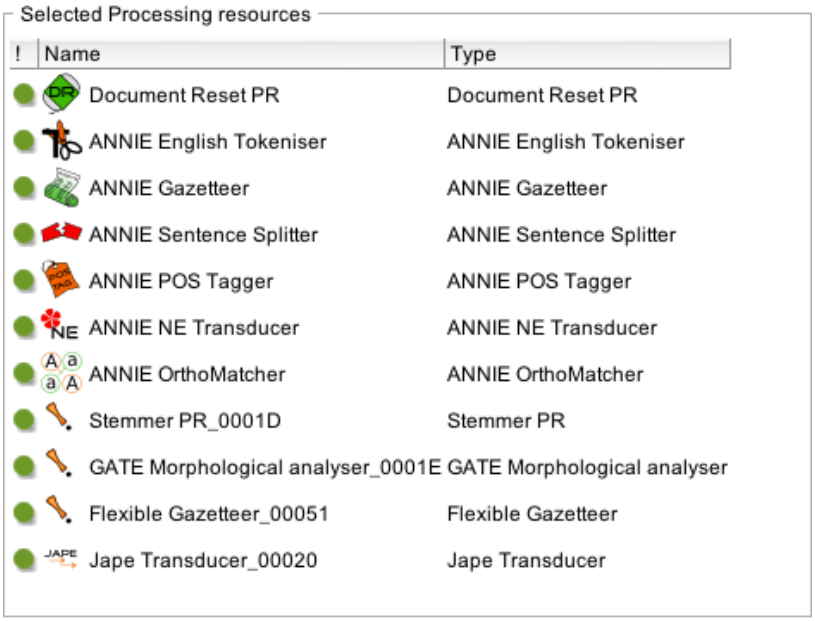


Figure 3.14: The Application Pipeline Developed for this Research

The first module in the pipeline is Document Resetter. This module removes all existing annotations from the documents stored in the corpus. It is very important to have this module as the first running process to delete any annotations remaining from any previous runs. Document Resetter can accept annotation sets as arguments to ignore from deletion. This option is very useful and can be used to prevent Document Resetter from deleting user defined annotation sets.

The tokeniser component performs the process of converting a sequence of characters into a sequence of tokens such as numbers, punctuation and words of different types. Since the English tokenization is necessary for further text analysis, ANNIE English Tokeniser was located in the application pipeline as the first processing resource right after the Document Resetter module.

Every time the application is run, the document corpus is processed by the Gazetteer module two times. At first, the corpus is processed by the standard ANNIE Gazetteer that searches for exact matches among words (both lowercase and uppercase formats) existing

in the corpus and words in the Gazetteer lists. This means that a particular word from a document is only annotated when the Gazetteer module can find it exactly as it exists in the Gazetteer list. In other words, suppose the word “optimization” is added to a Gazetteer list. If the ANNIE Gazetteer processor finds the word “optimize” in the text, it will be ignored. This is not desirable, since the word “optimize” might pertain to a part of the text that is related to “optimization” and as a result should be annotated under the corresponding class.

To address this issue, the text was further processed with both a Stemmer Module and a Morphological Analyzer. These two modules simply remove all prefixes and suffixes to generate the root form of each word. The stemmer annotates each token with a new feature “stem” that holds the stemmed version of the token as its value. The Stemmer Module does not complete any linguistic analysis, which sometimes results in weak outcomes. On the other hand, Morphological Analyzer considers tokenization and part of speech tags, one at a time, to identify a word’s lemma (the canonical form of a word) and an affix (an affix is a morpheme that is attached to a word stem to form a new word). The word will receive a “root” feature that holds the root of the word.

Stemmer and Morphological Analyzer are not present in the default ANNIE system. These plugins are called CREOLE plugins and are separately added to GATE. When there is a need to employ any of these CREOLE plugins, they must be loaded and added manually into the application pipeline. CREOLE plugins can be managed using the Manage CREOLE Plugins interface in the File menu. Figure 3.15 shows the graphical interface for selecting the plugins to be loaded.

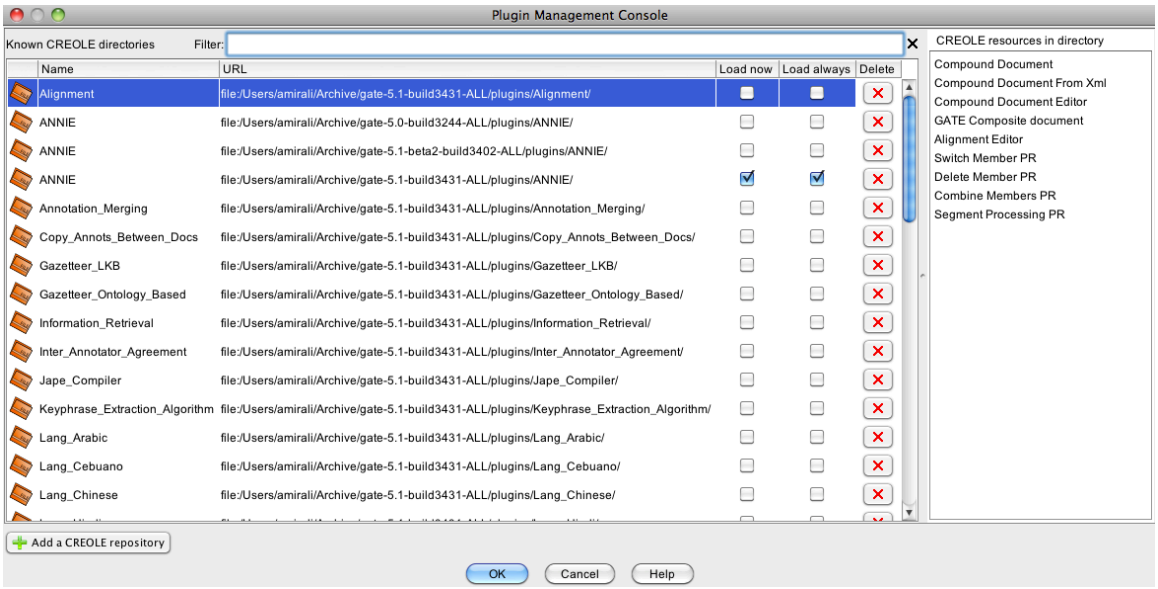


Figure 3.15: CREOLE Plugins List

This window contains a list of available plugins. Available plugins are stored on the hard drive and can be found in a folder named “Plugins” in the GATE application main directory. Installed plugins are found automatically when GATE starts up. For each plugin, there are two check boxes. The check box labeled ‘Load now’ is for loading a plugin for a single run. The check box labeled ‘Load always’ will add the plugin to the list of startup plugins. Stemmer plugin is listed as “Snowball Stemmer” and the Morphological Analyzer is a “Tools” plugin.

To add these plugins into the application, corresponding plugins must be loaded. A right click on Processing Resources or by selecting ‘New Processing Resource’ from the File menu will enable these plugins to be loaded.

After adding the Stemmer and Morphological Analyzer, a second Gazetteer module was executed over the corpus. However, this time, the Flexible Gazetteer module was used instead of the standard ANNIE Gazetteer. The Flexible Gazetteer does not have a Gazetteer engine and provides users with the flexibility of choosing the input and an

external Gazetteer. The flexible Gazetteer performs lookup based on the values of a feature of an annotation type using an external Gazetteer engine. Thus, in order to compare words in base form with those saved in the Gazetteer lists, features “stem” and “root” are used as inputs, and the standard ANNIE Gazetteer is used as the engine.

Between the first and second rounds of Gazetteering, a set of four ANNIE modules operated on the corpus. These four modules prepared the corpus for further analysis, by adding necessary annotations and linguistic features. Sentence splitter is a domain independent module that segments the text into sentences. POS Tagger annotates each word with a value based on the part of speech. Named Entity (NE) Transducer and Ortho Matcher modules work together to identify persons, organizations, and locations and to annotate them as such.

3.7.1 Loading Documents Into an Application

After creating a new application loaded with the necessary processing resources, the documents were loaded into the GATE platform as language resources. Language resources are used to store relevant documents, and can be created as a single document or as a corpus. In GATE, a corpus works similarly to folders in an operating system’s file manager.

To create an empty corpus, the user must right click on language resource and select “New Gate Corpus.” The user can either assign a name to the corpus, or GATE will assign a random name automatically. To add documents to the corpus, the user selects “populate” after right clicking on the created corpus. This will invoke a new dialogue box. In this window, the user must select the folder where the documents are located, the extension that the text files are save with (such as .txt, .rtf, etc.) and the encoding of the text files. The encoding often must be set to ‘UTF-8.’ Then, by clicking the ‘OK’ button,

the documents will be populated under the corresponding corpus. The user can access and modify the documents by double clicking on the corpus name from the right pane.

Each document is loaded and stored with a unique name in the corpus. For this study, file names started with either an ‘E’ or an ‘L,’ followed by a number. If the file name started with an ‘E,’ the document resulted from an interview conducted with an executive. If the file name started with an ‘L,’ the document resulted from an interview with a line worker.

3.7.2 Running an Application

After setting up the processing resources as one or more applications, the user can run the application by opening the application window. To open the application window, the user must double click on the application name in the right pane. Each application window has two panes. The pane on the left shows the processing resources loaded but not used in the application pipeline. The pane on the right shows the resources incorporated in the pipeline. The user can use the arrow buttons in the middle to manage processing resources. It is not mandatory to include all the loaded resources in the pipeline. Since the input of many modules are dependent on the output from one or more other modules, the order that processing resources are arranged in a pipeline is very important. For this study, the order used is summarized in Figure 3.14.

For almost any processing resources included in the GATE, a number of run-time parameters can be assigned. These parameters define processing behaviors during execution of the application. The user can access these parameters by selecting a resource located in the right pane of the application window. Typically, there is no need to change any of these parameters. However, the Reseter Module has a parameter labeled ‘setsToKeep’ that was very helpful for this research. The user can specify annotation sets to keep. The user must select the corpus to be executed. A user can create many different

corpuses and applications. Each application can process any corpus. The user can start the execution processing by hitting the ‘Run this Application’ button.

3.7.3 Application Output

To check process results, the user can open the corpus and examine the annotations added to the documents. Created annotations are added to a default set that has no name. To see the annotations in the text, the user must turn the annotations layer on by clicking the ‘Annotation Sets’ button at the top of the page. The user can enable any of the annotations from the list using the checkmarks next to the label. The text will be colored based on the color of the corresponding label (See Figure 3.16).

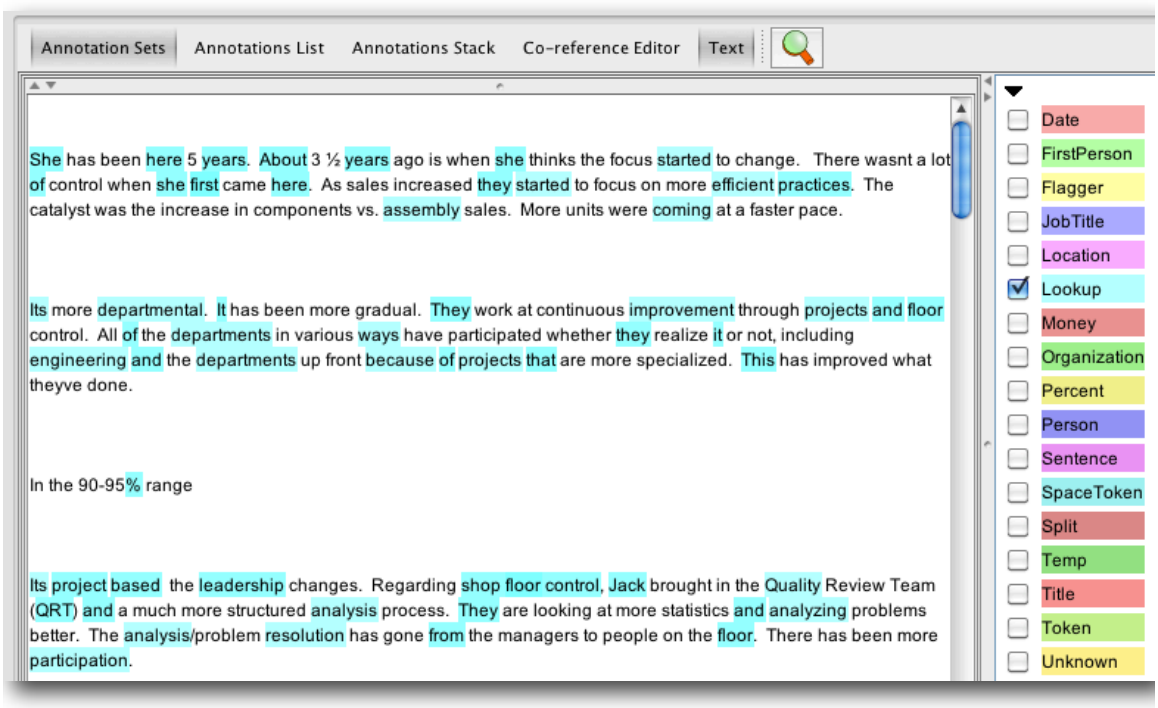


Figure 3.16: GATE Document Editor Interface

To see detailed information about any annotation tag, the user must click on ‘Annotations List’ button at the top of the page. This will open a new pane at the bottom of the page that contains detailed information such as annotation type, start and end points of the tag, and values for annotation features.

