Defining patterns in unstructured manifests in a volatile cross-domain environment

Master Thesis

Supervisors for university of Groningen: Prof. Dr. M. Aiello
Prof. Dr. M. Biehl

Supervisors for Logica: Drs. W. Mulder
Drs. A. Stoter

Author: Simon Dalmolen (S1676075)

Date: Wednesday, February 24, 2010

Version: 1.0
Abstract

With a growing amount of data in multiple domains, users have access to loads of data. Retrieving information from the data is a challenge, when data is un/semi-structured. An extra challenge is the relation between data in different domains, given the domain restrictions.

This thesis describes an approach for defining patterns in un/semi-structured data. Finding structures in manifests using genetic computation. Genetic computation is a problem-solving strategy by following biological evolutions. The input is a set of potential solutions to the specific problem. A fitness function is used (a metric function) to select the best candidate solution in addressing the problem.

The current study has taken concepts and methods from Information Retrieval and Text Mining. Each corpus (solution pattern) is represented as a graph. A graph consists of nodes. In the current study, these nodes are regular expressions. By taking the right order of regular expression, a graph is presented as a pattern in the un/semi-structured data.

Finally, a collaborative learning mechanism is introduced to find patterns in un/semi-structured data. The current study shows the result of a prototype of a Multi-Agent System created in a framework called JADE. Agents collaborate to find patterns, and each agent uses different fragments of a priori knowledge to discover the patterns (corpora).

Keywords: agent mining, genetic computation, text mining, multi-agents, graph-based corpora, collaborative environment
Table of Content

Abstract .................................................................................................................................................. i
Foreword ............................................................................................................................................... 3
1. Introduction ...................................................................................................................................... 4
  1.1 Problem description ....................................................................................................................... 5
  1.1.1 Collaborative environment ......................................................................................................... 5
  1.1.2 Domain- and privacy restrictions ............................................................................................... 5
  1.1.3 Retrieving relevant information ................................................................................................. 5
  1.2 Case: Supporting grid management and administration ............................................................... 7
  1.2.1 Problem description .................................................................................................................... 7
  1.3 Research questions ....................................................................................................................... 9
  1.4 Approach ...................................................................................................................................... 10
  1.4.1 Experiment: Corpus creation using genetic algorithms .............................................................. 10
  1.4.2 Prototype: Multi-agent system ................................................................................................... 11
  1.5 Scope ........................................................................................................................................... 12
  1.5.1 Assumptions .............................................................................................................................. 13
1.5 Scope ........................................................................................................................................... 12
  1.5.1 Assumptions .............................................................................................................................. 13
2. Background ..................................................................................................................................... 14
  2.1 Information Retrieval & Extraction ............................................................................................... 14
  2.2 Data mining .................................................................................................................................. 15
  2.3 Text mining ................................................................................................................................... 16
  2.4 Intelligent algorithms .................................................................................................................... 17
  2.5 Multi agents .................................................................................................................................. 18
  2.5.1 Properties of environments ...................................................................................................... 19
  2.6 State of the Art ............................................................................................................................. 20
3. Pre-processing in Information Retrieval .......................................................................................... 23
  3.1 Domain administrators .................................................................................................................. 23
  3.2 Extracting algorithms .................................................................................................................... 23
4  Finding structures in manifests........................................................................................................... 25
   4.1 Genetic computation .......................................................................................................................... 25
   4.2 Experimental setup ............................................................................................................................ 27
      4.2.1 Methodology ............................................................................................................................... 27
      4.2.2 Dataset ......................................................................................................................................... 28
      4.2.3 Fitness function ............................................................................................................................ 29
   4.3 Verification, Validation and Evaluation .............................................................................................. 30
   4.4 Results ................................................................................................................................................ 31
      4.4.1 The results of random mutation .................................................................................................. 31
      4.4.2 The results of optimized mutation .............................................................................................. 34
5  Collaborative environment ...................................................................................................................... 36
   5.1 Multi-Agent architectures .................................................................................................................. 36
   5.2 JADE framework ............................................................................................................................... 38
      5.2.1 FIPA ............................................................................................................................................. 38
      5.2.2 JADE .......................................................................................................................................... 39
   5.3 Coordination ....................................................................................................................................... 40
   5.4 Discussion .......................................................................................................................................... 42
6  ISEA: A prototype for finding structures in manifests using a Multi-Agent system ............................. 44
   6.1 Setup .................................................................................................................................................. 45
   6.2 Results ................................................................................................................................................ 46
7  Discussion & conclusion .............................................................................................................................. 47

References .................................................................................................................................................. 49

Appendix A ................................................................................................................................................ 51
Appendix B ................................................................................................................................................ 52
Appendix C ................................................................................................................................................ 55
Foreword

This thesis was written for my Master degree in Computing Science at the University of Groningen. The research was executed as an internship at Logica in Groningen. One of the research projects of Logica is Collaborative Network Solutions (CNS). CNS is an approach on collaboration between businesses, using services, information sharing and exchange, and system integration. CTIS is one of the implementations of the CNS philosophy. CTIS stands for Collaborative Tracing & Information Service. The research was performed in the context of the CTIS implementation.

I would like to thank the following people, without whose help and support this thesis would not have been possible. First I like to show my gratitude to the people of Logica. My supervisor Arjan Stoter for his suggestions, encouragements and guidance in writing the thesis and approaching the different challenges during the thesis. And my supervisor Wico Mulder for all the input and thoughts about the subject and the everlasting positive energy and motivation. I want to thank my supervisor Marco Aiello from the university of Groningen for his support and advice. And I would like to thank Michael Biehl for his input and willingness to be my second supervisor from the university of Groningen. I would also like to thank Eduard Drent of Logica for his practical support, vision, and help.

Finally I would like to thank my parents, and my girlfriend for their constant support during the time I studied.

De Wilp, January 2010,

Simon Dalmolen
1. Introduction

The web is growing everyday and contains huge amounts of data. Users are provided with many tools for searching relevant information. Keyword searching, topic- and subject browsing, and other techniques can help users to find relevant information quickly. Index search mechanisms allow the user to retrieve a set of relevant documents. Sometimes however these search mechanisms are not sufficient. The amount of available data is increasing rapidly, which makes it difficult for humans to distinguish relevant information. Gaining new knowledge, retrieving the meaning of (partial) text documents and associate it to other knowledge is a major challenge.

The current study focuses on finding useful facts or parts of knowledge in text documents, text databases, log files, and contracts. Techniques from machine learning, data mining, information retrieval (IR), information extraction (IE), natural language processing (NLP), and pattern recognition were explored. These techniques are commonly combined in a research area known as text mining.

Text mining is an solution that allows combination and integration from separated information source. With text mining it is possible to connect previously separated worlds of information.

The web has a huge amount of resources, whereby the resources can be available at anytime. The environment is very volatile, because the content can change (e.g. add, remove or change resources). The web consist of linked content, because of the linking parts its collaborates.

The main focus of this study is:

*Defining patterns in unstructured manifests in a volatile cross-domain environment.*
1.1 Problem description
Finding relevant information in unstructured data is a challenge. The data is unknown in terms of structure and values. The lifecycle of each part of data is in a specific domain, whereby a domain expert is available for a priori knowledge. Domain experts can create structures by hand in the data, however this is a time-consuming job and it is done for one dataset in one domain.

An additional challenge is connecting data from different domains.

1.1.1 Collaborative environment
All the data is (geographically) spread over multiple domains. Whereby the environment consists of more than one domain and they have the intention to collaborate. Users are physically located at different places exchanging knowledge and share information by interacting. Nowadays collaborative environments have the characteristics of being a complex infrastructure, multiple organizational subdomains, information sharing is constrained, heterogeneity, changes, volatile, and dynamic.

1.1.2 Domain- and privacy restrictions
Each domain can have its own privacy restrictions. A domain has its own standards in communication, and interaction, data storage, data structures, and culture. Another item are the privacy restrictions, domains have their own policies and restriction. So not all privacy data can be accessed by foreigners.

1.1.3 Retrieving relevant information
The main problem is retrieving relevant information from multiple domain resources. Figure 1-1 gives an overview of multiple domains with cross-domain information sharing. Each domain consists of multiple manifests, where by these manifest can change from structure in time. Each domain expert tries to create a structure or pattern by hand with his/her a priori domain knowledge. However, this is done by hand. Each domain administrator does this for his domain for fulfilling the goal of retrieving rich information from manifest in a readable structure. When there is the need for collaboration in connecting and coupling two domains for creating a shared conceptualization the domain experts have to perform the job together. By communication and creating a conformity, reducing the noise of interacting, and both physically and virtual different worlds are connected in creating a holistic environment.
Introduction

Figure 1-1 Collaborative environment

**Single domain**
At this moment, a domain administrator creates a structure from a manifest whereby rich information is retrievable. The set of manifest will change over time in structure (volatility), so the domain administrator is adaptive and changes the old structure in a new fit on the changed set of manifest. Al this is done by hand.

**Multiple domains**
Each domain has its own structure, but they want to interact and collaborate with other domains. By creating a contact moment with the responsible domain administrator, they create a confirmation about the relevancies in their structures, so relevant information is retrievable from both worlds.
1.2 Case: Supporting grid management and administration

Grid infrastructures are distributed and dynamic computing environments that are owned and used by a large number of individuals and organizations. In such environments information and computational power are shared, making it a large networked platform where applications are running as services. Operational maintenance of grid infrastructures can be a complex task for humans. There are many grid–monitoring tools available (e.g. Monalisa, Nagios, BDII, RGMA and various dashboard applications). Most of these are designed to be extensible, but limited to single organizational boundaries. Although many administration support-tools exist, interoperability between organizations is still a subject for improvements. The complex and dynamic settings of grid infrastructures requires an intelligent information management system which is not only flexible and extensible, but is also able to relate information from different organizations.

CTIS stands for Collaborative Tracing & Information Service. The CTIS program is a research project of Logica and aims to develop a system of collaborating learning agents that support grid administration. The agents observe data in their local environments and collaborate to realize a coherent global model of job flows. A job is a unit of work that is carried out by the grid. Along this execution, a job and its related data run through a number of nodes of the grid.

One of the fundamental design aspects of CTIS is that it performs in a dynamic, distributed environment, and that while communicating with a variety of existing systems, CTIS is able to learn and find relations between various information sources that are located in multiple organizations.