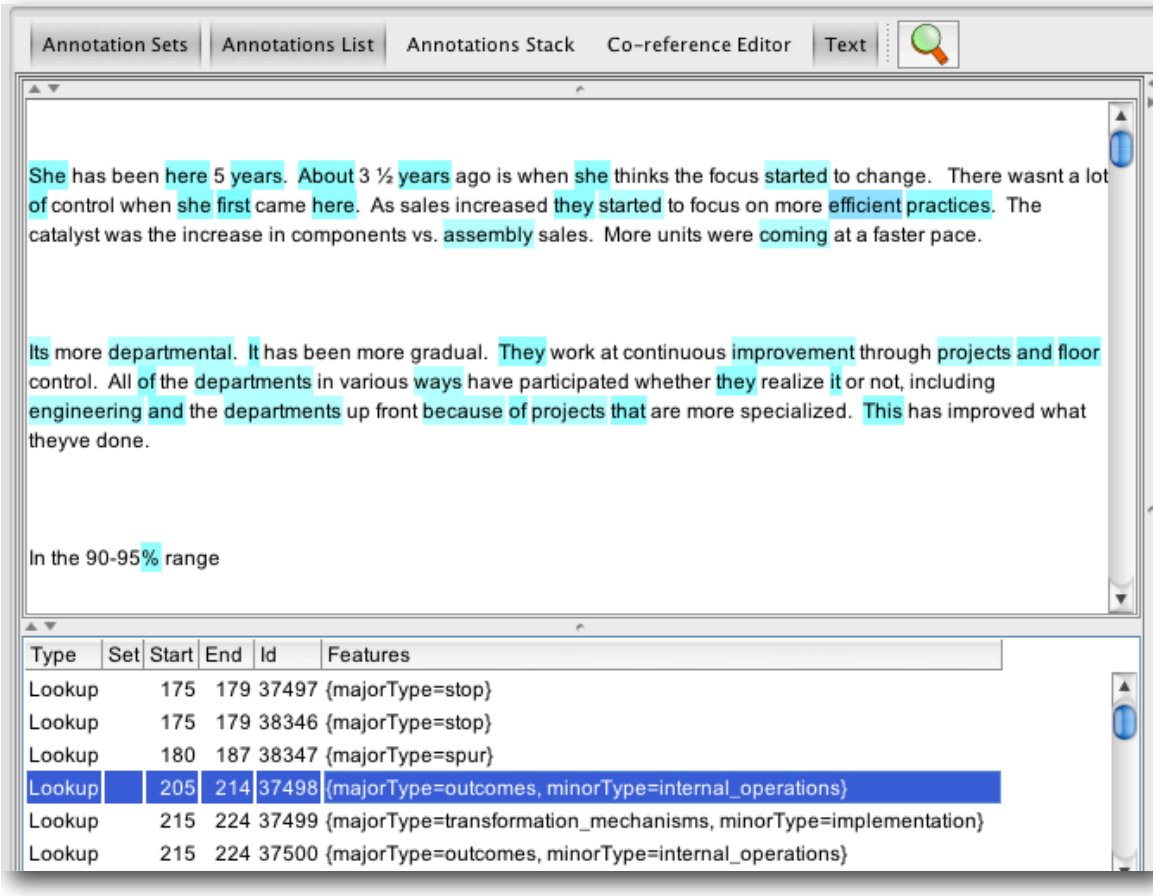


Figure 3.17: An Example of Annotations Generated by Gazetteer

Figure 3.17 shows an example of an annotated word. The word ‘efficient’ is tagged with Lookup annotation which means the word was identified from the Gazetteer list. The annotation string started from character number 205 and ended at the character

number 214. Because the word is annotated by the Gazetteer, it has two features, majorType value ‘outcomes’ and minorType value ‘internal_operations’.

3.8 Output Exporting and Printing

Unfortunately, the GATE platform is not equipped with any easy mechanism for printing or exporting annotated text. The latest release of GATE lets users save documents with annotation tags in XML format. This type of output is called ‘Preserving Format.’ To save a document in the preserving format the user (1) opens the document in the GATE document viewer (i.e. double click on the document), (2) selects annotations of interest by selecting the checkmarks in the Annotations List, and (3) right clicks on the document name in the left pane, and selects ‘Save Preserving Format.’

The document can be saved in any location on the hard disk specified by the user. The file can be opened using a standard text editor such as notepad in Microsoft Windows or VIM in UNIX. Figure 3.18 illustrates a paragraph from a document saved in preserving format.

People don't see the value of barcoding she thinks its a subversive part of their personality the operators think they might be measured on piece work this may not be conscious. Bar coding is <Flagger>one step away from a time card it goes into the company culture. This</Flagger> is a less disciplined environment. Discipline here gets fought every inch. Many still want to operate out of a job shop / garage shop mentality, that people on the floor are direct labor. Discipline is not supported by the supervisors on the floor. There is more discipline now with <Flagger>just Jack and Chris (manufacturing manager and hand adds supervisor</Flagger>) and the leads. The former production manager affected the culture.

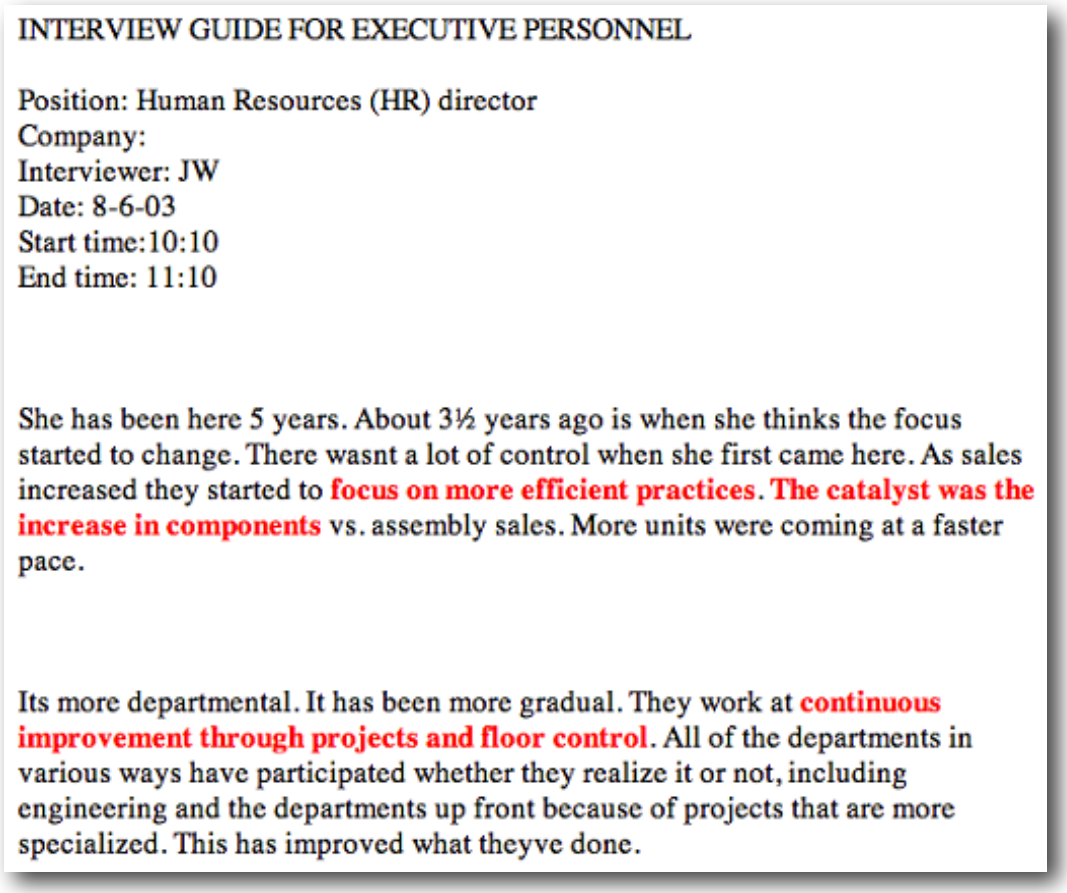
Figure 3.18: A File Saved in Preserving Format Containing Flagger Annotations

The segments of the text annotated under the set ‘Flagger’ are specified using the xml markup format. ‘<Flagger>’ shows the start point and ‘</Flagger>’ shows the end point of each Flagger annotation instance.

This representation is not clear or intuitive enough for viewing or printing. Therefore, for this research, a supplementary application was developed using JAVA. The application, called ‘Output Builder’, operates as a stand-alone application. This application transforms the file from a preserving XML format into a standard HTML webpage. The source code for ‘Output Builder’ is included in Appendix D.

Output Builder can process more than one file at a time. File locations i.e. file paths, are passed to the application as runtime arguments. The application opens a binary data stream for each file and looks for XML annotation markups. XML annotation markups are replaced by HTML tags representing the annotation with different color and font formats. Moreover, necessary lines of code are added to the start and the end of the file to create the necessary body of the HTML format.

The transformation to HTML allows the user to customize the output to a very high level. Also, the user is able to review the documents in any environment such as the user’s favorite text editor or a web browser, print them, or upload them on a web server, which makes them accessible through the internet. Figure 3.19 shows an example of a document processed by the Output Builder. In this example, annotations are highlighted with bolded red font.



INTERVIEW GUIDE FOR EXECUTIVE PERSONNEL

Position: Human Resources (HR) director
Company:
Interviewer: JW
Date: 8-6-03
Start time:10:10
End time: 11:10

She has been here 5 years. About 3½ years ago is when she thinks the focus started to change. There wasnt a lot of control when she first came here. As sales increased they started to **focus on more efficient practices**. **The catalyst was the increase in components** vs. assembly sales. More units were coming at a faster pace.

Its more departmental. It has been more gradual. They work at **continuous improvement through projects and floor control**. All of the departments in various ways have participated whether they realize it or not, including engineering and the departments up front because of projects that are more specialized. This has improved what theyve done.

Figure 3.19: Output Builder Generated Document Example

3.9 Automatic Summarizer Application

The JAPE Transducer was introduced before as a semantic tagger module available in GATE platform. The JAPE Transducer deploys Java Annotation Pattern Engine (JAPE) and uses existing annotation sets in a document to add new annotations and features to the text.

Like any other processing resource, the JAPE Transducer must be added to the application pipeline. The transducer is available in the default ANNIE system and can be loaded through the new processing resource menu (i.e. right clicking on processing resources and choosing new JAPE Transducer). Each JAPE Transducer instance must be

linked to a grammar file. The JAPE Transducer has an init-time parameter for the JAPE grammar file URL (URL specifies the location that the grammar file is located which can be either on a network or a local drive). Transducer grammars are written as files with the extension 'JAPE' and are compiled at run time and executed over corpus documents. By serializing more than one transducer, the outcome produced by a transducer can be used as an input for the next transducer in the queue.

Since the JAPE Transducer uses the outcome from the application, it must be located at the end of the application pipeline. Once the transducer is initialized, further modifications in the grammar file do not affect the grammars loaded into the transducer. Therefore, if a grammar file is modified, the user must reinitialize the transducer with the updated grammar file.

In this project, the JAPE Transducer was used to create an automatic summarization application that identified the interesting parts of the text and then annotates them under a new annotation set named 'Flagger.' This application used the concept of relative word distance. This concept identified instances where more than one word or phrase from the Gazetteer lists were recognized in relatively close proximity. The distance between two specific words was defined as the number of words located between those two words. To find the best pattern, a number of JAPE grammar files were created. Although all the files implemented the same idea of relative word distance, each of them used a different approach for defining the selection pattern.

3.9.1 Manual Text Summarization

To evaluate the performance of the summarizer module, a reference set must be developed. This reference set must contain the actual corpus with manual summarization annotations. The computer generated summaries are compared to the reference summaries.

Reference summaries were generated by content experts. Two members manually summarized a hand copy of the entire corpus. After reaching a state of agreement between the two researchers, the results were entered into the GATE platform. A new annotation set named ‘MySet’ was added to the system to distinguish between computer generated annotations and user-added annotations. Again, summaries were annotated under the same annotation name ‘Flagger.’ Storing the summary annotations with a single name in both the default set and user defined-set allow the AnnotationDiff tool to compare the two sets.

3.9.2 Automatic Text Summarization Algorithms

A number of different summarization strategies were developed for this research. These strategies implemented an extractive summarization approach using the keywords method (Edmundson, 1998) and were based on the concept of relative word distance. For this purpose, relative distance for two words (or more specifically two tokens) was defined as the number of tokens located between two reference words.

Figure 3.20 demonstrates a schematic representation of the summarization algorithms used in this study. The concept of relative distance along with the produced ‘Lookup’ annotations were used to create a set of patterns. These patterns were used to identify interesting parts of the text. The algorithm simply looked for areas of the text that contained a number of ‘interesting words’ and were located in a relatively close distance. In other words, if a part of a text matched the presented pattern in the summarization algorithm, it was annotated as important content and was included in the generated summary.

By taking the advantages of JAPE Grammars, it is possible to code summarization patterns into a set of rules. These rules are written in JAPE syntax and are stored as JAPE grammar files on the disk. By adding a ‘JAPE Transducer’ instance to the end of the

application pipeline and loading the JAPE rule files into the transducer, the framework was able to identify the important parts of the text containing patterns that represent what was important from user's point of view. Interesting parts of the text that were captured by the pattern (i.e. the JAPE rule) were annotated under the name 'Flagger.'

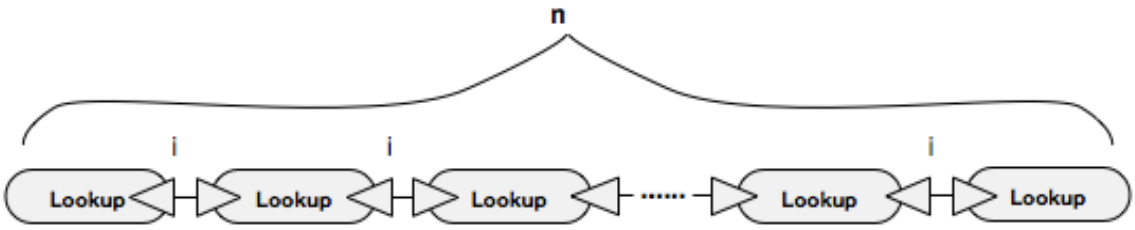


Figure 3.20: Summarization Algorithm Schema

The algorithm in its simplest state has two setup parameters. The first parameter (n) defines the number of lookup instances from Gazetteer lists located in the pattern string, and the second parameter (i) defines the maximum relative distance between the lookup instances. Due to computational resource limitations and to enable the analysis to run faster, only algorithms with similar distances between lookup instances were considered. In other words, if an algorithm is defined with 'i = 5', then the distance between all 'n' lookup instances was set to be less than or equal to five.

The challenge of finding an effective and efficient summarization algorithm is to find the answers for two specific questions: 1) how to handle the Lookup term in the pattern and 2) what numbers should be assigned for 'n' and 'i'? In the next section, the performance evaluation process used to measure the quality of the summaries produced for the lean manufacturing implementation dataset are discussed.

To address the first question, three different approaches for handling Lookup annotations in the pattern were generated, each of which was saved as an independent

JAPE grammar file. The first approach, which is the simplest form, considers all generated lookup tags equally. In this approach, Lookup tags consist of words from both platform-provided and user-generated Gazetteer lists. In the second approach, Lookup tags that are generated from user defined Gazetteer lists only are considered. The third approach uses the annotation 'features' of the Lookup tags instead of the Lookup tags themselves. The corresponding JAPE grammars for all algorithms are provided in Appendix E.

For the second question, a sensitivity analysis study (Saltelli et al., 2004) was conducted. In this study, a range of feasible 'n's and 'i's were selected and tested for all developed algorithms to find the best configuration. Since a wide variety of different 'n's and 'i's can be used for the algorithm, an appropriate approach should be taken to guide the direction of the experiments and limit the number of trials.

3.9.3 Summarization Performance Measurements

To measure the accuracy and effectiveness of developed algorithms and to compare them, the AnnotationDiff tool was used. This performance evaluation method is very similar to the approach used by Yu et al. (2007) to evaluate an automatic text summarization system. In this method, a summary generated by experts in the field was compared to the summaries produced by an automatic summarization system. For this evaluation, machine learning performance measures including Precision, Recall, and F-measure were used. There is a reverse relationship between precision and recall, where it is possible to increase one only at the cost of reducing the other. Since the ultimate goal of this framework is to create summary annotations that include all manually entered user summaries (i.e. reference summaries), higher recall is preferred. However, completely ignoring the precision factor can increase the length of the produced summaries, which means retrieving more irrelevant sentences. Thus, to keep the size of generated

summaries reasonable, the precision factor must also considered. For this study, an F-measure that is the weighted average of precision and recall was used, with a concentration on higher recall averages.

The AnnotationDiff tool compares two annotations that have been saved in two different annotation sets. To use the AnnotationDiff tool, a number of inputs must be set in the AnnotationDiff GUI. 'Key and response documents' are used to introduce documents containing reference (manually generated) and produced annotations. 'Key and response set' options are used to enter annotation sets under which the reference and produced annotations are stored. The "Type option" is used to select the annotation existing in both reference and response sets that should be compared. The annotation name 'Flagger' were used for both manual and computer generated summaries and these summaries were stored under different annotation sets.

In this study, AnnotationDiff tool was used to calculate performance measures for all the documents saved in the corpus. The process of calculating performance measures was repeated for all summarization strategies described. Precision, recall, and F-score were calculated and collected for all twenty interview documents stored in the corpus.

A wide range of summarization patterns can be created using different combinations of 'n's and 'i's. As a first step, the combination that produced the highest F-measure for a dataset was selected. Theoretically, algorithm arguments can be set to any number from one to infinity. Using large 'n's and 'i's will result in getting larger F-scores. That is because the system will consider larger sentences and therefore, the possibility of finding a match with a reference sentence will increase. However, increasing the algorithm arguments comes at the price of drastic performance loss. Since it is impossible to examine all 'n' and 'i' combinations one by one, there was a need for a strategy to find the best feasible answer. To design this experiment, a matrix of argument combinations was developed. Figure 3.21 shows a portion of the matrix that was built for

summarization algorithms in the simplest state. This summarization algorithm considered all ‘Lookup’ annotations.

(3,4)	(3,5)	(3,6)	(3,7)	(3,8)	(3,9)	(3,10)
(4,4)	(4,5)	(4,6)	(4,7)	(4,8)	(4,9)	(4,10)
(5,4)	(5,5)	(5,6)	(5,7)	(5,8)	(5,9)	(5,10)
(6,4)	(6,5)	(6,6)	(6,7)	(6,8)	(6,9)	(6,10)
(7,4)	(7,5)	(7,6)	(7,7)	(7,8)	(7,9)	(7,10)

Figure 3.21: Summarization Pattern’s Arguments Matrix

Shaded cells are all possible argument combinations in the format of (n,i). The first observation was conducted with the first combination in upper left corner of the matrix. The combinations were started from (3,4), because this was the minimum number of lookup annotations that the researchers expected to see in the pattern. For each observation, a table similar to Table 3.2 was used to record the resulting performance measures for each document in the corpus.

Table 3.2: Performance Measures Data Collection Table

	Recall	Precision	F-Score
E2			
E3			
E4			
E5			
E6			
E7			
E8			
E9			
L8			
L9			
L19			
L18			
L17			
L16			
L15			
L14			
L13			
L12			
L11			
L10			
Average			

After collecting data for a combination, new observations were scheduled using the combinations from the cells located below and right of the previous cell. The observations were continued for each branch until no improvement was recorded in the F-score. When observations reached a cell that decreased the performance measures or did not increase the performance measures considerably, moving in that direction was stopped and the observations were continued to other open branches. This strategy prevented checking all possible combinations while guaranteeing that continuing observations would keep the performance measures increasing. The process ended when moving to an adjacent cell (i.e. one unit increase of ‘i’ or ‘n’) did not improve the F-score.

Chapter 4 - Results

4.1 Performance Measures Observations

Figure 4.1 shows the overall results for the performance measure observations and comparisons. Arrows in the cells between the combinations shows the direction of performance measures improvement. For example, an arrow from left to right means that using the argument combination in the right cell of the arrow produced higher performance measures than the argument combination located in the cell left to the arrow (or the improvement in performance measures was little in comparison to achieved improvement by following another open branch).

(3,4)	⇒	(3,5)	⇒	(3,6)	⇒	(3,7)	⇐	(3,8)		(3,9)		(3,10)
⇓		⇓		⇓		⇓						⇓
(4,4)	⇒	(4,5)	⇒	(4,6)	⇒	(4,7)	⇒	(4,8)	⇒	(4,9)	⇒	(4,10)
⇓		⇓		⇓		⇓		⇓		⇓		⇓
(5,4)	⇒	(5,5)	⇒	(5,6)	⇒	(5,7)	⇒	(5,8)	⇒	(5,9)	⇒	(5,10)
		⇓		⇑		X		X		X		X
(6,4)		(6,5)	⇒	(6,6)	X	(6,7)	X	(6,8)	X	(6,9)	X	(6,10)
(7,4)		(7,5)		(7,6)		(7,7)		(7,8)		(7,9)		(7,10)

Figure 4.1: Performance Measure Comparisons

In this study, the best performance measures were obtained when 5 lookups were used with a relative distance of 10 tokens. The application was unable to process the corpus when ‘n’ was more than 5 and ‘i’ was greater than 7. Thus the algorithm was blocked in one direction and the observations were continued by increasing the ‘i’. For ‘i’s greater than 10, the performance measures stopped increasing. Therefore, the process was

stopped at the combination (5,10). Performance measures recorded for the optimum combination and its neighbors are provided in Tables 4.1, 4.2, and 4.3.

Table 4.1: Performance Measures for Combination (5,10)

	Recall	Precision	F-Score
E2	0.90	0.80	0.85
E3	0.75	0.41	0.53
E4	0.92	0.43	0.59
E5	0.76	0.73	0.75
E6	0.76	0.57	0.65
E7	0.91	0.67	0.77
E8	1	0.68	0.81
E9	0.96	0.42	0.59
L8	1	0.50	0.67
L9	0.78	0.29	0.42
L19	1	0.55	0.71
L18	1	0.57	0.73
L17	0.67	0.60	0.63
L16	0.92	0.46	0.62
L15	0.75	0.43	0.55
L14	0.76	0.45	0.57
L13	0.80	0.50	0.62
L12	0.80	0.57	0.67
L11	0.91	0.71	0.80
L10	0.67	0.57	0.62
Average	0.851	0.545	0.657

Table 4.2: Performance Measures for Combination (4,10)

	Recall	Precision	F-Score
E2	0.87	0.63	0.73
E3	0.94	0.41	0.57
E4	0.92	0.34	0.50
E5	0.84	0.60	0.70
E6	0.86	0.47	0.61
E7	1	0.58	0.73
E8	1	0.52	0.68
E9	1	0.36	0.53
L8	0.80	0.33	0.47
L9	0.89	0.26	0.40
L19	1	0.43	0.60
L18	0.88	0.39	0.54
L17	0.78	0.58	0.67
L16	0.92	0.38	0.53
L15	0.92	0.41	0.56
L14	0.84	0.38	0.53
L13	1	0.56	0.71
L12	0.93	0.52	0.67
L11	0.86	0.51	0.64
L10	0.83	0.56	0.67
Average	0.904	0.461	0.602

Table 4.3: Performance Measures for Combination (5,7)

	Recall	Precision	F-Score
E2	0.90	0.62	0.74
E3	0.81	0.39	0.53
E4	0.69	0.29	0.41
E5	0.76	0.58	0.66
E6	0.76	0.46	0.57
E7	0.91	0.53	0.67
E8	1	0.57	0.72
E9	0.87	0.32	0.47
L8	1	0.45	0.62
L9	0.78	0.26	0.39
L19	0.67	0.33	0.44
L18	0.88	0.50	0.64
L17	0.56	0.42	0.48
L16	0.85	0.38	0.52
L15	0.75	0.36	0.49
L14	0.88	0.42	0.57
L13	0.8	0.5	0.62
L12	0.6	0.43	0.50
L11	0.91	0.56	0.69
L10	0.83	0.62	0.71
Average	0.810	0.449	0.572

The data for other examined combinations are available in Appendix F. For checking the validity of the combination selection strategy, a number of extra combinations were tested besides the combinations selected through the optimization process (i.e. combinations tested in Figure 4.1). This was to confirm that combinations not located on the optimization path did not produce higher F-scores than combinations located in the test region.

Combinations located at the lower right corner of the matrix produced high recalls, with satisfactory F-scores. Overall, the combination (5,10) produced the highest F-score. However, the combination (4,10) is also a good choice, since it produced higher recalls in comparison to the (5,10) combination and, it took less time to run over the entire corpus.

Researchers and domain experts can consider this tradeoff when selecting running parameters for specific purposes.

To be able to compare results and to illustrate the sensitivity of using different parameters for the summarization pattern, the F-scores observed for a number of combinations are summarized in Figure 4.2 and 4.3. Each line illustrates the F-score for all documents in the corpus for a specific value of n and i. In the legend, the first digit represents the number of Lookups in the pattern (n), and the second digit represents the relative word distance (i), separated by a '-' character.