1.2.1 Problem description

During problem analysis, system administrators combine information from various monitoring tools. Domain administrators try to relate their findings to administrators from different domains. This process involves manual inspection of log files. This is a time-consuming job, which sometimes involves the inference of missing data. The current study aims to find patterns in the observed data, and build models of the structure of information in the log files. By relating information from multiple sources (log files) an automated system should be able to infer missing information (1).

The problem in managing grid infrastructures is that local domain information has no connection to other domains. Manually combining relevant data of two or more domains is a time-consuming job, especially with grid infrastructures having dynamic characteristics and considering the domain privacy aspect.
A job is a computation task launched by a client to be handled by the grid. In the grid, a job is pointed to a resource by scheduler and then executed. However a job can be executed on multiple resources in different domains. In other words, a job is spread through the grid (see figure 1-2).

The task of administrator is to monitor the state of the executed jobs. Administrators find relevant information about job states in log files by using their domain knowledge. Because a job is executed on multiple nodes in the grid, the state of the job has be retrieved from the multiple log files.

Due to domain privacy-restrictions an administrator is restricted to his or her own domain. Therefore, when a job is executed on multiple domains administrators have to collaborate to retrieve the entire job flow of an executed job. Scanning the log files for job data and merging this data with findings of other administrators is currently done by hand. This is a time consuming process that involves challenges, such as accurate domain knowledge, and accurate extraction of relevant data.

Figure 1-2 Job execution on the grid
1.3 Research questions

The main research questions in the current study are:

- How to define patterns in unstructured manifests for Information Retrieval?

- Prototyping a decentralized system for finding patterns in volatile cross-domain environments.
1.4 Approach
The following approach is used to address the research questions. First, state of the art literature research is performed to get an overview of learning mechanisms and agent technology in distributed systems.

1.4.1 Experiment: Corpus creation using genetic algorithms
The first experiment is using or creating an intelligent algorithm for Region of Interest (ROI) extraction. ROI’s are possible parts of the solution domain (the retrievable structure/pattern, see figure 1-1). The algorithm has to have the ability to read manifests, to interpret them, and have a learning function. The result should be a structured set containing regular expressions. The structured set is called a corpus. The regular expressions represent the value data of ROI’s. A ROI in a log file can be a sentence, verb, IP address, date or a combination (see figure Figure 1-3).

The algorithm uses a log file and domain knowledge as input. Domain knowledge contains descriptive attributes e.g. {; ,Time = . * date= }. A descriptive attribute is a data pattern that the agent will search for in the log file. A descriptive attribute is a regular expression. With the manifest and the domain knowledge the algorithm will return a corpus. The corpus represents an information pattern in the log file. Information retrieval is achieved by querying the corpus. These queries return the values of the ROI’s.

Figure 1-3 general ROI Extraction algorithm
Validation of the algorithm will be done using test data. The desired output is set-up by an expert and validated by the expert. In this specific case is assumed that one environment contains multiple log files with only one general structure.

Querying the extracted ROI’s falls within the scope of information retrieval (IR) and information extraction (IE). The common measures of IR will be used, e.g. recall, precision. Chapter 4.2.1. describes the measurement and evaluation process in detail.

1.4.2 Prototype: Multi-agent system
The prototype of the decentralized system was based on agent technology. Agents are described in more detail in 0. In the current study agents are used to collect and retrieve information from manifests. They are equipped with software components to collect data. A multi agent system (MAS) contains multiple agent which can communicate which each other. With a MAS it is possible to deal with domain- and privacy restrictions. Agents in a MAS can cooperate to achieve cooperative learning and distributed problem solving. A MAS can increase its self-efficiency and communicate across domains and work autonomously.

The prototype in the current study was a Multi-Agent System build from an open-source package called JADE described in 5.2.2. The Multi-agent system was merged with the pattern retrieving algorithm. This resulted in an autonomous Multi-agent system with learning capabilities, for automated pattern extraction across domains.

The agents use Regions of Interest (ROI) to analyze manifests. A Region of Interest is a (part of a) pattern or a (part of a) model from a manifest. The agents can communicate with each other via the network. The agents collaborate with other agents; to form an alliance, in order to exchange ROI’s.

By exchanging their knowledge, the agents can learn from one another and can achieve higher levels of data abstraction.
1.5 Scope

The scope in the current study is shown in figure Figure 1-4, every chapter will zoom in to a part of the figure. The scope is retrieving data from manifest(s) using domain knowledge and machine learning techniques. Figure 1-4 describes the various levels of the current study, and their relations. The top level describes the data that was used, in this case manifest containing log data. The second level involved pre-processing. That is, retrieving/creating domain knowledge about the manifests. This domain knowledge was used in the next level by the algorithm to find patterns in the manifest, based on the domain knowledge. Finally, in the level of the Multi-agent system, the found patterns -or ROI’s- were exchanged between agents.

Figure 1-4 Text mining executed by Multi-agents
The research does not involve

- creating a framework for multi-agent systems;
- graphical representation of the job state or graphical user interfaces;
- agent communication techniques;
- security issues;

1.5.1 Assumptions
The research was done with no restrictions to computational power.
The research is not mining relevant data in different resources, finding patterns or structures and try map values, because of the domain restrictions.
2 Background

In the introduction chapter, figure 1-4 describes the scope of this study. It uses several research areas in computing science. This chapter will describe the State of the Art of these areas; Information Retrieval (IR) and Information Extraction (IE) belong to the section of pre-processing. Describing a dynamic en collaborated environment is seen in the Multi-Agent part. Text mining uses intelligent algorithms and has an overlap in the pre-processing section.

2.1 Information Retrieval & Extraction

Information retrieval (IR) and extraction (IE) are concepts that derive from information science. Information retrieval is searching for documents and for information in documents.

In daily life, humans have the desire to locate and obtain information. A human tries to retrieve information from an information system by posing a question or query. Nowadays there is an overload of information available, while humans need only relevant information depending on their desires. Relevance in this context means returning a set of documents that meets the information need.

Information extraction (IE) in computing science means obtaining structured data from an unstructured format. Often the format of structured data is stored in a dictionary or an ontology that defines the terms in a specific domain with their relation to other terms. IE processes each document to extract (find) possible meaningful entities and relationships, to create a corpus. The corpus is a structured format to obtain structured data.

Information retrieval is an old concept. In a physical library books are stored on a shelf in a specific order e.g. per topic and then alphabetic. When a person needs information on a specific topic, he or she can run to the shelf and locate a book that fits the most to his or her needs. With the advent of computers, this principle can also be used by information systems. Well-known information-retrieval systems are search engines on the web. For instance, Google tries to find a set of available documents on the web, using a search phrase. It tries to find matches for the search phrase or parts of it. The pre-processing work for the search engines is the information extraction process; to create order in a chaos of information. Google crawls the web for information, interprets it, and stores in a specific structure so that it can be quickly accessed when users are firing search phrases.
2.2 Data mining

After creating order in “multiple bags of words or a data set” the next aim is “mining” the data for knowledge discovery. Mining data in a **structured** format e.g. multiple databases, or text mining: **how to deal with unstructured data** e.g. natural language documents.

**An extra challenge in mining data is trying to find related data in other resources, and clustering data.**

Data mining aims to find useful patterns in data. A problem for many enterprises is the large availability of rich data. More specific to extract useful information from these large amounts of data. Analyzing (often) large datasets and trying to find trivial (hidden) patterns / relationships is a challenge. With the growing amount of data it is harder to retrieve knowledge from several datasets. Data mining can help to find unsuspected relations between data. Data mining is also known as Knowledge Discovery from Data (KDD). Data mining is used to replace or enhance human intelligence by scanning through massive storehouses of data to discover meaningful new correlations.

Data mining consists of an iterative sequence of the following steps, see figure 2-1 (2):

1. Data cleaning (to remove noise and inconsistent data)
2. Data integration (where multiple data sources may be combined)
3. Data selection (where data relevant to the analysis task are retrieved from the database)
4. Data mining (an essential process where intelligent methods are applied in order to extract data patterns)
5. Pattern evaluation (to identify the truly interesting patterns representing knowledge based on some interestingness measures)
6. Knowledge presentation (where visualization and knowledge representation techniques are used to present the mined knowledge to the user)
The concept of data mining, e.g. finding valuable non-trivial patterns in large collections of data, is not an emerging technology anymore. Multiple companies have developed software for mining data. Application of the software is however far from universal. Only the bigger companies are using the software for appliance on their Business Intelligence (BI) e.g. software like STATISTICA. Data mining is becoming more mature, the techniques are highly developed and much research is performed in this area.

### 2.3 Text mining

Text mining uses the same analysis approach and techniques as data mining. However data mining requires structured data, while text mining aims to discover patterns in unstructured data (3). For commercial use text mining will be the follow-up of data mining. With the growing number of digitized documents and having large text databases, text mining will become increasingly important. Text mining can be a huge benefit for finding relevant and desired text data from unstructured data sources. NACTEM\(^1\) performs research in Text mining and applies the found methods on the MEDLINE data source, it’s a huge database containing medical information.

With text mining, the input will be a text set which can be unstructured or semi structured. For example a text document can have a few structured parts like title, author, publication date, and category. The abstract and content might be unstructured components with a high potential information value. It is hard to retrieve information from those parts with conventional data mining.

Text mining uses unstructured documents as input data. In other words, documents that are hard to interpret in terms of meaning. There are few companies working on profitable applications for text-mining. Because of the challenges involved in working with text and the differences between languages it is a challenge to create a general solution or application. The research area is currently “too young” to deal with all of the aspects of text and natural language processing and linking information to each other. However, the first results are promising and perform well e.g. the work performed by TextKernel\(^2\). (In light of the current study the author visited TextKernel to discuss text mining techniques.) Textkernel is a company specialized in mining data, and is working on text mining with promising results e.g. parsing and finding structures in Curriculum Vitae (C.V.) documents. These C.V.’s are being collect and parsed in a general format for international staffing & recruitment agencies.

---

\(^1\) The National Centre for Text Mining (NaCTeM), www.nactem.ac.uk

\(^2\) http://www.textkernel.com
2.4 Intelligent algorithms

This paragraph begins with a basic explanation of *intelligence* and *learning* in software components.

*Intelligence* - in software is being characterized as:
- adaptability to a new environment or to changes in the current environment;
- capacity for knowledge and the ability to acquire it;
- capacity for reason and abstract thought;
- ability to comprehend relationships;
- ability to evaluate and judge;
- capacity for original and productive thought (4).