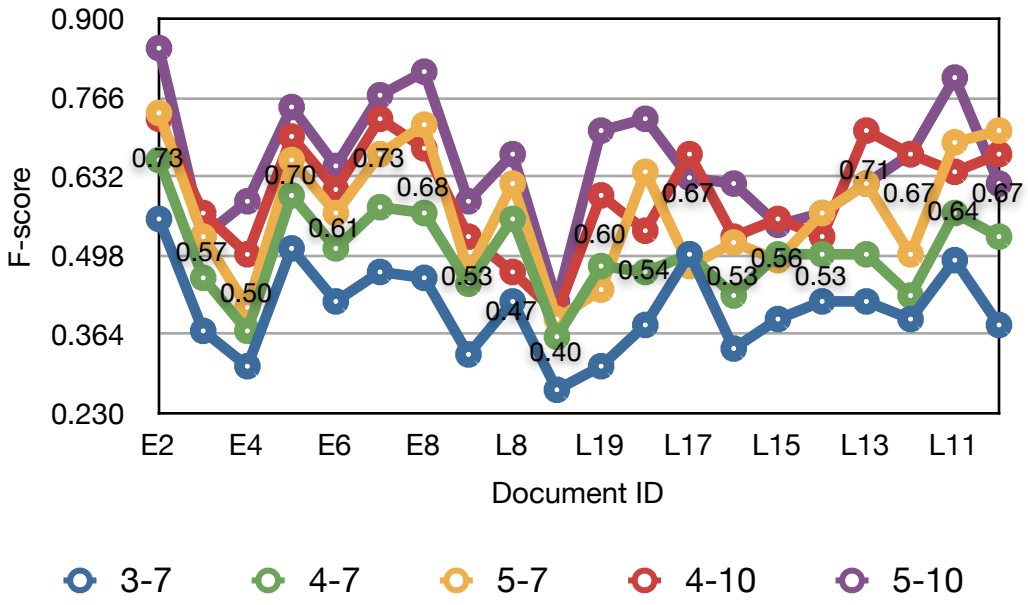


Figure 4.2: F-scores Plot for five High-performance Combinations

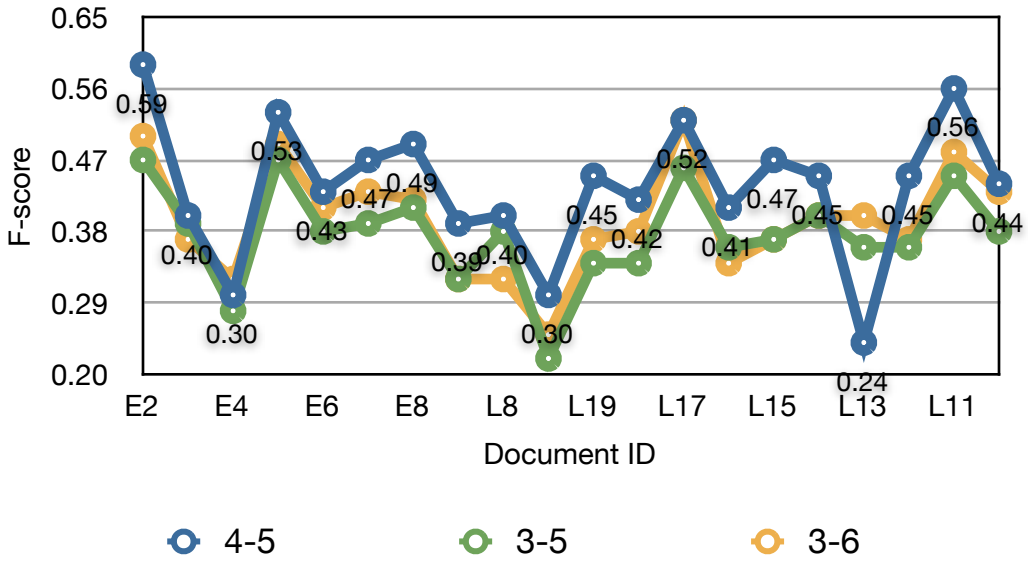


Figure 4.3: F-scores Plot for five Low-performance Combinations

Figure 4.2 illustrates the F-scores for the optimum combination and its neighbors. This plot also supports the statement about the (4,10) combination by showing how close the results are. Figure 4.3 illustrates the measures for combinations located at the upper left corner of the matrix. These combinations were tested at the beginning of the experiment path and produced the lowest performance measures among all other tested combinations.

An effort was initiated to identify different alternatives for summarization pattern other than the one used for the previous observations. In addition to the summarization approach with simple Lookup handling, two other approaches were developed. Tables 4.4, 4.5, and 4.6 and Figure 4.4 summarize the results of using these approaches. Tables 4.4 and 4.5 correspond to the pattern that only considers Lookup annotations generated from user-defined lists and ignores the annotations generated from lists created by the GATE developers. Table 4.6 shows the results for a pattern using Lookup annotation features only. In this approach, ‘Major Types’ along with variables relationship defined in

the performance model were used to detect interesting parts of the text. The JAPE grammar for both approaches are included in Appendix E.

The additional approaches were also tested using the same optimization strategy described in this section. The performance measures observed for these approaches were much lower than the first approach which considered all Lookup annotations in the simplest state. The drastic decrease of the performance measures emphasized the importance of using Gazetteer lists provided by the developers, along with the user defined Gazetteer lists. These comprehensive word lists that contain essential elements of day to day vocabulary such as names, countries, cities, airports, companies, etc. can better connect words and phrases added by the user. In other words, the combination of the initial lists provided by developers with the sets created by the user can add more context to the text, and the system can better detect the important parts of the text.

Table 4.4: Performance Measures for Partial Lookup Combination (5,6)

	Recall	Precision	F-Score
E2	0.32	0.67	0.43
E3	0.12	0.40	0.19
E4	0.15	0.40	0.22
E5	0.16	0.80	0.27
E6	0.10	0.40	0.15
E7	0.41	0.69	0.51
E8	0.31	0.57	0.40
E9	0.30	0.41	0.35
L8			
L9	0.11	0.33	0.17
L19	0.17	0.50	0.25
L18			
L17	0.11	1	0.2
L16	0.15	0.40	0.22
L15	0.08	1	0.15
L14	0.2	0.71	0.31
L13	0.40	0.67	0.50
L12			
L11	0.14	1	0.24
L10			
Average	0.201	0.621	0.285

Table 4.5: Performance Measures for Partial Lookup Combination (4,6)

	Recall	Precision	F-Score
E2	0.25	0.44	0.32
E3	0.39	0.52	0.44
E4	0.23	0.25	0.24
E5	0.28	0.54	0.37
E6	0.14	0.33	0.20
E7	0.55	0.63	0.59
E8	0.46	0.46	0.46
E9	0.39	0.35	0.37
L8			
L9	0.22	0.29	0.25
L19	0.17	0.50	0.25
L18			
L17	0.11	1	0.20
L16	0.15	0.22	0.18
L15	0.08	0.5	0.14
L14	0.32	0.44	0.37
L13	0.40	0.67	0.50
L12	0.20	0.27	0.23
L11	0.27	0.75	0.40
L10	0.17	1	0.29
Average	0.265	0.508	0.322

Table 4.6: Performance Measures for Algorithm Using Lookup Annotation Features

	Recall	Precision	F-Score
E2	0.29	0.69	0.41
E3	0.06	0.33	0.11
E4	0.23	0.5	0.32
E5	0.28	0.78	0.41
E6	0.10	0.67	0.17
E7	0.41	0.69	0.51
E8	0.23	0.50	0.32
E9	0.09	0.67	0.15
L8			
L9	0.11	0.33	0.17
L19	0.17	0.33	0.22
L18	0.12	0.50	0.2
L17	0.11	1	0.2
L16	0.08	0.50	0.13
L15	0.08	0.33	0.13
L14	0.20	0.71	0.31
L13			
L12			
L11	0.32	0.70	0.44
L10	0.17	0.50	0.25
Average	0.179	0.572	0.261

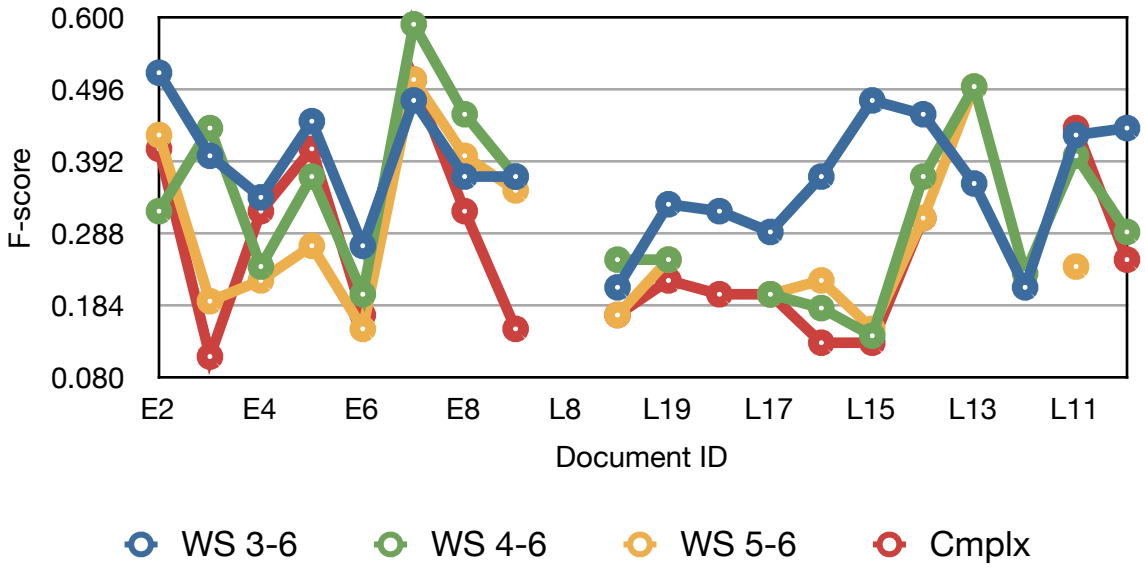


Figure 4.4: Second Level Summarization Algorithms Example

Another interesting fact that was observed is that the average F-score calculated for documents obtained from executive interviews were higher than the F-score calculated for documents obtained from line worker interviews and observations. Two reasons were hypothesized for this difference. First, executives may be more aware of the overall lean implementation processes. Therefore, these interviews were more relevant and were more likely to have addressed topics related to the implementation process. Second, the Gazetteer lists developed for this study were based on a management perspective and as a result contain words and phrases that were more likely to be found in the vocabulary of managers.

4.2 Outcome Validity

To confirm the validity of the models used to analyze the data, a set of assumption checking analyses were conducted. In previous section, the outcome performance for applying different summarization patterns were compared by simply checking the F-score averages calculated for each table of observations. It is necessary to systematically check the comparisons and to test if there is a significant difference in the output performance generated using different summarization patterns. To compare the means of more than two groups of observations, a Factorial Design of Experiments can be used. To do so, a set of single factor experiments were designed. In this design, the treatment factor was defined as the summarization algorithms. To demonstrate the analysis, four different treatments (i.e. argument combinations for the optimum and three adjacent cells from Figure 4.1) were selected. In the example shown below, combinations are selected from the lower right corner of the matrix which produce the highest performance measures. StatGraphics software was used to perform the ANOVA. Since there were 20 documents in the corpus, for each treatment, 20 F-scores were observed.

4.2.1 Analysis of Variance

StatGraphics was used to generate a Box-and-Whisker and a scatter plot. The results showed considerable differences in F-score means calculated by using different argument combinations (Figure 4.5).

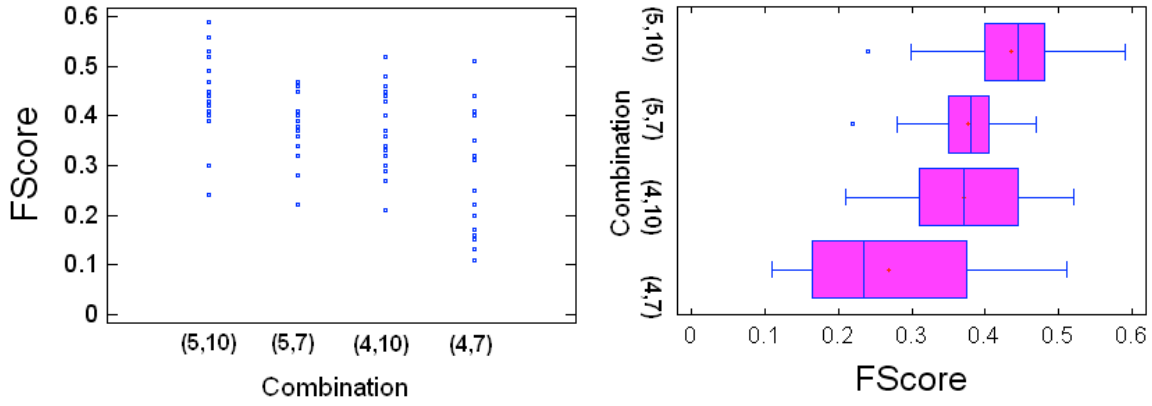


Figure 4.5: Variance Analysis Using Box-and-Whisker and Scatter Plots

Table 4.7 contains the results of the ANOVA generated for the experiment. The p-value is very close to zero, which indicates that there is evidence that a significant difference in F-score means generated using different summarization patterns exists.

Table 4.7: ANOVA Results for comparing F-scores of the four highest combinations

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
Between groups	0.29023	3	0.0967433	11.45	0.0000
Within groups	0.64207	76	0.00844829		
Total (Corr.)	0.9323	79			

The ANOVA table decomposes the variance of observed F-scores into two components: a between-group component and a within-group component. The F-ratio,

which in this case equals 11.45, is a ratio of the between-group estimate to the within-group estimate. Since the p-value of the F-test is less than 0.05, there is a statistically significant difference between the mean F-score from one level of summarization algorithm to another at the 95.0% confidence level.

4.2.2 Residuals Normality Assumption

To validate the result, model adequacy checks were performed. To check the normality assumption for residuals, StatGraphics software was used to fit normal plots on the generated F-scores and the residuals. Figure 4.6 shows the results of fitting a normal distribution to the data on combination (5,10).

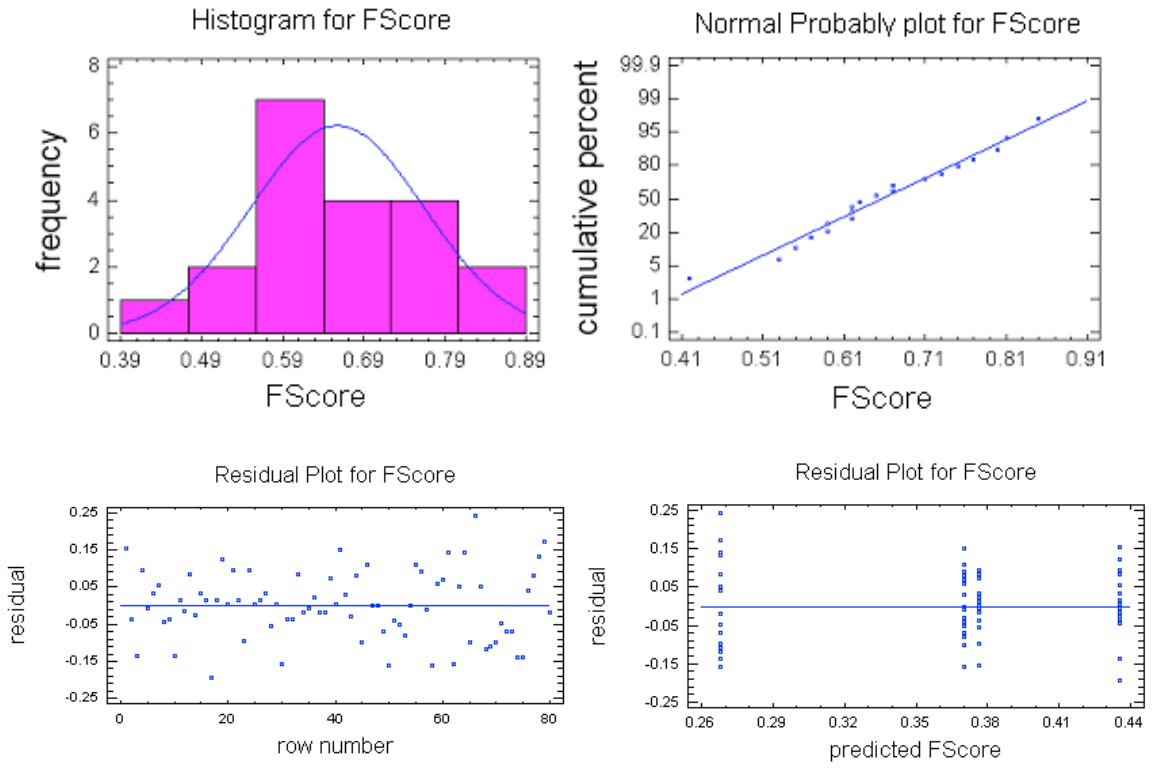


Figure 4.6: Checking Normality Assumption for Residuals at Combination (5,10)

The estimated parameters of the fitted distribution on F-Scores are a mean of 0.6575 and a standard deviation of 0.1066. Table 4.8 shows the results of the Goodness-of-Fit test which shows that the normal distribution fits the data adequately.

Table 4.8: Chi-Square Goodness-of-Fit Test for Combination (5,10)

	Lower Limit	Upper Limit	Observed Frequency	Expected Frequency	Chi-Square
	at or below	0.554309	3	3.33	0.03
	0.554309	0.611556	3	3.33	0.03
	0.611556	0.6575	5	3.33	0.83
	0.6575	0.70344	2	3.33	0.53
	0.703444	0.760691	3	3.33	0.03
above	0.760691		4	3.33	0.13
Chi-Square = 1.59997 with 3 d.f. P-Value = 0.659394					

4.2.3 Constant Variance Assumption

The next assumption that was checked was the constant variance assumption. A number of different tests are available to check this assumption and the results are summarized in Table 4.9.

Table 4.9: Constant Variance Assumption Check for the Combination (5,10)

Cochran's C test: 0.366955	P-Value = 1.0
Bartlett's test: 1.00821	P-Value = 0.796311
Hartley's test: 1.33309	
Levene's test: 0.280619	P-Value = 0.756353

The four statistics displayed test the null hypothesis that the standard deviations of F-scores within each of the four levels are the same. Since the smallest p-value is greater than or equal to 0.05, there is not a statistically significant difference of variance among the observations at the 95.0% confidence level.

4.2.4 Independence Assumption

The last assumption is the independence of the observations. Since, the interviewees were selected randomly and the interview sessions were conducted independently, the independence assumption is valid.

Chapter 5 - Discussion and Future Research

5.1 Discussion

A computer assisted framework was developed to help researchers in conducting qualitative data analysis. This framework leveraged the GATE platform to develop an automatic text summarization system. A dataset from a previous research study of lean implementation practices were used to conduct this development and testing. This dataset was used to evaluate the performance of the developed summarization system.

A lean performance model, developed from the literature on lean, continuous improvement, and manufacturing strategy was converted to an ontology to better suit the requirements of this study. Using the variables defined in the ontology, Gazetteer lists for lean implementation practices were developed. Gazetteer lists are a lexicon database that contain key words and phrases representing major concepts of a domain. The tools and features available in the GATE platform were used to develop an application for processing text. Natural language processing and text mining techniques were used to perform automatic annotations (i.e. automatic coding) based on the developed ontology.

A number of different summarization techniques were developed and tested to select the most appropriate summarization algorithm. The study framework was based on a customized sensitivity analysis and conducted using a factorial design of experiments. A customized sensitivity analysis method was developed and used to systematically perform summarization algorithms comparisons. This method guided the experiments in a direction that prevented running redundant experiments and improved performance measures. Performance measures for different algorithms were compared using both average comparisons and Analysis of Variance (ANOVA). Different summarization algorithms were used as different treatment levels with the average F-score values used as the response variable.

Three classes of summarization algorithms were tested on the data set and based on average calculated performance measures, the most appropriate class was selected. For the best summarization algorithm, an average F-score of 0.657 was recorded. This F-score was based on a recall of 0.851 and a precision of 0.546. According to the literature on machine learning, this F-score is high and has an acceptable level. This performance was obtained for a summarization algorithm in the simplest state of annotation handling. For this algorithm a combination of five Lookup tags and a relative word distance of ten was used. The second best result was obtained for the algorithm with a combination of four Lookup tags and a relative distance of ten. The performance measures were recorded as 0.602, 0.904, and 0.461 for F-score, recall, and precision, respectively.

5.2 Future Work

This research can be further extended to investigate the effects of using other lean implementation performance models or different summarization algorithms on the execution of the summarizer. Selection of different lean implementation performance model would affect the performance of the developed system, as the model is the basis for the development of Gazetteer lists. The implementation of different summarization algorithms could also alter performance measures (i.e. F-score, precision, and recall). One potential area for work is the development of intelligent summarization patterns. For example, in the algorithms developed for this study, only generated annotations were considered in summarization patterns, without taking parts of speech into account. However, by expecting annotation tags in specific places of sentences, the algorithm might be able to better identify important regions and subsequently capture more fluent segments.

Content aware text analysis and machine learning are other areas of extension that could potentially improve the accuracy of the automatic summarizer. There are many opportunities for the developed framework to benefit from machine learning techniques.

These opportunities include incorporating text classification, chunk recognition (information extraction), and relation extraction. For example, by using machine learning techniques, the framework could classify documents based on the content (e.g. labeling a newspaper's articles based on their topics such as news, sport, business, travel, etc.), find relevant documents, and update the summarization and chunk recognition patterns by learning from dataset (e.g. learning important words or phrases by finding new words that are frequently repeated is a dataset).

5.3 Conclusion

Research findings in the areas of computer science, machine learning, and data mining offer a large number of automatic data analysis tools and techniques that other disciplines can benefit from. Research studies completed in engineering management domain have a great potential to employ such tools and techniques to increase the efficiency and effectiveness of data analysis during a research study. This research study demonstrated one of those potential areas in the sub-discipline of engineering management. The framework developed for this study can modify the classical procedure of conducting a qualitative study. The results of this study showed that the qualitative data collected during a lean implementation research study, can be analyzed, annotated, and summarized using computer software. Using this framework at early stages of a qualitative study has a great potential to save researchers time by reducing the time spent on reading and annotating large datasets. The lexicon dataset developed for this study prevents researchers from starting from the scratch in related research studies. Instead, the extra time resulting from this saving can be used on the development of more accurate data models and deeper interpretation of the results.

Bibliography

Afantenos, S., Karkaletsis, V., & Stamatopoulos, P. (2005), Summarization from medical documents: a survey, Artificial Intelligence in Medicine, Elsevier, 33(2), 157-177.

Alavi, S. (2003). Leaning the right way. Manufacturing Engineer, 82(3), 32-35.

Allen, J. F. (2003). Natural language processing, Encyclopedia of Computer Science, John Wiley and Sons Ltd, 1218-1222.

Allen, J. H. (2000). Make lean manufacturing work for you. Manufacturing Engineering, 124(6), 54-61.

Amini, M. R., Usunier, N. & Gallinari, P. (2005). Automatic text summarization based on word-clusters and ranking algorithms, Proceedings of the 27th European Conference on Information Retrieval, Springer, 142-156.

Auerbach, C. F. and Silverstein, L. B. (2003) Qualitative data: An introduction to coding and analysis, NYU press.

Bamber, L., & Dale, B. G. (2000). Lean production: A study of application in a traditional manufacturing environment. Production planing & control, 11(3), 291-298.

Barry, C. A. (1998). Choosing Qualitative Data Analysis Software: Atlas.ti and Nudist Compared, Sociological Research Online, 3(3).

Borst, W. N. (1997). Construction of engineering ontologies for knowledge sharing and reuse, Universiteit Twente.

Boyer, M., & Sovilla, L. (2003). How to identify and remove the barriers for a successful lean implementation. Journal of Ship Production, 19(2), 116-120.

Bransford-Koons, G. R. (2005). DYNAMIC SEMANTIC ANNOTATION OF CALIFORNIA CASE LAW (Doctoral Dissertation), San Diego State University.

Cohen, A. M., & Hersh, W. R. (2005). A survey of current work in biomedical text mining, Briefings in Bioinformatics, 6(1), P. 57.

Crittenden K. S., & Hill, R. J., (1971). Coding Reliability and Validity of Interview Data, American Sociological Review, 36(6), 1073-1080.

Cunningham, H. & Crawford, M. (2005). Gate information extraction. Retrieved December 20, 2004 from http://gate.ac.uk/ie/index.html.

Cunningham, H., Maynard, D., Bontcheva, K., & Tablan, V. (2002). Gate: A framework and graphical development environment for robust NLP tools and applications. In Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics.

Cunningham, H., Maynard, D., Bontcheva, K., & Tablan, V. (2002). GATE: an architecture for development of robust HLT applications, Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, July, 7-12.

Cunningham, H., Maynard, D., & Tablan, V. (2000) Jape: a java annotation patterns engine, Advances in Text Processing, TIPSTER Program Phase II, 185-189.

Doolen, T. L., Hacker, M. E., &Van Aken, E. M. (2003). The impact of organizational context on work team effectiveness: A study of production team, IEEE Transactions on Engineering Management, 50(3), 285-296.

Edmundson, H. P. (1998). New Methods in Automatic Extracting, Journal of the ACM (JACM), 16(2), 264-285.

Erdmann, M., Maedche, A., Schnurr, H. P., & Staab, S. (2000) From manual to semi-automatic semantic annotation: About ontology-based text annotation tools, COLING 2000, Citeseer.

Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., & others (1996). From data mining to knowledge discovery in databases, Communications of the ACM, 39(11), 24-26.