*Learning* - is the process of obtaining new knowledge. It results in a better reaction to the same inputs at the next session of operation. It means improvement. It is a step toward adaptation. Learning is an important characteristic of intelligent systems. There are three important approaches to learning:

I. *Learning through examples*. This means learning from trainings data, For instance \((X_i, Y_i)\), where \(X_i\) is a vector from the domain space \(D\) and \(Y_i\) is a vector from the solution space \(S\), \(i = 1, 2, \ldots, n\), are used to train a system to obtain the goal function \(F: D \rightarrow S\). This type of learning is typical for neural networks.

II. *Learning by being told*. This is a direct or indirect implementation of a set of heuristic rules in a system. For example, the heuristic rules to monitor a car can be directly represented as production rules. Or instructions given to a system in a text form by an instructor (written text, speech, natural language) can be transformed into internally represented machine rules.

III. *Learning by doing*. This way of learning means that the system starts with nil or little knowledge. During its functioning it accumulates valuable experience and profits from it, and performs better over time. This method of learning is typical for genetic algorithms.

The first two approaches are top down, because there are many possible solutions available. These approaches can learn from experience or via an instructor (supervised). The third one is a bottom-up strategy, beginning with a little knowledge and try to find the best possible solution. But the final or optimal solution is not known only parts of the solution domain are given.
2.5 Multi agents

In information technology, an agent is an autonomous software component. Autonomous, because it operates without the direct intervention of humans or others and has control over its actions and internal states. It perceives its environment through sensors and acts upon its environment through effectors (6). Agents communicate with other agents. By communication, agents can work together which can result in cooperative problem solving, whereby agents have their own tasks and goals to fulfill in the environment. See figure 2-2 Multi agent system canonical view (7) where the agents interact with each other. And an agent perceives a part of the environment with its actions, an agent can influence the environment (partial).

Multi agent system (MAS) is a multiple coupled (intelligent) (software) agents. Multi agents which interact with one another through communication and are capable of to perceive the environment. Multi agent systems solve problems, which are difficult or impossible for a single agent system.

Figure 2-2 Multi agent system canonical view (7)

The agents’ decision-making process depends on the environment their acting in. Agents can act in an environment, but also affect it. An environment has several properties and can be classified according to these properties.
2.5.1 Properties of environments

- **Accessible vs. inaccessible**
  An accessible environment is when agent has a complete state of the environment. Agents detect all relevant states of the environment to base their decision on. If the environment is highly accessible it’s easier for an agent to make more accurate decisions (accurate, as being related to – or appropriate given the state of the environment). Examples of environments with high inaccessible potential are the internet, and the physical real world.

- **Deterministic vs. non-deterministic**
  A deterministic environment is one in which any action has a single guaranteed effect - there is no uncertainty about the state that will result from performing an action (8).

- **Episodic vs. non-episodic**
  In an episodic environment agent perform in several episodes, without any relevance or relations between the episodes. Actions executed in previous episodes by the agent after acting and perceiving the environment has no relevance in the following (i.e. new) episodes. Agent don’t need the ability to reason ahead, the quality of performance does not depend on previous episodes. In a non-episodic environment however, agents should “think” about possible next steps.

- **Static vs. dynamic**
  Static environment remains unchanged even after actions performed by an agent. Dynamic environments are changing environment and are harder to handle for agents. If an environment is static, but agents can change characteristics of the environment, it is called semi-dynamic.

- **Discrete vs. continuous**
  Discrete environment: there are a fixed, finite number of actions and percepts in it e.g. chess game.

A complex environment for an agent is an environment where the agent possibly can act and affect it, but cannot have complete control over the environment.
2.6 State of the Art
As described in paragraph 0 data mining (DM) is becoming a more mature area. Companies are adopting these concepts and techniques. DM addresses one of the main questions in artificial intelligence; “Where does knowledge come from?” Several studies prove that DM successfully attacks emergent problems such as finding patterns and knowledge in massive data volumes. Witten, Ian H. and Frank, Eibe describe in their *Data Mining, Practical Machine Learning Tools and Techniques* (9) the most techniques and approaches which are adopted in Machine Learning and DM. For instance decision tree learning, bayes learning and rule-based learning.

Text data mining (TM) is a natural extension of DM. The paper, *Untangling text data mining* (10) clarifies the concept and terminology. Kroeze, Jan H., Matthee, Machdel C. and Bothma, Theo J.D. (11) describe in their paper *Differentiating Data- and Text-mining Terminology* a summarized survey over the text mining research field and it’s challenges. This study tries to discover structures in unstructured text environments. They conclude that it has become important to differentiate between advanced information retrieval methods and various text mining approaches to create intelligent text mining instead of real text data mining.

In text mining however, the focus of pre-processing is the identification and extraction of relevant features in documents, where unstructured data is first transformed into a structured intermediate format. To achieve this, text mining exploits techniques and methodologies from the areas of information retrieval, information extraction and corpus-based linguistic techniques (3). A basic element in text mining is the document (3). Describe the document as a unit of textual data that usually, but not necessarily, correlates with some real-world documents, such as a business report, manuscript, or log file.

The current study focuses on finding structures in (semi-) unstructured data, manifests. Of the fact that text mining is very broad and challenging. There is not one solution or study representing the Holy Grail for text mining.

The unstructured data with noise should be processed to find rich or relevant information. There is a research area where algorithms are used to perform graph-based pattern/structure finding in manifests. In these studies, the solution pattern or structure and the input data are represented as a graph. Several studies have shown that these graph-based pattern matching techniques can be very useful for finding structures in semi-structured data. For instance Gallagher, Brian - *A survey on Graph-Based pattern matching* applies a data-graph to data. Representing the input data as a graph can be done in a structural way or a semantic approach. Setting the input data as a graph requires prior knowledge on the semantics of the data. After the input data is described in a graph, a graph-based algorithm can calculate a corpus. The input data and the output corpus are both represented as a graph (12).
Another approach in finding ROI's and structures is done by Conrad, Eric - *Detecting Spam with Genetic Regular Expressions* (13). This study uses a totally different approach in detecting structures and performing it. In this study the author performed spam detection in emails. Using regular expressions as building blocks for the to build corpus. The regular expression consisting of a priori knowledge. This study created a mechanism for automated spam detection. A genetic algorithm (GA) produces a set of regular expressions and fits it onto a email messages, with the evolution function of the GA the set of regular expression will get a fitness score. Running the GA the set of regular expressions will mutate and change in trying to get a higher fit over time. This study uses a corpus-based approach and using the building blocks unstructured files can be interpreted in search of relevant information, in this case spam.

The current study uses the approach of graph based pattern matching. The output the corpus is represented as a directed graph, see figure 2-3. Gallagher, Brian (12) used for the input and output data a graph representation. When all the date is represented as a graph. It takes too much steps in preprocessing in a volatile and changeable environment. Of the fact of the need for domain knowledge in semantics.

Using the methods that are described in the spam detection of Conrad, Eric (13), regular expressions representing spam and the GA computation, the current study aims to find a graph based pattern-matching technique (corpus-based) in a volatile environment where the technique should be adaptive. *This is also the main challenge of the current study finding corpora in a volatile environment.*

Retrieving rich information via a pattern or structure from manifests whereby the manifest structure can be changed. This study tries to find an adaptive way with the help of GA in finding structures of manifest, see figure 2-3 Graph-Based pattern matching. Where the founded pattern / structure is named a corpus. A corpus is used terminology in text mining (see paragraph 1.4.1 for an explanation of Region of Interests).

*Figure 2-3 Graph-Based pattern matching*
An extra challenge in the current study that it performs in a collaborative environment, see figure 2-4.

Mining text in a collaborative environment where the environment is challenging (see paragraph 1.1.1 Collaborative environment).

In computer science, agents can be used in a collaborative environment (7) much research has been performed in Agent technology. This study tries to merge (text) data mining with agent technology. So agent are equipped and can be self-learning, and self-organized in a dynamic environment, these agent are adaptive in finding corpora, learn, and share knowledge in a collaborative environment (14), (15), (16)

The synergy of Agent and Data Mining is promising in the paradigm for dealing with complex environments such as openness, distribution, adaptability, and volatility.
3 Pre-processing in Information Retrieval

This chapter describes how to pre-process manifests, log files and text documents in gaining knowledge about these files. This prior knowledge is named domain knowledge in this study (see the figure in paragraph 1.5).

This domain knowledge is required as input for Intelligent Algorithms. For retrieving domain knowledge there are two approaches, manually or by extracting algorithms.

3.1 Domain administrators
Manually gaining knowledge can be done by asking an expert or administrator in the specific domain. This person can provide the required information about the manifest. Which knowledge is needed and give a priority order on each item.

In this specific case on gaining knowledge and retrieving it via domain experts has some disadvantages. For these experts it’s a time consuming job, providing knowledge to other people or systems. In the case of expert providing to a other person. The domain knowledge should be provided in a consumable format. Where a non domain expert can work with the provide knowledge in retrieving rich information from manifest. In practice the domain knowledge can change, so that’s an extra challenge for the domain experts.

This study assumes that a domain expert provides knowledge in readable machine format as input for finding a corpus (structure) of the manifest. It is important to have a readable Human machine communication. An ontology which is very helpful in creating a compromise and an agreement in communication and the meaning about the given domain knowledge.

3.2 Extracting algorithms
Delegating these processes and responsibilities raises the need for machine-process form. A machine (extracting algorithms) reads a set of manifests and tries to extract domain knowledge. This will be used to find a corpus out of a set of manifests. An advantage that the domain experts have a lower load of work in providing knowledge. Their task can change from an executing task to a controlling one. For checking the outcome of these algorithms and give these algorithms feedback, so in the future unseen data will be handled better. This is only the case in supervised learning, paragraph 0. Another approach can be that domain experts only change the outcome of an domain knowledge gaining algorithm.
In this study the algorithms has to have the ability to extract features from manifest. Commonly used features are: Characters, Words, Terms, and Concepts (3)

Domain knowledge extracting algorithms can be very useful in rapidly changing domains. This is characteristic of a GRID volatility, rapidly changing of a domain and new subscribed domains to the GRID. For the current study, a simple word frequency order algorithm is used. This algorithms read an input file for extracting all the words by frequency. It is also possible to feed the algorithm with an ignore list. The result is a list of all words contained by the input file minus the ignore list and orders by frequency.

For this study (working in a collaborative and volatile environment), it was chosen to use a simple frequency order algorithm. This part of the research does not have the main focus. In finding possible partial domain knowledge. The result of the algorithm can be used as an input for the domain experts, for adding or removing data.

**Regular Expressions**
The domain knowledge is in this particular case represented by regular expressions. A regular expression is very powerful, because they represent a search pattern in a manifest.

This format of domain knowledge is used as input for the genetic algorithms. In the GRID case for analyzing log files key value pairs are used, see table 3-1 regular expressions. Log files have mostly a hierarchical structure containing a lot of key value pairs. Using domain knowledge represented by regular expressions as input for corpus creating algorithms is very powerful. The corpus can be queried and it consists of multiple regular expressions. Indeed regular expressions are search patterns, which can be queried.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value (regular expression)</th>
<th>Log file line example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>TIME\p{Punct}<em>s</em>([\w+\s*:]*))</td>
<td>TIME: Sun Dec 7 04:02:09 2008</td>
</tr>
<tr>
<td>PID</td>
<td>PID\p{Punct}<em>s</em>([\w{1,})</td>
<td>PID: 13048</td>
</tr>
<tr>
<td>IP</td>
<td>(([01]?\d\d?</td>
<td>2[0-4]\d</td>
</tr>
</tbody>
</table>

*Table 3-1 Regular expressions*
4 Finding structures in manifests

After preprocessing (see chapter 3) of the manifests, domain knowledge is available. This domain knowledge is a vocabulary consisting of words, terms and key-value pairs. This chapter describes how domain knowledge is used to find patterns in the manifests. The result is a reoccurring pattern named a corpus. This corpus can then be queried for extracting value information.

The algorithm described in this chapter creates the corpus, using the vocabulary (i.e. domain knowledge) and optimizing the order of used items from the vocabulary. The resulting corpus contains a linked set of vocabulary items. These connected items are represented as a directed graph. The nodes of this graph represent the ROI’s and the edges between the nodes represent sequential relations between the ROI’s. The directed graph is explained in more detail in 4.2.1.

This optimization problem in ordering and linking items of the preprocessed vocabulary is an NP-problem.

4.1 Genetic computation

Definition: A genetic algorithm (GA) is a search technique used in computing to find exact or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination) (17). Genetic algorithms give a potential solution to a specific problem.

A genetic algorithm typically has the following logic (18):
1. Create a population of random chromosomes.
2. Test each chromosome for how well it fits the problem.
3. Assign each chromosome a fitness score.
4. Select the chromosomes with the highest fitness scores and allow them to survive to a next generation.
5. Create a new chromosome by using genes from two parent chromosomes (crossover).
6. Mutate some genes in the new chromosome.

Genetic algorithms require a genetic representation (primitives) of the solution domain in combination with a fitness function for evaluating the solution domain. A chromosome represents a (possible) solution. Each chromosome consists of multiple genes, where the gene value (the allele) is a piece of the solution.

Initialization: create a random generated population of chromosomes. Each chromosome represent a possible solution to address the given problem.
Generations (repeating process): a genetic algorithm follows an evolutionary path towards a solution. Existing chromosomes evolve into new generations of chromosomes by means of recombination and mutation. This evolutionary process is guided by a so-called fitness function. Remember that chromosomes represent a possible solution to a problem. The fitness function, as its name suggests, evaluates the fitness of the solution to the given problem. The higher a chromosome’s fitness score, the more likely it is to proceed to the next generation of chromosomes. This is what Darwin defined as “survival of the fittest” in natural evolution (19).

Crossover: this is a genetic operator that combines (mates) two chromosomes (parents) to produce a new chromosome (offspring). Now there is a probability that new chromosome can be “better” than both of the parents, if it takes the best characteristics from each of the parents (20).

Mutation rate: this is a generic operator; the rate is the probability that a gene is changed at random.

Termination: the evolution process is repeated until a termination condition is reached. Possible terminating conditions are:

- A solution is found that satisfies minimum criteria
- Fixed number of generations reached
- Allocated budget (computation time/money) reached
- The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- Manual inspection
- Combinations of the above (17)
4.2 Experimental setup

The goal of the experiment was to find a corpus (pattern) which fits on one or more manifest(s). The input for the genetic algorithm was a priori knowledge (domain knowledge). The approach is *learning by doing*. This approach was chosen because of its bottom-up strategy. In the grid, (i.e. collaborative environment) knowledge about log-file structures is not always available.

4.2.1 Methodology

Domain knowledge format

Domain knowledge is represented by a regular expression e.g. (PID: (.*)) matches the line PID: 125690). Every regular expression is called a Region of Interest (ROI).

Corpus

Is a collection of ROI’s ordered in a sequence, that has the highest fit on a reoccurring pattern in a manifest. The corpus can be represented by a (directed) graph (see figure 4-1).

The formula of this graph $G$ is $G = (N, E)$

Where $N$ is the set of nodes and $E$ is the set of edges. The edge set $E$ is a subset of the cross product $N \times N$. Each element $(u, v)$ in $E$ is an edge joining node $u$ to node $v$. A node $v$ is neighbor of node $u$ if edge $(u, v)$ is in $E$. The number of neighboring nodes with which the node is connected is called the degree.

Chromosome

An $N \times N$ matrix called an adjacency matrix, it represents which vertices of a graph $G$ are adjacent to other vertices. A chromosome consists of multiple adjacency matrix index numbers. With these numbers a graph can be represented e.g. figure 4-1.

The size of chromosome is a fixed size; it has to be set before run-time.

<table>
<thead>
<tr>
<th>V</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Node 2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Node 3</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 4-1 Adjacency matrix containing index numbers

For instance, index number 7 represent: Parent node 2 has child node 3.
Approach
The experiment uses the following logical steps:

1) Read sample data for instance a manifest or a log file.
2) Gain domain knowledge, currently count words by order, and an available option is an ignore wordlist.
3) Create a list of building blocks, with the help of the domain administrator. Regular expressions are created from step 2 plus additional domain knowledge generated by the administrator.
4) Define a root node for starting point in the graph, see figure 4-1 where PID is the root node.
5) Genetic computation following the logical steps explained in paragraph 4.1.
6) Result is the fittest chromosome containing adjacencies, represented as a graph with regular expressions. This graph is now called a corpus.

4.2.2 Dataset
The dataset was created from a dataset creator. The input parameters are P number of types of structures starting with the root node and N is the size of the dataset in line numbers. R is the number of structures.

With these three parameters, a dataset was created. A dataset was filled with P number of different types of structures and that will be done of an total of R times divided over N lines of the dataset. The dataset was filled up with random noise.

The exact amount of relevant data in the test data was known. Relevant information had the structure of key-value pairs. The key-value pairs were being used for filling the dataset.

The **parameters for this experiment using a dataset was:**

- P = 2, types of structures
- R=25, occurring times a root node with its structure from the P is added in the set
- N=524, number of lines in the dataset

Appendix A gives a detailed description of the generated dataset. It describes patterns that were available in the dataset. The optimal corpus had a hit score of 151 and consisted of 11 adjacencies to represent the graph (see figure 4.2 for the optimal corpus represented in graph).
4.2.3 Fitness function

A fitness function gives a score to an individual chromosome. First step is translating adjacency index numbers to a graph of regular expressions. This graph always has a root node for a starting point.

The function runs through the manifest and tries to overlay its own graph. The function tries to validate the root node on (a part of) a line of the manifest, when this happens. The function saves the fit on the root node. The next step is checking if the node has some child edges. It tries to fit each child node on the manifest. When a node fits on a part of a line of the manifest, the fitness score is saved. The function will then look for next child nodes of the current node that fit on the manifest. This recursive process repeats until it reaches the end of the manifest or when the root node fits once again. (Remember that a root node was taken as the starting point of the pattern.) The starting point for fitting each following node is the position of the parent node (in the manifest).

Figure 4-2 Optimal corpus graph
When the root node fits again, it tells that a re-occurring pattern was found. The function ScoreGraph uses the argmax function. A path consist of nodes that fit on the manifest. Using the argmax will take the longest path (or run) during the range between two root nodes. Summarized the path with the most nodes that fit on a part of the manifest. The result of ScoreGraph is taken and added to the total fitness score.

\[ \text{ScoreGraph} = \text{argmax}(\text{followedpaths}) \]

When the whole manifest is covered and scored by the graph the total score can be summed. N are the times a root node will fit and starting with k=1 summing the scores of ScoreGraph.

\[ \text{Fitness function} = \sum_{k=1}^{n} \text{ScoreGraph}^k \]

Because of the fact that a chromosome is fixed size and there is a possibility that describing the shortest graph will not fill all the spaces in the chromosome. For optimizing to the shortest path description, the Occam’s razor principle is used. Instead of creating a fully connected graph, the shortest path description is preferred. For every item-space in the chromosome that is not filled a bonus of 0.1 is added to the total score of the chromosome. This is a very small bonus for making the difference for shortest description. Therefore, argmax tries to make the longest description possible, while Occam’s razor principle tries to add the shortest possible description. Together these combined “forces” can result in a complete as well as shortest path description.

4.3 Verification, Validation and Evaluation

Precision and recall were used as evaluation measures.

**Precision** - is a measure of exactness or fidelity meaning nothing but the truth.

\[ \text{Precision} = \frac{|TP| + |FP|}{|TP|} \]

**Recall** - is the probability that a (randomly selected) relevant item is retrieved in a search

\[ \text{Recall} = \frac{|TP|}{|TP| + |FN|} \]

The TP (True Positive) represents the number of detected outliers those are real alerts; FP (False Positive) stands for the number of detected outliers that are not actual alerts; FN (False Negative) represents the number of the unidentified data that are actually real alerts. TN (True negative) stands for the number of unidentified data that are not alerts.
4.4 Results
Two approaches were used for the experiment. Using the fitness function (see 4.2.3.) two different mutation operators were applied.

4.4.1 The results of random mutation
This section describes the experiment settings and results using a “standard” random mutation operator. This means a new random mutation rate was created for each evolution and applied to the current generation. Chromosome size and population size were manipulated. Six configurations were tested, and all configurations were tested 5 times (runs) over 300 evolutions. In this case the mutation operator was randomized every evolution.