Filson, A., & Lewis, A. (2000). Cultural issues in implementing changes to new product development process in a small to medium sized enterprise (SME). Journal of Engineering Design, 11(2), 149-157.

Gallupe, R. B., Dennis, A. R., Cooper, W. H., Valacich, J. S., Bastianutti, L. M., & Nunamaker Jr, J. F. (1992). Electronic brainstorming and group size, The Academy of Management Journal, 35(2), 350-369.

Gorden, R. (1992), Basic Interviewing Skills, Itasca, IL: F. E. Peacock.

Hand, D. J., Mannila, H., & Smyth, P. (2001). Principles of data mining, The MIT Press.

Hayes, R. H., & Wheelwright, S. C. (1984). Restoring our competitive Edge, New York, NY, John Wiley and sons.

Hayes, R. H., Wheelwright, S. C., & Clark, K. B. (1988). Dynamic Manufacturing: Creating the Learning Organization, New York, NY, The Free Press.

Hirschman, L., Brown, E., Chinchor, N., Douthat, A., Ferro, L., Grishman, R., Robinson, P., & Sundheim, B. (1999). Event99: A proposed event indexing task for broadcast news, Broadcast News Workshop'99 Proceedings, Morgan Kaufmann.

Holden, E. (2003). Lean in the right places. IEE Manufacturing Engineer, 82(3), 18-20.

Ittycheriah, A., Franz, M., Zhu, W.J., Ratnaparkhi, A., & Mammone, R. J. (2001). IBM's statistical question answering system, Nist Special Publication, National Institute Of Standards & Technology, 229-234.

Jing, H. (2000). Sentence reduction for automatic text summarization, Proceedings of the 6th Applied Natural Language Processing Conference, 310-315.

Joachims, T., Nedellec, C., & Rouveirol, C. (1998). Text categorization with support vector machines: learning with many relevant, Machine Learning: ECML-98 10th European Conference on Machine Learning, Chemnitz, Germany, Springer, 137-142.

Joshi, A. K. (1991). Natural language processing, Science, AAAS, 253(5025), P. 1242.

Kaplan, R. S., & Norton, D. P. (1992). The balanced scorecard - Measures that drive performance, Harvard business review, January-February, 71-79.

Keim, D. A. (2002) Information Visualization and Visual Data Mining, IEEE Transactions On Visualization And Computer Graphics, 8(1), 1-8.

Kentne, T. & Maynard, D. (2005). Using gate as an annotation tool. Retrieved December 31, 2009 from http://gate.ac.uk/sale/am/annotationmanual.pdf.

Kim, J. D., Ohta, T., Tateisi, Y., & Tsujii, J. (2003). GENIA corpus-a semantically annotated corpus for bio-textmining, Bioinformatics-Oxford, Oxford Univ Press, 19 (1), 180-182.

Krafcik, J. F. (1988). Triumph of the lean production system. Sloan Management Review, 30(1), 41-52.

Lewis, R. B. (2004). NVivo 2.0 and ATLAS.ti 5.0: A comparative review of two popular qualitative data-analysis programs, Field Methods, 16(4), P. 439.

Luhn, H. P. (1958). The automatic creation of literature abstracts, IBM Journal of research and development , IBM Corporation, 2(2), 159-165.

Mani, I., & Bloedorn, E. (1998). Machine Learning of Generic and User-Focused Summarization. In Proceedings of the Fifteenth National Conference on AI (AAAI-98), 821-826.

Melamed, I. D., Green, R., & Turian, J. P. (2003). Precision and Recall of Machine Translation, AACL '03: Proceedings of the 2003 Conference of the North American

Chapter of the Association for Computational Linguistics on Human Language Technology, 61-63.

Miles, M. B., & Huberman, A. M. (1994). Qualitative data analysis, Sage Publ.

Miltenburg, J. (1995) Manufacturing Strategy: How to Formulate and Implement a Winning Plan, Portland, OR, Productivity Press.

Mitchell, T. (1997). Machine Learning. McGraw-Hill.

Mitra, M., Singhal, A., & Buckley, C. (2002). Automatic text summarization by paragraph extraction, Compare, Citeseer, P. 26.

Nagao, K., & Hasida, K. (1998). Automatic text summarization based on the Global Document Annotation, Proceedings of the 17th international conference on Computational linguistics, Association for Computational Linguistics Morristown, NJ, USA , 2, 917-921.

Neto, J. L., Freitas, A. A., & Kaestner, C. A. A. (2002). Automatic text summarization using a machine learning approach, Lecture notes in computer science, Springer, 205-215.

Noy, N. F., & McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology, Citeseer.

Price, C., & Spackman, K. (2000). SNOMED clinical terms. BJHC&IM-British Journal of Healthcare Computing & Information Management 17(3): 27-31.

Ravichandran, D., & Hovy, E. (2002). Learning surface text patterns for a question answering system, Proceedings of ACL, 2, 41-47.

Saltelli, A., Chan, K., & Scott, E. M. (2004). and others, Sensitivity analysis, Wiley New York.

Shim, J. C., Dorai, C., & Bolle, R. (1998). Automatic text extraction from video for content-based annotation and retrieval, International conference on pattern recognition, IEEE COMPUTER SOCIETY PRESS, 14, 618-620.

Smeds, R. (1994). International Journal of operations and production management, International Journal of operations and production management, MCB University Press, 14, 66-66.

Sparck-Jones, K. (1999). Automatic summarizing: factors and directions. In Mani, I.; Maybury, M. Advances in Automatic Text Summarization. The MIT Press, 1-12.

Stamatatos, E., Fakotakis, N., & Kokkinakis, G. (2000). Automatic text categorization in terms of genre and author, Computational linguistics, MIT Press, 26(4), 471-495.

Sutton, R. I., & Hargadon, A. (1996). Brainstorming groups in context: Effectiveness in a product design firm, Administrative Science Quarterly, Cornell University, Johnson Graduate School, 41(4).

Tan, A. H. (1999). Text mining: The state of the art and the challenges, Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases, Citeseer, 65-70.

Taylor-Powell, E., & Renner, M. (2003). Analyzing qualitative data, Program Development and Evaluation. UW-extension. Cooperative Extension, Madison WI.

Tesch, R. (1990). Qualitative research: Analysis types and software tools, Routledge.

University of Sheffield Natural Language Processing Group. Gate, a general architecture for text engineering. Retrieved December 20, 2004 from http://gate.ac.uk.

Van Driel, M. A., Bruggeman, J., Vriend, G., Brunner, H.G., & Leunissen, J. A. (2006). A text-mining analysis of the human phenome, European journal of human genetics, 14 (5), 535-542.

Weston, C., Gandell, T., Beauchamp, J., McAlpine, L., Wiseman, C., & Beauchamp, C., (2001). Analyzing interview data: The development and evolution of a coding system, Qualitative Sociology, 24(3), 381-400.

Weitzman, E. A., & Miles, M. B. (1995). Computer programs for qualitative data analysis: A software sourcebook, Sage Publications Thousand Oaks, CA.

Womack, J. P., Jones, D. T., & Roos, D. (1990). The machine that changed the world. New York, New York: 1st Harper Perennial Edition.

Worley, J. M. (2004). The Role of Sociocultural Factors in a Lean Manufacturing Implementation (Master Thesis), Oregon State University.

Worley, J. M., & Doolen, T. L. (2006). The role of communication and management support in a lean manufacturing implementation, Management Decision, 44(2), 228-245, Emerald.

Yu, L., Ma, J., Ren, F., & Kuroiwa, S. (2007). Automatic Text Summarization Based on Lexical Chains and Structural Features, Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing. SNPD 2007. Eighth ACIS International Conference.

Appendices

Appendix A - Manual Coding Example

EXT	CR			Transf.	Internal C.
104	207	One of their customers got them to participate.	E2	+	
104	213	It is customer driven. The 5S activity was very customer focused – they want quicker responses and a better product. A side benefit is they want the customer impressed with the look of the manufacturing floor so they are focusing on cleani up. They have JIT delivery for several customers. They use kanbans that were s up for customers but sales have fallen off so they haven't exercised them as they would like.	E2	+	+
105	227	A large part of it is customer driven.	E3	+	
106	241	I asked E3 about the Kanban system – what had prompted it. E3 said the custom demanded it.	E3	+	
106	246	E3 said the customers are also driving the shortages. They want 500, but will tak 200 now. will put the kit for 500 and split the kit so 200 goes out.	E3		−
134	463	The number of people in Stuffing depends on demand. There are usually 2 peopl but it could be 1-3+.	L10	NA	NA
138	530	Some customers will take an order if it is missing a board. If the customer won't do that, usually Test will try to fix the board unless it's beyond repair.	L12		−
138	532	If it's decided that an item is still under warranty Test will wait for the customer's o.k. to proceed. Sometimes the customer just wants a credit. If that is the case Test will repair the item if it's still good and then re-sell it.	L12		NA
142	557	Sometimes, in order to get the price break for a large quantity, a customer will commit to buying a large amount, but will ask for staggered shipping dates.	L17		−
142	563	They do get forecasts from customers – L17 enters them into MAS 90's MRP so purchasing can see what's coming.	L17		+
142	567	If a customer wants a different carrier for shipping, like a cab or it has to be shippe internationally, Order Entry takes care of it.	L17	NA	NA
142	572	If a price needs to be fixed they will also use a credit memo. The price may be fix because the customer wants a price break for buying a certain quantity.	L17		−
122	313	It's customer and competition driven. If implemented correctly and followed throu it will cut costs and save time.	L3	+	
123	329	To keep costs down, give the company a competitive edge. It makes them more attractive to customers – it makes it more apparent that they are running the business well.	L4	+	

Trans	
+	−
6	0
S = 4	

Internal C	
+	−
2	4
S = 4	

Appendix B - Interview Transcripts Example

Source: E2
Position: Human Resources (HR) director
Company:
Interviewer: JW
Date: 8-6-03
Start time:10:10
End time: 11:10

1. When did your company first start focusing on becoming more efficient? When did actual implementation of more efficient practices first start?

She has been here 5 years. About 3 ½ years ago is when she thinks the focus started to change. There wasn't a lot of control when she first came here. As sales increased they started to focus on more efficient practices. The catalyst was the increase in components vs. assembly sales. More units were coming at a faster pace.

2. Did you begin the manufacturing improvement campaign company wide or did you implement gradually by area? If the campaign was implemented gradually, can you give a schedule of implementation by area or department? What percent of your organization has participated in improvement efforts to date (by production line/cell)?

It's more departmental. It has been more gradual. They work at continuous improvement through projects and floor control. All of the departments in various ways have participated whether they realize it or not, including engineering and the departments up

front because of projects that are more specialized. This has improved what they've done.

3. What percent of your workforce has participated in these efforts? (Does everyone in an area become involved or is the focus more on people in supervisory roles?)

In the 90-95% range

4. Who leads the manufacturing improvement efforts in your company? (a dedicated department, HR, a member of management, etc.?)

It's project based – the leadership changes. Regarding shop floor control, J. brought in the Quality Review Team (QRT) and a much more structured analysis process. They are looking at more statistics and analyzing problems better. The analysis/problem resolution has gone from the managers to people on the floor. There has been more participation.

They are driven by the concept of what quality does. The whole quality team expects operators to look more carefully at what they do. Quality has changed from an inspector to an auditor. This changed the whole concept behind quality review, not just the title. This was supported by IPC training. Operators are expected to inspect their own work. It's more inclusive vs. exclusive as it was before.

Project identification comes from the strategic planning process and the direct reports. Projects are implemented/executed across the organization. The leaders and teams

change depending on the project. It's a dynamic process. They try to get the people affected involved in it.

5. If a department has been created to lead these improvement efforts, what is the make-up of the team? (HR personnel, people from all areas of the company, IEs, MEs, etc.?)

The leads, supervisors, direct reports.

6. How do the personnel leading these improvement efforts learn about efficient/lean practices? (formal training, reading, etc.?)

It is experience related. They also brought in a couple of different kinds of instructors on Apex related topics from [a local community college] over the last three years. Problem resolution and quantitative SPC metric processes were some of the ideas covered.

7. How has the concept of implementing manufacturing improvement practices been introduced to the company? (mandatory participation, voluntary, incentives, etc.?)

When the process is changed significantly there are work instructions or ISO procedures related to it. The ISO process drives it (so it's mandatory) – they say let's try it and see how it works. They then go back and see what works.

8. What are the resources available for implementing new practices? (Budget? Facilitation? Training?)

Most of it falls under ISO process and training requirements. Some of it involves reading and applying the new practice. Much of it is hands on. It is driven by the travelers. Engineering participates in how work flows through the floor.

9. What have typically been the catalysts for change determining the need for new practices? (customer-driven? Competition-driven? Etc.)

Customers and competition. They have external influences and the general economy also driving changes. They are now involved in strategic planning and they are exploring what other area of business they could capitalize on. They are looking into things like nano-technology and lead free manufacturing. They are also looking at how they can change processes to utilize more environmentally friendly materials. The lead free manufacturing idea was generated by the political processes in western Europe. They are starting to see this requirement in global contracts.

10. Does the company have particular goals it is working towards with the implementation of these practices? (such as set-up reduction? Waste reduction?)

If they can be a leader in lead free technology they will have a market edge.

The cost effectiveness of a new practice is looked at. They are very money driven in this type of organization. They are more money driven so they look at how they can differentiate their organization from others so they have some exclusivity.

11. What types of performance measures are typically targeted when implementing a new idea or practice?

It's more informal, they don't measure exactly. It's formal in the sense that they put it in a sales plan to the corporate office.

12. What is the typical format for rolling out a new practice and how long is spent in each portion of the roll out? (e.g., kick-off? Training? Analysis? Designing future state?)

Most of it comes from ISO related. It's generally implemented through that kind of structure. If they are penetrating more of an aerospace industry then the customer requires a job be done to IPC standards and then they have to train people more formally on IPC and NASA certification. They do have some formal in house training – then the travelers are changed to reflect the quality level.

13. How many people are typically involved in an implementation effort?

It depends on the project – it could be one or two people. On average, about 25-30% of the workforce is involved at one time on any project.

14. What is the composition of various teams involved in the implementation of a new practice? (everyone, managers, line, clerical, etc.?)

See q. 13

15. What are some of the practices implemented so far in your company?

The quality team, shop floor control measures are two. Barcoding was a big project.

They also made changes in how the test department operates. They are doing more complex work with fewer people. B. (information technology manager) drove it – coming in with fresh eyes changed how the department operates. The sales team is selling this resource directly. Internally this is viewed more as a professional resource. It has changed processes and attitudes. It has changed people's attitude – they sell the service more in the marketing perspective. This change didn't really get driven out of the strategic plan. It has had a much broader effect as Bernard was give the project to look at.

16. Have certain practices been more successful than others? Do you have a feel as to why some practices performed better than others? Examples?

Barcoding had to be sold several times.

Because they are such a flat organization everyone wears multiple hats. For example, the shop floor control or barcoding projects – since they are so flat it creates an environment where the direct report buys in. There is more involvement since they are so flat so they are much more successful. There was more involvement by everyone in QRT so it was embraced more.

17. Have certain practices not performed as expected? If yes, do you know why the practice didn't succeed in your area?

Barcodes haven't been as successful as they can't get as much quantitative information as they would like and people don't use it correctly. It hasn't been used correctly because there wasn't as much buy in – people using it don't see the value.

In terms of accurate labor reporting it has affected the real labor times. The significance is that the quote data is compromised.

People don't see the value of barcoding – she thinks it's a subversive part of their personality – the operators think they might be measured on piece work – this may not be conscious. Bar coding is one step away from a time card – it goes into the company culture. This is a less disciplined environment. Discipline here gets fought every inch. Many still want to operate out of a job shop/garage shop mentality, that people on the floor are direct labor. Discipline is not supported by the supervisors on the floor. There is more discipline now with just J. and C. (manufacturing manager and hand adds supervisor) and the leads. The former production manager affected the culture.

Because they are a federal subcontractor they must practice EEO and affirmative action. They have to identify if they are underutilized in a specific category such as gender or race.

18. Have you discontinued any practices that were implemented but not mentioned above? If yes, why were the practices discontinued?

N/A

19. What types of objective/measurable benefits/results have you achieved with the practices implemented to date?

With bar coding they have more data than before. With QRT they have quality metrics. Five years ago they had two or three metrics that were tracked so they have a much more quantifiable process overall.

20. What types of non-measurable benefits have you realized as a result of implementing these practices?

Training people is a given – the company has better educated/trained people. They have improved the education of people. This benefits the company and the people – it makes the people more marketable if they choose to leave.

Employees take pride in their work.

21. To what extent are practices related to manufacturing improvement viewed as a success in your organization?

Generally, yes – It's hard right now. It's disappointing as they have done a lot of good things here but they aren't reaping the profits. They have implemented lots of things but they aren't getting profit sharing because of the outside economy. Practices aren't translating to money for people on the floor. In the U.S. culture we measure so much by the money yardstick, but really they have improved processes and they have better trained people now.

22. What are some of the positive or negative issues, previously not mentioned, that you have experienced in the implementation of new practices?

There was some push back regarding the bar codes. The most successful practices occurred where there was organization wide involvement and buy in.

23. Do you have any practices scheduled for implementation? If yes, what do you hope to achieve with these practices?

Depot repair – they are doing this because a customer asked them to do it but they had wanted to look into it anyway. They had discussed this in past strategic planning sessions. They want to do a whole process for a company, not just particular boards. They are trying to sell this service to other companies, too. There is a big learning curve. They haven't done much work in this area. They have done some for Hewlett-Packard and Intel on particular boards, not the whole thing.

Appendix C - Gazetteer Lists

This section contains the complete collection of Gazetteer lists developed for the lean manufacturing performance model.

Assessment
metric
assessment
create measure
evaluate result
track progress
measure
measurement
measuring
analysis
analyze
analyzing
non-measurable benefit
non measurable benefit
non-measurable benefits
quantitative information
quality goal
quality goals
feedback
inspection

Customer Requirements
customer request
requirement
customer need
competitive pressure
expediting
expedite
order size
order quantity
ISO
external influence

Deployment
educate
education
educations
train
training
pilot test
test
mid-managed
mid managed
resistance to change
communicate
communicating
facilitation
process change
change the process
leadership
driving change
driving changes
engineering
lean concept

Economic Conditions
budget
economy
economic
cash flow
of scale

Customer Service
reduced lead time
reduce lead time
reduced lead-time
reduce lead-time
customer service
customer-driven
customer satisfaction
happy customer
customer base
single source supplier
customer
customer satisfaction
meet demand
satisfy demand
QRT

Development
external driver
internal driver
process development
product development
development
develop
technology based
technology-based
innovation
innovate
incremental
radical

Financial
ROI
roi
revenue
budget
expense
inventory turn
profit
cash generation
sale increase
sales increase
sales decrease
sale decrease
limit on resource
limit on resources
limits on resource
limits on resources

Learning Growth
external training
learning & growth
learning and growth
lean is good
lean is bad
safety
employee reduction
workforce reduction
train
skill development
knowledge growth
team development
team skill
empowerment
involvement
engagement
work instruction
non measurable benefit
satisfaction

HR Management
worker skill
staff skill
staff
staffing
line personnel
personnel ratio
union
non-union
non union
work force
workforce
human resource
layoff
multi skill
team work
visual work
zero defect
professional resource
time card
resource constraint
crew
operator

Organization Structure
decision making
level
leveling
hierarchy
centralization
centralize
decentralized
decentralize
decentralization
centralized
cross functional
team work
as a team

Implementation
instruction sheet
assessment sheet
training sheet
top down
top-down
determined plan
implementation
implement
define value
maximize flow
maximum flow
manufacture to
customer need
SOS
sos
SOP
sop
standard work
work cell
practices
practice
operator
floor

Integrate & Sustain
improve
improvement
improvements
sustain
sustaintion
replicate
replication
integrate
integration

Internal Operations
set-up reduction
setup reduction
waste reduction
work well
works well
not working
unstable
un-stable
internal operation
rework
re-work
reduced build
increased quality
high quality
good quality
decreased quality
bad quality
low quality
reduce shortage
lower inventory
reduced inventory
setup time
5S
purchasing
procedure
practices
efficient
inspection
schedule
work instruction
benefit
result
participation
participate
quality
cycle time

Internal Operations
error reduction
reduction in error
DPMO
time issue
minor issue
reduce time
reduce response time
efficiency

Market Demand
market demand
competitiveness
global
location
regional
market leader
market follower
stable demand
variable demand
seasonal demand
high entry barrier
competition-driven
competition driven
external influence
external influences
competition
marketplace
market place

Process Technology
equipment
process complexity
process technology
automate
selector
assembly
assemble

Planning
plan
planning
management commitme
lean strategy
value stream
identify future state
VSM
vsm
top down
top-down
bottom up
bottom-up
management support
particular goal
particular goals
strategic
strategies

Sourcing
supplier
number of
sourcing
supply source
vertical integration
single source
single-source
lower bid
lowest bid
linkage with supplier
horizontal integration

Production Planning
material flow
information system
mix
inventory volume
product volume
inventory management
manage inventory
inventory strategy
financial control
performance measure
goal setting
product variety
product volume
shift change
metric
capacity
kanban
expediting
expedite
information flow
procedure
inspection
work instruction
mechanical assembly
production area
production line
shop floor control
barcoding
barcode
MRP
work flow
scheduling system

Technology Advances
technology
high tech
information technology
material advance
product advance
process development
technology licensing

Appendix D - Output Builder JAVA Code

The JAVA code presented in this Appendix belongs to the Output Builder Application. This application creates a data stream to XML files containing annotations in preserved format. Then, converts them to user-friendly HTML files which are ready for publishing.

```
//Import required IO classes
import java.io.*;

//This class implements the converter module
public class Replacer {


      public static void main(String[] args) {

            //This loop repeats for the number of passed input files
            for (int i=0;i<=args.length-1;i++){
            try
                  {
                        //Instantiate a new file
                  File file = new File(args[i]);
                  BufferedReader reader = new BufferedReader(new
FileReader(file));
                  String line = "", oldtext = "";
                  while((line = reader.readLine()) != null)
                      {
                        //Building the beginning of an HTML file
                      oldtext += line + "<br>\r\n";
                       }
                  reader.close();
                  //Changing the XML tags with HTML tags
                  String newtext = oldtext.replaceAll("<Flagger>",
"<FONT COLOR=\"#ff0000\"><b>");
                  newtext = newtext.replaceAll("</Flagger>", "</b></
FONT>");
                  //Ending the HTML file
                  newtext = "<html>\n<body>\n\n" + newtext + "\n</
body>\n</html>";
                  //Saving the file
```

```
                    FileWriter writer = new FileWriter("file" + (i+1) +
".htm");
                    writer.write(newtext);writer.close();









                    //Printing execution outputs
                    System.out.println(args[i] + ": Transformation is
Complete!");
                }
                 //Error handling segments
                catch (IOException ioe)
                    {
                    ioe.printStackTrace();
                }
            }
        }

}
```

Appendix E - JAPE Grammar Files

First Summarization Algorithm: All Lookup Annotations

This algorithm considers all Lookup annotations generated from gazetteer lists (including developers and user generated lists). ‘n’ and ‘i’ are parameters of the algorithm.

```
Phase: MyRule
Input: Token Lookup Sentence
Options: control = appelt


Rule: SimpleLookup
(
{Lookup}
        ({Token})[0,i]
{Lookup}
        ({Token})[0,i]
.
.
.
({Token})[0,i]
        {Lookup}
                ({Token})[0,i]
        {Lookup}
        )

        :flagsentence -->
                :flagsentence.Flagger = {kind="flags", rule="SimpleLookup"}
```

Second Summarization Algorithm: Lookup Annotations (from User-defined Gazetteer Lists)

This algorithm considers Lookup annotations generated from user-defined gazetteer lists and ignores lists provided by GATE's developers. 'n' and 'i' are parameters of the algorithm.

```
Phase: MyRule
Input: Token Lookup Sentence
Options: control = appelt


Rule: NoStop
(
{Lookup, !Lookup.majorType=="stop"}
        ({Token})[0,i]
{Lookup, !Lookup.majorType=="stop"}
        ({Token})[0,i]
.
.
.
        ({Token})[0,i]
{Lookup, !Lookup.majorType=="stop"}
        ({Token})[0,i]
{Lookup, !Lookup.majorType=="stop"}
)

:flagsentence -->
        :flagsentence.Flagger = {kind="flags", rule="NoStop"}
```

Third Summarization Algorithm: Lookup Annotation Features

This algorithm uses Look annotation features to identify important parts of the sentence. ‘i’ is the parameter of the algorithm. It defines the maximum distance between annotation features in summarization patterns.

```
Phase: MyRule
Input: Token Lookup Sentence
Options: control = appelt


Macro: Space
(({Token})[0,i])

Macro: Outcome
({Lookup.majorType == outcomes})

Macro: Internal
({Lookup.majorType == internal_context})

Macro: External
({Lookup.majorType == external_factors})

Macro: Transf
({Lookup.majorType == transformation_mechanisms})


Rule: Flagger
(
(Transf) (Space) (Outcome)|
(External) (Space) (Transf)|
(Internal) (Space) (Transf)|
(External) (Space) (Internal)|
(Internal) (Space) (Outcome)|
(External) (Space) (Outcome)
)
:flagsentence -->
        :flagsentence.Flagger = {kind="flags", rule="Checkwithin"}
```