1) Chromosome size = 11 (equal to the ultimate adjacencies), population size = 50
2) Chromosome size = 11 (equal to the ultimate adjacencies), population size = 100
3) Chromosome size = 15, population size = 50
4) Chromosome size = 15, population size = 100
5) Chromosome size = 20, population size = 50
6) Chromosome size = 20, population size = 100

Experiment summary results - The detailed results of each configuration and run are described in Appendix B. The summarized results are shown in table 4-2.
<table>
<thead>
<tr>
<th>Chromosome size</th>
<th>Population size</th>
<th>Run</th>
<th>Start fitness at evolution 1</th>
<th>Max fitness score</th>
<th>Max fitness score at evolution number</th>
<th>Max fitness score at evolution</th>
<th>Ultimate corpus found</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>50</td>
<td>1</td>
<td>79.0</td>
<td>137.2</td>
<td>37</td>
<td>NO</td>
<td>1</td>
<td>0.907</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>2</td>
<td>99.1</td>
<td>144.1</td>
<td>101</td>
<td>NO</td>
<td>1</td>
<td>0.954</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>3</td>
<td>72.0</td>
<td>130.3</td>
<td>23</td>
<td>NO</td>
<td>1</td>
<td>0.861</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>4</td>
<td>89.0</td>
<td>137.1</td>
<td>261</td>
<td>NO</td>
<td>1</td>
<td>0.907</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>5</td>
<td>99.0</td>
<td>144.1</td>
<td>25</td>
<td>NO</td>
<td>1</td>
<td>0.954</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1</td>
<td>99.0</td>
<td>144.1</td>
<td>18</td>
<td>NO</td>
<td>1</td>
<td>0.954</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>2</td>
<td>89.0</td>
<td>144.1</td>
<td>26</td>
<td>NO</td>
<td>1</td>
<td>0.954</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>3</td>
<td>89.1</td>
<td>144.1</td>
<td>21</td>
<td>NO</td>
<td>1</td>
<td>0.954</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>4</td>
<td>106.0</td>
<td>144.1</td>
<td>30</td>
<td>NO</td>
<td>1</td>
<td>0.954</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>5</td>
<td>99.0</td>
<td>130.3</td>
<td>27</td>
<td>NO</td>
<td>1</td>
<td>0.861</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1</td>
<td>96.0</td>
<td>151.4</td>
<td>56</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>2</td>
<td>85.1</td>
<td>151.4</td>
<td>76</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>3</td>
<td>106.0</td>
<td>151.4</td>
<td>94</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>4</td>
<td>96.0</td>
<td>151.4</td>
<td>279</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>5</td>
<td>106.0</td>
<td>144.5</td>
<td>73</td>
<td>YES</td>
<td>1</td>
<td>0.954</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1</td>
<td>106.0</td>
<td>151.4</td>
<td>32</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>2</td>
<td>113.0</td>
<td>151.4</td>
<td>28</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>3</td>
<td>99.0</td>
<td>144.5</td>
<td>65</td>
<td>NO</td>
<td>1</td>
<td>0.954</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>4</td>
<td>99.0</td>
<td>144.5</td>
<td>38</td>
<td>NO</td>
<td>1</td>
<td>0.954</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>5</td>
<td>113.0</td>
<td>151.4</td>
<td>35</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1</td>
<td>106.0</td>
<td>151.9</td>
<td>274</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>2</td>
<td>106.0</td>
<td>151.9</td>
<td>225</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>3</td>
<td>103.1</td>
<td>151.9</td>
<td>197</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>4</td>
<td>113.0</td>
<td>151.9</td>
<td>200</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>5</td>
<td>96.0</td>
<td>151.9</td>
<td>112</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1</td>
<td>113.0</td>
<td>151.9</td>
<td>121</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>2</td>
<td>116.1</td>
<td>151.9</td>
<td>95</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>3</td>
<td>113.0</td>
<td>151.9</td>
<td>132</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>4</td>
<td>120.1</td>
<td>151.9</td>
<td>63</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>5</td>
<td>116.1</td>
<td>151.9</td>
<td>82</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Table 4-2 Summarized result GA
The max hit score was 151, and the bonus score for the shortest description was added to find the optimal corpus. In the case of chromosome size 20 the optimal score 151.9 of the fact that 9 adjacencies should be empty in the chromosome itself.

The results show that when the chromosome size was equal to the size of the optimal adjacencies (in this case 11) the optimal corpus was not found. A chromosome length of 15 required a lower number of generations to reach the optimal corpus than a chromosome size of 11 and 20. The precision score was always 1 in this test. This was because of the fact that the descriptive attributes were implemented as regular expressions. Regular expressions have a hard fit, so FP are less common.

Two out of six configurations never reached the optimal corpus. The configurations of chromosome size 11 with a population of 50 and 100. The parameter configuration of chromosome size 11 and population size 50 reached 144.1 two times. So the corpus graph score had 144 fit points, and one allele in the chromosome remained empty. This resulted in the additional bonus score of 0.1. The optimal score was 151 and all the alleles in the chromosome should be filled. Results show that the maximum fitness score was reached at an average of 90 evolutions, with an average fitness score of 140.

With a population size of 100 and chromosome size of 11, the average highest fitness score of 141.3. The average of reaching the highest score was after 25 generations. It suggests that a higher population size results in a higher fitness score comparing to the population of 50. And the highest score is also reached at an earlier stage of its evolution. With a population of 50, the average of 140 evolutions is needed to reach the highest score, and with a population of 100 the result is an average of 25 evolutions. Subtracting these results show that using a population of 100 reaches its maximum score 115 evolutions sooner, compared to a population of 50. These results suggest that using a chromosome size equal to its optimal adjacencies (in this case 11, known from the created dataset) does not result in the optimal corpus. The genetic algorithm then tends to reach local optimal solutions.

Using the parameters of chromosome size 15 and population 50 resulted in an average evolution number of 116, before reaching its average maximum fitness score of 150. During 4 out of 5 runs the optimal corpus was found. With a population size 100, the average fitness score ≈ 149 and the average evolution number ≈ 40. This time 3 out of 5 runs reached the optimal corpus.

A chromosome size of 20 and population of 50 resulted in the optimal corpus during all runs of the experiment. In this case the average evolutions (resulting in a maximum score) was 201.6. Both cases resulted in a recall of 1. In other words, it tells that the optimal corpus was found, because the precision is also 1. When the population is set on 100 the average of the evolution finding its maximum fitness score was ≈ 99.

Results suggest that doubling the chromosome size reduces the chance of local optima solutions. Also, a higher starting population seems to reduce the number of evolutions required to reach the optimal corpus. In this experiment chromosome size 20 and population 100 showed the best result of getting the optimal corpus (without respect to the computation time).
In the above experiment, the optimal score was known. In practice this information is not known.

4.4.2 The results of optimized mutation
This section describes the experiment settings and results using a fixed mutation operator. The six configurations of the previous set-up were used with a static mutation parameter for all the evolutions in the algorithm, until termination. Using the six configuration and adding four different settings for the mutation operator (1/5, 1/10, 1/25, 1/50). Mutation set on 1/5 means that 1 of each 5 chromosomes mutates.

The GA with its configuration were tested 5 times (runs) over 300 evolutions.

1) Chromosome size = 11 (equal to the ultimate adjacencies), population size = 50, mutation = 1/5
2) Chromosome size = 11 (equal to the ultimate adjacencies), population size = 50, mutation = 1/10
3) Chromosome size = 11 (equal to the ultimate adjacencies), population size = 50, mutation = 1/25
4) Chromosome size = 11 (equal to the ultimate adjacencies), population size = 50, mutation = 1/50
5) Chromosome size = 11 (equal to the ultimate adjacencies), population size = 100, mutation = 1/5
6) Chromosome size = 11 (equal to the ultimate adjacencies), population size = 100, mutation = 1/10
7) Chromosome size = 11 (equal to the ultimate adjacencies), population size = 100, mutation = 1/25
8) Chromosome size = 11 (equal to the ultimate adjacencies), population size = 100, mutation = 1/50
9) Chromosome size = 15, population size = 50, mutation = 1/5
10) Chromosome size = 15, population size = 50, mutation = 1/10
11) Chromosome size = 15, population size = 50, mutation = 1/25
12) Chromosome size = 15, population size = 50, mutation = 1/50
13) Chromosome size = 15, population size = 100, mutation = 1/5
14) Chromosome size = 15, population size = 100, mutation = 1/10
15) Chromosome size = 15, population size = 100, mutation = 1/25
16) Chromosome size = 15, population size = 100, mutation = 1/50
17) Chromosome size = 20, population size = 50, mutation = 1/5
18) Chromosome size = 20, population size = 50, mutation = 1/10
19) Chromosome size = 20, population size = 50, mutation = 1/25
20) Chromosome size = 20, population size = 50, mutation = 1/50
21) Chromosome size = 20, population size = 100, mutation = 1/5
22) Chromosome size = 20, population size = 100, mutation = 1/10
23) Chromosome size = 20, population size = 100, mutation = 1/25
24) Chromosome size = 20, population size = 100, mutation = 1/50

In Appendix C were the detailed described (see table appendix-1 Summarized result GA fixed mutation rate)
The results show that when the chromosome size was equal to the size of the optimal adjacencies (in this case 11) finding the optimal corpus was seldom. Even with different settings on the mutation rate, the result does not differ much. The result show with a population size of 100 and a chromosome size of 11 that this configuration in an earlier stage of evolutions reaching its highest fitness score, compared to the population size of 50.

A chromosome length of 15 required a lower number of generations to reach the optimal corpus than a chromosome size of 11 and 20. The precision score was almost always 1 in this test. Using the highest population of 100. The algorithm reaches in an early stage its highest score and in the current test most of the time the optimal corpus score 151.4 (including the bonus). Also a higher mutation rate gives an better result, because less evolution are needed to reach it optimal score. In this case the mutation rates of 1/5, and 1/10.

With a chromosome size of 20 the algorithm always find the optimal corpus. Also with different settings of the population size and mutation rate, see a recall of 1. With a higher population size the test give a better result. Changing the mutation rate did not give many differences. However, this chromosome size showed that reaching a score of 151 was done very quickly, which means that the founded corpus fits the test data. Nevertheless, the corpus is too big and consists of too many adjacencies. Reaching its optimal corpus shown to the bonus addition takes a lot of time compared to reaching the 151 corpus score.

These result show that reaching the optimal score is possible with an equal of the chromosome size and needed adjacencies for drawing the optimal corpus. However using a higher chromosome size than the needed adjacencies the result are better.

Using a higher mutation, and population rate the algorithm converges in an earlier stage, by reaching its highest fitness score. Another suggestion in the result are when the chromosome size is much higher than the number of adjacencies a corpus is found that will fit the structure (in this test a score of 151 and higher). However, the optimal score including all the bonus additions is very hard to find. The Occam’s razor effect needs a longer computation time (seen to the evolution of the algorithm)
5 Collaborative environment

In paragraph 2.5 an introduction is given to Multi-Agent systems. This chapter goes deeper in to the workings of these systems, the available architectures. In addition, how agents can support managing dynamic collaborated environments.