Appendix F - Performance Measures Results

1. First Summarization Algorithm: All Lookup Annotations

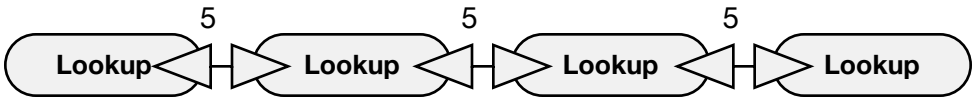
	Recall	Precision	F-Score
E2	0.9	0.32	0.47
E3	0.88	0.25	0.39
E4	0.85	0.17	0.28
E5	0.80	0.33	0.47
E6	0.86	0.24	0.38
E7	0.86	0.25	0.39
E8	1	0.25	0.41
E9	1	0.19	0.32
L8	0.67	0.27	0.38
L9	0.78	0.13	0.22
L19	0.83	0.22	0.34
L18	0.88	0.21	0.34
L17	0.65	0.37	0.46
L16	0.92	0.22	0.36
L15	0.92	0.23	0.37
L14	0.96	0.25	0.40
L13	0.80	0.24	0.36
L12	0.73	0.24	0.36
L11	0.95	0.29	0.45
L10	0.67	0.27	0.38
Average	0.845	0.247	0.376

	Recall	Precision	F-Score
E2	0.94	0.35	0.50
E3	0.88	0.23	0.37
E4	0.92	0.19	0.32
E5	0.84	0.34	0.49
E6	0.86	0.27	0.41
E7	0.91	0.29	0.43
E8	1	0.27	0.42
E9	0.96	0.19	0.32
L8	0.80	0.20	0.32
L9	0.89	0.15	0.25
L19	0.83	0.24	0.37
L18	0.88	0.24	0.38
L17	0.78	0.39	0.52
L16	0.85	0.22	0.34
L15	0.92	0.23	0.37
L14	0.96	0.26	0.40
L13	0.80	0.27	0.40
L12	0.73	0.24	0.37
L11	0.95	0.32	0.48
L10	0.83	0.29	0.43
Average	0.876	0.259	0.394

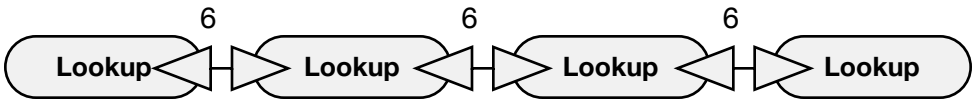
	Recall	Precision	F-Score
E2	0.97	0.39	0.56
E3	0.88	0.23	0.37
E4	0.85	0.19	0.31
E5	0.84	0.37	0.51
E6	0.86	0.28	0.42
E7	0.91	0.31	0.47
E8	1	0.30	0.46
E9	0.96	0.20	0.33
L8	1	0.26	0.42
L9	0.89	0.16	0.27
L19	0.67	0.20	0.31
L18	0.88	0.24	0.38
L17	0.78	0.37	0.50
L16	0.85	0.22	0.34
L15	0.92	0.25	0.39
L14	0.96	0.27	0.42
L13	0.80	0.29	0.42
L12	0.73	0.27	0.39
L11	0.95	0.33	0.49
L10	0.67	0.27	0.38
Average	0.868	0.270	0.407

	Recall	Precision	F-Score
E2	0.97	0.48	0.64
E3	1	0.30	0.46
E4	0.92	0.24	0.39
E5	0.92	0.50	0.65
E6	0.90	0.35	0.51
E7	1	0.41	0.58
E8	1	0.38	0.55
E9	1	0.26	0.41
L8	1	0.31	0.48
L9	0.89	0.18	0.30
L19	1	0.32	0.48
L18	1	0.32	0.48
L17	0.78	0.39	0.52
L16	0.92	0.27	0.41
L15	0.92	0.31	0.46
L14	1	0.32	0.48
L13	1	0.42	0.59
L12	0.87	0.36	0.51
L11	0.95	0.42	0.58
L10	0.83	0.38	0.53
Average	0.943	0.346	0.500

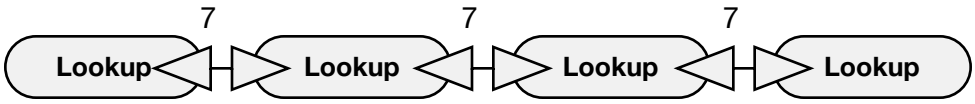
	Recall	Precision	F-Score
E2	0.94	0.43	0.59
E3	0.75	0.27	0.40
E4	0.69	0.19	0.30
E5	0.72	0.42	0.53
E6	0.76	0.30	0.43
E7	0.82	0.33	0.47
E8	0.92	0.33	0.49
E9	0.96	0.25	0.39
L8	0.80	0.27	0.40
L9	0.78	0.18	0.30
L19	0.83	0.31	0.45
L18	0.88	0.28	0.42
L17	0.67	0.43	0.52
L16	0.85	0.27	0.41
L15	0.83	0.32	0.47
L14	0.84	0.31	0.45
L13	0.4	0.17	0.24
L12	0.67	0.34	0.45
L11	0.91	0.40	0.56
L10	0.67	0.33	0.44
Average	0.784	0.306	0.435



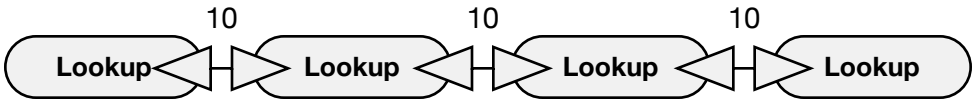
	Recall	Precision	F-Score
E2	0.97	0.47	0.63
E3	0.75	0.31	0.44
E4	0.69	0.20	0.31
E5	0.80	0.48	0.60
E6	0.86	0.37	0.51
E7	0.91	0.43	0.59
E8	0.92	0.36	0.52
E9	0.96	0.25	0.40
L8	1.00	0.38	0.56
L9	0.78	0.18	0.29
L19	0.83	0.31	0.45
L18	0.88	0.29	0.44
L17	0.67	0.38	0.48
L16	0.85	0.28	0.42
L15	0.83	0.29	0.43
L14	0.88	0.34	0.49
L13	0.60	0.27	0.37
L12	0.67	0.31	0.43
L11	0.91	0.39	0.55
L10	0.83	0.42	0.56
Average	0.829	0.335	0.473



	Recall	Precision	F-Score
E2	0.94	0.51	0.66
E3	0.75	0.33	0.46
E4	0.77	0.24	0.37
E5	0.80	0.48	0.60
E6	0.86	0.37	0.51
E7	0.91	0.43	0.58
E8	0.92	0.41	0.57
E9	1	0.29	0.45
L8	1	0.38	0.56
L9	0.89	0.22	0.36
L19	0.83	0.33	0.48
L18	0.88	0.32	0.47
L17	0.67	0.40	0.50
L16	0.85	0.29	0.43
L15	0.92	0.34	0.50
L14	0.88	0.35	0.50
L13	0.80	0.36	0.50
L12	0.67	0.32	0.43
L11	0.91	0.42	0.57
L10	0.83	0.38	0.53
Average	0.854	0.358	0.501



	Recall	Precision	F-Score
E2	0.87	0.63	0.73
E3	0.94	0.41	0.57
E4	0.92	0.34	0.50
E5	0.84	0.60	0.70
E6	0.86	0.47	0.61
E7	1	0.58	0.73
E8	1	0.52	0.68
E9	1	0.36	0.53
L8	0.80	0.33	0.47
L9	0.89	0.26	0.40
L19	1	0.43	0.60
L18	0.88	0.39	0.54
L17	0.78	0.58	0.67
L16	0.92	0.38	0.53
L15	0.92	0.41	0.56
L14	0.84	0.38	0.53
L13	1	0.56	0.71
L12	0.93	0.52	0.67
L11	0.86	0.51	0.64
L10	0.83	0.56	0.67
Average	0.904	0.461	0.602



	Recall	Precision	F-Score
E2	0.97	0.62	0.76
E3	0.75	0.40	0.52
E4	0.69	0.26	0.38
E5	0.72	0.55	0.62
E6	0.67	0.39	0.49
E7	0.91	0.50	0.65
E8	0.85	0.42	0.56
E9	0.87	0.31	0.45
L8	0.80	0.36	0.50
L9	0.78	0.27	0.40
L19	0.50	0.25	0.33
L18	0.75	0.38	0.50
L17	0.56	0.45	0.50
L16	0.85	0.37	0.51
L15	0.67	0.32	0.43
L14	0.76	0.37	0.50
L13	0.60	0.33	0.43
L12	0.53	0.33	0.41
L11	0.91	0.51	0.66
L10	0.67	0.44	0.53
Average	0.740	0.391	0.506

	Recall	Precision	F-Score
E2	0.90	0.62	0.74
E3	0.81	0.39	0.53
E4	0.69	0.29	0.41
E5	0.76	0.58	0.66
E6	0.76	0.46	0.57
E7	0.91	0.53	0.67
E8	1	0.57	0.72
E9	0.87	0.32	0.47
L8	1	0.45	0.62
L9	0.78	0.26	0.39
L19	0.67	0.33	0.44
L18	0.88	0.50	0.64
L17	0.56	0.42	0.48
L16	0.85	0.38	0.52
L15	0.75	0.36	0.49
L14	0.88	0.42	0.57
L13	0.8	0.5	0.62
L12	0.6	0.43	0.50
L11	0.91	0.56	0.69
L10	0.83	0.62	0.71
Average	0.810	0.449	0.572

	Recall	Precision	F-Score
E2	0.90	0.80	0.85
E3	0.75	0.41	0.53
E4	0.92	0.43	0.59
E5	0.76	0.73	0.75
E6	0.76	0.57	0.65
E7	0.91	0.67	0.77
E8	1	0.68	0.81
E9	0.96	0.42	0.59
L8	1	0.50	0.67
L9	0.78	0.29	0.42
L19	1	0.55	0.71
L18	1	0.57	0.73
L17	0.67	0.60	0.63
L16	0.92	0.46	0.62
L15	0.75	0.43	0.55
L14	0.76	0.45	0.57
L13	0.80	0.50	0.62
L12	0.80	0.57	0.67
L11	0.91	0.71	0.80
L10	0.67	0.57	0.62
Average	0.851	0.545	0.657

2. Summarization algorithm based on User-developed Gazetteer Lists

	Recall	Precision	F-Score
E2	0.55	0.49	0.52
E3	0.38	0.43	0.40
E4	0.46	0.27	0.34
E5	0.36	0.60	0.45
E6	0.24	0.31	0.27
E7	0.50	0.46	0.48
E8	0.46	0.32	0.37
E9	0.48	0.30	0.37
L8			
L9	0.22	0.20	0.21
L19	0.33	0.33	0.33
L18	0.38	0.27	0.32
L17	0.22	0.40	0.29
L16	0.38	0.36	0.37
L15	0.42	0.56	0.48
L14	0.44	0.48	0.46
L13	0.40	0.33	0.36
L12	0.20	0.23	0.21
L11	0.36	0.53	0.43
L10	0.33	0.67	0.44
Average	0.374	0.396	0.373

Partial Lookup 3-6

	Recall	Precision	F-Score
E2	0.25	0.44	0.32
E3	0.39	0.52	0.44
E4	0.23	0.25	0.24
E5	0.28	0.54	0.37
E6	0.14	0.33	0.20
E7	0.55	0.63	0.59
E8	0.46	0.46	0.46
E9	0.39	0.35	0.37
L8			
L9	0.22	0.29	0.25
L19	0.17	0.50	0.25
L18			
L17	0.11	1	0.20
L16	0.15	0.22	0.18
L15	0.08	0.5	0.14
L14	0.32	0.44	0.37
L13	0.40	0.67	0.50
L12	0.20	0.27	0.23
L11	0.27	0.75	0.40
L10	0.17	1	0.29
Average	0.265	0.508	0.322

Partial Lookup 4-6

	Recall	Precision	F-Score
E2	0.32	0.67	0.43
E3	0.12	0.40	0.19
E4	0.15	0.40	0.22
E5	0.16	0.80	0.27
E6	0.10	0.40	0.15
E7	0.41	0.69	0.51
E8	0.31	0.57	0.40
E9	0.30	0.41	0.35
L8			
L9	0.11	0.33	0.17
L19	0.17	0.50	0.25
L18			
L17	0.11	1	0.2
L16	0.15	0.40	0.22
L15	0.08	1	0.15
L14	0.2	0.71	0.31
L13	0.40	0.67	0.50
L12			
L11	0.14	1	0.24
L10			
Average	0.201	0.621	0.285

Partial Lookup 5-6

3. Summarization Algorithm based on Lookup Annotation Features

	Recall	Precision	F-Score
E2	0.29	0.69	0.41
E3	0.06	0.33	0.11
E4	0.23	0.5	0.32
E5	0.28	0.78	0.41
E6	0.10	0.67	0.17
E7	0.41	0.69	0.51
E8	0.23	0.50	0.32
E9	0.09	0.67	0.15
L8			
L9	0.11	0.33	0.17
L19	0.17	0.33	0.22
L18	0.12	0.50	0.2
L17	0.11	1	0.2
L16	0.08	0.50	0.13
L15	0.08	0.33	0.13
L14	0.20	0.71	0.31
L13			
L12			
L11	0.32	0.70	0.44
L10	0.17	0.50	0.25
Average	0.179	0.572	0.261

Major Type Classification