5.1 Multi-Agent architectures
An agent internal structure can be categorized into four main types: logical based, reactive, belief-desire-intention (BDI), and layered. The four architectures are implementations of intelligent agents. These architectures determine and address possible agent implementation choices (21).

Logical – is derived from “Traditional AI” where the environment is symbolically presented. The reasoning mechanism is done by logical deduction. A new states of the agent to interact with the environment is achieved with logic. This architecture has two main disadvantages:

(1) The transduction problem. The problem of translating the real world into an accurate, adequate symbolic description of the world, in time for that description to be useful.
(2) The representation/reasoning problem. The problem of representing information symbolically, and getting agents to manipulate/reason with it, in time for the results to be useful (22).

Reactive – direct mapping from situation to action. Agent perceive the environment with their sensors and react to the environment without reasoning about it (see figure 5-1). This architecture is the counterpart of logical with their known limitations. Rodney Brooks is the main contributor for this architecture. He thought out three theses about intelligent behavior.

1) Intelligent behavior can be generated without explicit representations of the kind that symbolic AI proposes.
2) Intelligent behavior can be generated without explicit abstract reasoning of the kind that symbolic AI proposes.
3) Intelligence is an emergent property of certain complex systems.

Figure 5-1 A Robust Layered Control System for a Mobile Robot (23)
**Belief-Desired-Intention** – Has its roots in philosophy and offers a logical theory which defines the mental attitudes of belief, desire and intention using a modal logic. This architecture is also the most popular of the four architectures. This architecture enables to view an agent as a goal directed entity that acts in a rational manner. Agents who adopt this architecture have the following characteristics.

- Situated - they are embedded in their environment
- Goal directed - they have goals that they try to achieve
- Reactive - they react to changes in their environment
- Social - they can communicate with other agents (including humans) (24)

**Beliefs** - Address the informational state of an agent. Representing the information it has about the environment (World) and itself. A belief is similar to knowledge with the difference that knowledge is true, a fact. A belief may not necessarily be true.

**Desires** - Can also be represented as (sub) goals that represent the motivational state of the agent. They represent objectives or situations that the agent would like to accomplish or bring about. Examples of desires might be: find the best price, go to the party or become rich (24).

**Intentions** - Represent the deliberative state of the agent: what the agent has chosen to do. Intentions are desires to which the agent has to some extent committed itself (in implemented systems, this means the agent has begun executing a plan) (24).

Plans: Are sequences of actions that an agent can perform to achieve one or more of its intentions. Plans may include other plans: my plan to go for a drive may include a plan to find my car keys. Therefore, plans are initially only partially conceived, with details being filled in as they progress (24).
Layered (Hybrid) – The last architecture building agents out of two (or more) subsystems:

1) Deliberative, containing a symbolic world model, which develops plans and makes decisions in the way proposed by symbolic AI.
2) Reactive, which is capable of reacting to events without complex reasoning.

The goal of the layered architecture is combining the deliberative and reactive to find a hybrid solution. This option is thought out by researchers argued that both systems individually did not reflect the demands for developing agents (Figure 5-3 Data and control flows in the layered architectures).

5.2 JADE framework

For the current study the JADE framework was used for setting-up a MAS. JADE (Java Agent DEvelopment framework) is framework for developing intelligent agent based system compliant with the FIPA specifications. JADE is free software and is distributed by Telecom Italia, JADE is released as open source software under the terms of the Lesser General Public License (LGPL).

5.2.1 FIPA

The Foundation for Intelligent Physical Agents (FIPA) is an international non-profit association of companies and organizations sharing the effort to produce specifications of generic agent technologies (26). FIPA specifications represent a collection of standards which are intended to promote the interoperability of heterogeneous agents and the services that they can represent (27) Agent Communication Language (ACL), reference model of an agent platform see figure 5-4 FIPA reference model of an agent platform (28).
5.2.2 JADE

The goal of JADE is to simplify developing agent systems. The framework provides group management and individual management of agents. JADE implements the FIPA reference model (figure 5-4). The Agent Management System (AMS) is the agent that supervises and controls over access to and use of the platform. AMS is responsible for accessing the white pages. Every agent is required to register with the AMS.

The Directory Facilitator (DF) is the agent that provides a yellow page service to the agent platform. Every agent can register his service with the DF.

Each agent lives in a container; this container is connected to the main container where the AMS and DF agent live. The agent container is an RMI server object that locally manages a set of agents, controlling the life cycle e.g. create, suspend, resume and kill agent. The container also deals with all the communication by dispatching incoming ACL messages, routing them according to the receiver (26). The container functions for ACC in figure 5-4.

The reason to use JADE in the current study is that the program CTIS already use JADE. It has an active community and is adopted by multiple companies. There is much information available about the framework and its design choices and developing documentation.
Functionality - Interoperability: The agents are able to easily communicate and interact with entities in other systems (even with those that were not foreseen during the original development), whether they are agents themselves, or more traditional applications. Thus, the agent can act in a heterogeneous, distributed environment.

Reliability: The platform has become mature so it will be more and more reliable. During the testing period JADE did not give any errors. To prevent a single point of failure the main container, JADE provides replication of the main container. If a main container fails another container will take over the roll and interacts then as main container.

Usability: JADE provides documentation. Agent based techniques are complex and difficult to understand. JADE tries to keep it simple as possible and works with examples and provide a graphical-user-interface for managing the MAS.

Maintainability: The framework was developed for creating agent-based system, the maintenance is low. Different agents can be developed and communicate with each other by using de FIPA ACL messages. When there are new types of agents created they can easily adopted to interact with the existing agent by communication and using their services (heterogeneity)

Portability: Because it’s an agent platform the system is adaptable. Agents try to adapt to their environment to achieve their goal. JADE provides mechanism like creating, suspending and removing agents from each domain. JADE can even let agents migrate between domains if and only if the source code is serializable.

5.3 Coordination
A multi agent system consists of multiple entities that can be heterogeneous. In that case, it is important have a good coordination between those entities to ensure less complexity and inefficiency in the system.

There are several definitions of coordination:

- Harmonious adjustment or interaction\(^3\)
- The act of making parts of something, groups of people, etc. work together in an efficient and organized way\(^4\)

Coordination in multi agent systems is very important. First one has to describe how the environment of a multi agent system is organized. After this step agent can interact with the environment. Because agents have the ability to interpret, and perceive the environment, and action to influence the environment; their interaction can influence it. Communication handles agent interactions. Challenging, and complex networks need to dynamically ensure adequate management of activities attributed to a large number heterogeneous entities. A structured way of communication is in this case very important.

---

\(^3\) Heritage dictionary: [http://dictionary.reference.com/browse/coordination](http://dictionary.reference.com/browse/coordination)

FIPA provides protocols for interaction and communication. For every interaction protocol mandated by FIPA specifications, two roles can be taken by an agent:

1. Initiator role: the agent contacts one or more other agents to start up a new conversation, evolving according a specific interaction protocol.

2. Responder role: in response to a message received from some other agent, the agent carries on a new conversation following a specific interaction protocol. (29)

There several interaction protocols proposed by the FIPA (30), common used is the Request Interaction Protocol (SC00026).

The sequence diagram, figure 5-5 Request Interaction Protocol explains the Request Interaction Protocol. The initiator requests other participants (agent in the network), to perform an action. The participant makes the decision to accept or refuse the request. The agree notification is only sent when a necessary notification is set.

When a participant [agreed], it always has to send a notification to the initiator.

- failure, the request can’t be fulfilled
- inform-done, the request is successfully executed
- inform-result, the request needs some information or results sent back to the initiator

By following conventions e.g. the Request Interaction Protocol or other variants it is possible to create a structure and format standardized of communication, (Agents follow a set of prescribed rules and with this type of communication coordination can be achieved).
5.4 Discussion

In this section, I discuss agent architectures and the JADE framework.

The current study addressed a dynamic, and unpredictable environment, where challenging organizational challenges were approached. This included self-organization, self-building, and adaption in order to cope with the effects of large-scale environments. A software solution for performing in these environments was proposed consisting of a multi-agent system.

The reason for choosing JADE was described in paragraph 5.2. It uses the standard ISO 9126 for software quality. JADE is FIPA compliant and was used in the CTIS program (see SIMON).

The next question was which architecture fits the most to the problem-description of the current study.

JADE provides a framework with which the developer can implement an architecture. The current situation required an architecture with which it is possible to find and share ROI’s or a corpus between agents. Agents have to have the ability to look for manifest, preprocess, find a corpus, and exchanging knowledge.

**Logical architecture:** it is hard to represent the environment in a totally logical representation. In this case an agent reasons about the next action at a given time. The agent reasons internal about its optimal action, and during this reasoning the environment can change. When the agent has found its optimal action, it might execute it in the changed environment, requiring a new optimal action.

**Reactive architecture:** in this architecture, agents only react on their environment without having their own goal or task. This architecture is not preferred in the specific problem description whereby the system should have the task or goal in finding structure in a set of manifests.

**Layered-agents:** this architecture was also an option. Combining the best of two worlds. However, implementing multiple architectures rapidly increases complexity.

**Belief-Desire-Intention:** this architecture seems to perform well given the problem description. This architecture tries to create a virtual brain and beliefs, desires and intention can be exchanged and imported by another agent that uses BDI. The domain knowledge can be represented as fact or beliefs. The main goal of finding structure in manifest is seen as an intention, whereby this goal can be split up in sub-goals. The sequence to achieve a goal is the intention or plan. For example, an agent has some behaviors to achieve a goal. The plan is a sequence of these behaviors with a preferred outcome state.
When agents work with BDI they can interact with other agents by exchanging beliefs, desires, intentions and plans. Much like a social human being. This architecture fits the challenges of the current study. Whereby the problem can be split into sub-goals, and knowledge about the environment can be mapped into beliefs. Belief-Desire-Intention seems to be the best approach for implementation, because it suits the most to the challenges e.g. volatility, unpredictability, and dynamic environment. An agent acts autonomous in achieving its (sub)-goals, using conventions for coordination, in order to cooperate. Agents have control over their own state and behavior and can perceive the environment.
6 ISEA: A prototype for finding structures in manifests using a Multi-Agent system

The prototype was created in Java using the JADE framework and following the FIPA protocols and agreements e.g. ACL messages, see paragraph 5.2 JADE framework. The prototype Intelligent SEIf-describing Agents (ISEA) uses genetic computation for finding structures in log-files, based on the case description(1.2 Case: Supporting grid management and administration). The Multi-Agent system act in a volatile and dynamic environment, where the Multi-Agent system is a collaborative learning environment. The challenge for this prototype is to learn from each other. Each agent uses different fragments of a priori knowledge in finding corpora.

JADE uses behaviors, representing a task or action that an agent can carry out. Each behavior can be started with an action command. The behavior performs a task or terminates on its given conditions. Possible behaviors are cyclic, one-shot and ticker.

Cyclic – Atomic behavior that executed forever, like a thread in Java.
One-shot – Executes one task and then the behavior will be terminated.
Ticker – The behavior that periodically executes a user-defined piece of code

The previous chapter described the BDI – model. This is an abstract way of possible agent architecture. Using this mind-set and implementation, the following approach was proposed:

An agent has beliefs. These represent then informational state of the agent. All the beliefs are stored in belief sets. Desires of an agent represent objectives: what the agent would like to achieve. They can be mapped as a goal. Intentions are plans to achieve a goal. They are a sequence of behaviors. There are also events to which an agent reacts. These can be received information from its sensors, e.g. a request call from a neighbor agent.

Agent life-cycle
Agents read their beliefs from an XML file. Each agent registers its services, which are the capabilities of an agent. When the agent is initialized, its state is known and it tries to achieve its goals by executing plans and reacting on events. In the implementation, desires and intention are merged into the type of behaviors of JADE. This means that the execution-plans to achieve a goal are merged into one behavior.
6.1 Setup

The prototype was built for the case study (1.2). The prototype is capable of creating a collaborative learning environment, where agents can learn from each other by exchanging knowledge. The goal of the prototype was to illustrate how the genetic computation-methods (see chapter 4) could be applied to an agent-based system.

The prototype was tested by using several agents with little knowledge about the same log file. Each agent knows the location of the log file and has the JADE behavior to analyze a log file using genetic computation. The genetic algorithm experiment described in chapter Genetic computation 4.1 was implemented in the JADE agents.

JADE can use an ontology. The prototype uses a minimal vocabulary and ontology to describe or map the environment. The prototype was used to show that agents can learn from each other and solve complex problems in a distributed way. In short the prototype vocabulary existed of:

\{ corpusScore, score, da, name, expression, ihaveScore, ihaveDA \}

- CorpusScore is the internal score of each agent, by using the genetic behavior on the log file.
- Score is an incoming score from neighbor agents.
- DA is a descriptive attribute.
  - Name is a part of a DA, the name of the DA.
  - Expression is a part of a DA, the regular expression of the DA
- IhaveScore is a description that an agent has a corpusScore
- IhaveDA is a description than an agent has one or more DA’s

See figure 6-1 sequence diagram agent mind mapping. After an agent is created and started, each agent checks if the location of the log files exists and can be opened, this is done by reading their beliefs. An agent tries to build a corpus from the known log file and the given descriptive attributes. For testing the prototype there were 4 agents created using the same log file, with different given set of descriptive attributes.

When an agent has build a corpus and a corpus has a fit score on the log file, the agent broadcasts its fitness score to the other agents. Building corpora is done according to a fixed period, depending on the configuration setting.

Each incoming score (originating from another agent) is checked by the agent internally. When the incoming score is higher than its own, the agent sends a request for descriptive attributes. The agent with the highest scores sends a “IhaveDA” with its descriptive attributes, the requesting agent puts the
*descriptive attributes* into its beliefs. When the corpus-building period is expired, a new corpus is build with the DA’s. So agents can create a better/fitter corpus.

![Sequence diagram agent mind mapping](image)

**Figure 6-1 Sequence diagram agent mind mapping**

### 6.2 Results
The prototype was tested by using four agents located on two different machines. Each agent had different beliefs (descriptive attributes) but all had the same root node. Each agent built a corpus every 5 minutes. Every agent used the sequence diagram explained in the previous paragraph (figure 6-1). The prototype showed that agents were able to learn from each other. Over time each agent was able to build identical corpora, because they knew all the descriptive attributes known in the agent network. With this prototype, it is shown that genetic computation can be applied in a collaborative environment (Multi-Agent system).
7 Discussion & conclusion

The first goal of the current study was automated text mining. Genetic computation was proposed to find structures in semi-structured data. The current study showed that this method was successful in extracting relevant information from semi-structured data. Using genetic computation it was possible to create a graph-based corpus, using descriptive attributes as input. These descriptive attributes consisted of regular expressions that represented a priori knowledge of log-file attributes. Previous studies showed promising result in spam detection using genetic computation (18) and graph-based structure finding (12). The current study implemented genetic computation in a collaborative learning environment.

The effectiveness of genetic computation depended on several parameters. First, a high mutation rate gave better results in finding the optimal corpus. Second, chromosome size had to be larger than the optimal size of adjacencies of the optimal corpus. It was shown that chromosome size had to be approximately 1.5 times the number of optimal adjacencies. Larger values resulted in longer periods of optimization without improving results.

The second goal of the current study was to implement a prototype of a Multi-agent system for autonomous text mining in a distributed environment. Using the BDI architecture the prototype consisted of agents that were able to exchange knowledge. Each agent was provided with fragmented knowledge about the log file. It was shown that agents could work together to combine knowledge, and find the ultimate (graph-based) corpus that described the test data.

Some considerations have to be taken into account. First, the current study did not address computational limitations or time constrains. Second, the chromosome size of the optimal solution was known in the current study (based on the test data). The current study revealed that there was a relation between the predefined length of the chromosome, and the length of the solution, in terms of optimal result. It was shown the predefined length of the chromosome had to be 1.5 times the length of the solution, in order to get the optimal corpus. Determining the solution length in real life situations might be a challenge. However, this is beyond the scope of the current research.

The current study was a proof of concept. It showed that automated text mining in a distributed environment is possible, using multi-agent systems. The automated text mining approach described in the current study showed that it is possible to extract corpora from semi-structured manifests. The Multi-agent prototype forms a collaborative learning environment. The BDI agent architecture performs well in addressing solutions to the current challenging problems. This approach could be used to assist domain administrators in the case problem description. Agent could be used to federate domain knowledge in a volatile and multiple-domain environment.
The combination of text-mining and agent technology (agent mining) is a relevant area of research. The Agents and Data Mining Interaction and Integration (ADMI) research group announced that a workshop on Agents & Data Mining Interaction (ADMI-10) has been accepted by AAMAS-2010.
References


2. Kamber, Micheline and Han, Jiawei. *Data Mining: Concepts and Techniques*.


http://giftedkids.about.com/od/glossary/g/intelligence.htm.


14. Ubiquitous Intelligence in Agent mining. Zhang, Chengqi, Luo, Dan and Cao, Longbing. s.l. : University of Technology Sydney, Australia.


28. **Poslad, Stefan, Buckle, Phil and Hadingham, Rob.** *The FIPA-OS agent platform: Open Source for Open Standards.*


Appendix A

Description of the dataset that was used for the Genetic Algorithm experiment

There were two patterns available and a total of 151 relevant key-value pairs set. Where PID is the root node.

1 - size: 7

| PID: \OP= Person: IP date |
|-------------------------|------------------------|
| 12 : 3, 13 : 1, 14 : 1, 15 : 3, 16 : 5, 17 : 0, 18 : 3, 19 : 5, 1211 : 2, 1223456 : 1 |

2 - size: 6

<table>
<thead>
<tr>
<th>PID: \qwert= IP date</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC 1 : 1, PC 2 : 1, PC 3 : 1, PC 4 : 0, PC 5 : 1, PC 6 : 0, PC 7 : 0, PC 8 : 0, PC 9 : 2, PC 10 : 1</td>
</tr>
</tbody>
</table>

The following data was inserted in the data set, according to the above re-occurring patterns. Relevant information items – Name: [value: number of occurrence, etc]

PID: [12 : 3, 13 : 1, 14 : 1, 15 : 3, 16 : 5, 17 : 0, 18 : 3, 19 : 5, 1211 : 2, 1223456 : 1]

Person: [Simon : 1, Sjaak : 0, Kees : 1, Klaas : 0, Pieter : 0, Jan : 1, Joke : 1, Stefan : 0, Erik : 0, Mi Hou : 3]


\CH= [PC 1 : 1, PC 2 : 1, PC 3 : 1, PC 4 : 0, PC 5 : 1, PC 6 : 0, PC 7 : 0, PC 8 : 0, PC 9 : 2, PC 10 : 1]

\qwert= [CERN : 4, GRID : 4, RUG : 5, UVA : 6, UMCG : 0, AMC : 1, UMC : 5, NIKHEF : 5, EGI : 2, IEEE : 1, CERN 13a : 1]

\OP= [De Wilp : 2, Marum : 0, Leek : 1, Tolbert : 1, Opende : 0, S’Veen : 1, Eelde : 0, Hoogkerk : 0, Groningen : 1, Leeuwarden : 1]
Appendix B

The detailed results for chromosome size = 11 and population 50

The detailed results for chromosome size = 11 and population 100
The detailed results for chromosome size = 15 and population 50

The detailed results for chromosome size = 15 and population 100
The detailed results for chromosome size = 20 and population 50

The detailed results for chromosome size = 20 and population 100
## Appendix C

<table>
<thead>
<tr>
<th>Chromosome size</th>
<th>Population size</th>
<th>Mutation rate</th>
<th>Run</th>
<th>Start fitness at evolution 1</th>
<th>Max fitness score</th>
<th>Max fitness score at evolution number</th>
<th>Max fitness score at evolution</th>
<th>Ultimate corpus found</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>50</td>
<td>1/5</td>
<td>1</td>
<td>106.0</td>
<td>134.2</td>
<td>142</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/5</td>
<td>2</td>
<td>99.1</td>
<td>144.1</td>
<td>23</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/5</td>
<td>3</td>
<td>89.0</td>
<td>151.0</td>
<td>31</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/5</td>
<td>4</td>
<td>89.0</td>
<td>144.1</td>
<td>128</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/5</td>
<td>5</td>
<td>106.0</td>
<td>144.1</td>
<td>19</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/10</td>
<td>1</td>
<td>85.0</td>
<td>134.2</td>
<td>32</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/10</td>
<td>2</td>
<td>82.0</td>
<td>127.2</td>
<td>28</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/10</td>
<td>3</td>
<td>89.0</td>
<td>137.2</td>
<td>23</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/10</td>
<td>4</td>
<td>89.1</td>
<td>137.2</td>
<td>70</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/10</td>
<td>5</td>
<td>89.0</td>
<td>144.1</td>
<td>41</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/25</td>
<td>1</td>
<td>82.0</td>
<td>137.2</td>
<td>35</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/25</td>
<td>2</td>
<td>96.0</td>
<td>151.0</td>
<td>71</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/25</td>
<td>3</td>
<td>89.0</td>
<td>137.2</td>
<td>47</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/25</td>
<td>4</td>
<td>85.0</td>
<td>127.3</td>
<td>88</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/25</td>
<td>5</td>
<td>85.0</td>
<td>144.1</td>
<td>22</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/50</td>
<td>1</td>
<td>89.0</td>
<td>144.1</td>
<td>39</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/50</td>
<td>2</td>
<td>82.0</td>
<td>144.1</td>
<td>97</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/50</td>
<td>3</td>
<td>99.0</td>
<td>144.1</td>
<td>81</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/50</td>
<td>4</td>
<td>82.0</td>
<td>151.0</td>
<td>131</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>50</td>
<td>1/50</td>
<td>5</td>
<td>79.1</td>
<td>134.1</td>
<td>78</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/5</td>
<td>1</td>
<td>89.0</td>
<td>144.1</td>
<td>37</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/5</td>
<td>2</td>
<td>89.0</td>
<td>151.0</td>
<td>31</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/5</td>
<td>3</td>
<td>99.0</td>
<td>137.2</td>
<td>27</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/5</td>
<td>4</td>
<td>82.1</td>
<td>151.0</td>
<td>39</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/5</td>
<td>5</td>
<td>89.0</td>
<td>137.2</td>
<td>19</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/10</td>
<td>1</td>
<td>89.0</td>
<td>144.1</td>
<td>29</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/10</td>
<td>2</td>
<td>103.0</td>
<td>144.1</td>
<td>31</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/10</td>
<td>3</td>
<td>106.0</td>
<td>137.2</td>
<td>19</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/10</td>
<td>4</td>
<td>106.0</td>
<td>151.0</td>
<td>151</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/10</td>
<td>5</td>
<td>106.0</td>
<td>144.1</td>
<td>265</td>
<td>NO</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table Appendix-1 Summarized result GA fixed mutation rate*
<table>
<thead>
<tr>
<th>Chromosome size</th>
<th>Population size</th>
<th>Mutation rate</th>
<th>Run</th>
<th>Start fitness at evolution 1</th>
<th>Max fitness score at evolution 1</th>
<th>Number</th>
<th>Max fitness score at evolution 2</th>
<th>Ultimate corpus found</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>100</td>
<td>1/25</td>
<td>1</td>
<td>89.0</td>
<td>137.2</td>
<td>47</td>
<td>NO</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/25</td>
<td>2</td>
<td>99.0</td>
<td>137.2</td>
<td>52</td>
<td>NO</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/25</td>
<td>3</td>
<td>106.1</td>
<td>144.1</td>
<td>31</td>
<td>NO</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/25</td>
<td>4</td>
<td>82.1</td>
<td>151.0</td>
<td>39</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/25</td>
<td>5</td>
<td>106.2</td>
<td>144.1</td>
<td>25</td>
<td>NO</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/50</td>
<td>1</td>
<td>89.0</td>
<td>144.1</td>
<td>24</td>
<td>NO</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/50</td>
<td>2</td>
<td>106.0</td>
<td>137.2</td>
<td>33</td>
<td>NO</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/50</td>
<td>3</td>
<td>106.0</td>
<td>134.2</td>
<td>180</td>
<td>NO</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/50</td>
<td>4</td>
<td>82.0</td>
<td>151.0</td>
<td>18</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>1/50</td>
<td>5</td>
<td>89.0</td>
<td>144.1</td>
<td>20</td>
<td>NO</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/5</td>
<td>1</td>
<td>106.0</td>
<td>151.4</td>
<td>189</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/5</td>
<td>2</td>
<td>116.0</td>
<td>151.4</td>
<td>105</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/5</td>
<td>3</td>
<td>106.0</td>
<td>151.4</td>
<td>83</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/5</td>
<td>4</td>
<td>99.0</td>
<td>151.4</td>
<td>168</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/5</td>
<td>5</td>
<td>99.0</td>
<td>151.4</td>
<td>167</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/10</td>
<td>1</td>
<td>106.0</td>
<td>151.4</td>
<td>156</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/10</td>
<td>2</td>
<td>106.0</td>
<td>151.4</td>
<td>48</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/10</td>
<td>3</td>
<td>99.0</td>
<td>151.4</td>
<td>61</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/10</td>
<td>4</td>
<td>106.0</td>
<td>144.5</td>
<td>81</td>
<td>NO</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/10</td>
<td>5</td>
<td>99.0</td>
<td>151.4</td>
<td>63</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/25</td>
<td>1</td>
<td>106.0</td>
<td>151.4</td>
<td>267</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/25</td>
<td>2</td>
<td>106.0</td>
<td>151.4</td>
<td>82</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/25</td>
<td>3</td>
<td>113.0</td>
<td>151.4</td>
<td>56</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/25</td>
<td>4</td>
<td>113.1</td>
<td>151.4</td>
<td>175</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/25</td>
<td>5</td>
<td>106.0</td>
<td>151.4</td>
<td>143</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/50</td>
<td>1</td>
<td>113.0</td>
<td>151.4</td>
<td>81</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/50</td>
<td>2</td>
<td>106.0</td>
<td>151.4</td>
<td>161</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/50</td>
<td>3</td>
<td>106.0</td>
<td>151.4</td>
<td>248</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/50</td>
<td>4</td>
<td>89.0</td>
<td>151.4</td>
<td>174</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>1/50</td>
<td>5</td>
<td>113.0</td>
<td>151.4</td>
<td>80</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table Appendix-2: Summarized results from GA fixed mutation rate

---

**Precision** indicates whether the ultimate corpus was found (YES or NO). **Recall** is marked with either 1 or 1, likely indicating a consistent value or count.
<table>
<thead>
<tr>
<th>Chromosome size</th>
<th>Population size</th>
<th>Mutation rate</th>
<th>Run</th>
<th>Start fitness at evolution</th>
<th>Max fitness score</th>
<th>Max fitness score at evolution number</th>
<th>Max fitness score at evolution</th>
<th>Ultimate corpus found</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>100</td>
<td>1/5</td>
<td>1</td>
<td>113.0</td>
<td>151.4</td>
<td>46</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/5</td>
<td>2</td>
<td>116.0</td>
<td>151.4</td>
<td>57</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/5</td>
<td>3</td>
<td>106.0</td>
<td>151.4</td>
<td>65</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/5</td>
<td>4</td>
<td>106.0</td>
<td>151.4</td>
<td>83</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/5</td>
<td>5</td>
<td>96.0</td>
<td>151.4</td>
<td>48</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/10</td>
<td>1</td>
<td>99.0</td>
<td>151.4</td>
<td>46</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/10</td>
<td>2</td>
<td>106.0</td>
<td>151.4</td>
<td>55</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/10</td>
<td>3</td>
<td>106.0</td>
<td>151.4</td>
<td>39</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/10</td>
<td>4</td>
<td>116.0</td>
<td>151.4</td>
<td>26</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/10</td>
<td>5</td>
<td>113.0</td>
<td>151.4</td>
<td>37</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/25</td>
<td>1</td>
<td>106.0</td>
<td>151.4</td>
<td>92</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/25</td>
<td>2</td>
<td>113.0</td>
<td>151.4</td>
<td>196</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/25</td>
<td>3</td>
<td>99.1</td>
<td>151.4</td>
<td>90</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/25</td>
<td>4</td>
<td>120.0</td>
<td>151.4</td>
<td>44</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/25</td>
<td>5</td>
<td>106.1</td>
<td>151.4</td>
<td>85</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/50</td>
<td>1</td>
<td>106.0</td>
<td>151.4</td>
<td>88</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/50</td>
<td>2</td>
<td>113.0</td>
<td>151.4</td>
<td>78</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/50</td>
<td>3</td>
<td>106.0</td>
<td>151.4</td>
<td>105</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/50</td>
<td>4</td>
<td>99.0</td>
<td>151.4</td>
<td>170</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>1/50</td>
<td>5</td>
<td>99.0</td>
<td>151.4</td>
<td>59</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/5</td>
<td>1</td>
<td>113.0</td>
<td>151.9</td>
<td>197</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/5</td>
<td>2</td>
<td>116.0</td>
<td>151.9</td>
<td>72</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/5</td>
<td>3</td>
<td>106.0</td>
<td>151.9</td>
<td>151</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/5</td>
<td>4</td>
<td>123.0</td>
<td>151.9</td>
<td>90</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/5</td>
<td>5</td>
<td>137.1</td>
<td>151.9</td>
<td>126</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/10</td>
<td>1</td>
<td>106.0</td>
<td>151.9</td>
<td>152</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/10</td>
<td>2</td>
<td>120.0</td>
<td>151.9</td>
<td>150</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/10</td>
<td>3</td>
<td>113.0</td>
<td>151.9</td>
<td>112</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/10</td>
<td>4</td>
<td>113.1</td>
<td>151.9</td>
<td>147</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/10</td>
<td>5</td>
<td>113.0</td>
<td>151.9</td>
<td>154</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Table Appendix-3 Summarized result GA fixed mutation rate
<table>
<thead>
<tr>
<th>Chromosome size</th>
<th>Population size</th>
<th>Mutation rate</th>
<th>Run</th>
<th>Start fitness at evolution 1</th>
<th>Max fitness score</th>
<th>Max fitness score at evolution number</th>
<th>Max fitness score found</th>
<th>Ultimate corpus found</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>50</td>
<td>1/25</td>
<td>1</td>
<td>123.1</td>
<td>151.9</td>
<td>115</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/25</td>
<td>2</td>
<td>106.1</td>
<td>151.9</td>
<td>184</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/25</td>
<td>3</td>
<td>120.0</td>
<td>151.9</td>
<td>162</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/25</td>
<td>4</td>
<td>113.0</td>
<td>151.9</td>
<td>147</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/25</td>
<td>5</td>
<td>99.0</td>
<td>151.9</td>
<td>73</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/50</td>
<td>1</td>
<td>106.0</td>
<td>151.9</td>
<td>137</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/50</td>
<td>2</td>
<td>120.0</td>
<td>151.9</td>
<td>185</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/50</td>
<td>3</td>
<td>120.0</td>
<td>151.9</td>
<td>221</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/50</td>
<td>4</td>
<td>106.0</td>
<td>151.9</td>
<td>147</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>1/50</td>
<td>5</td>
<td>106.0</td>
<td>151.9</td>
<td>60</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/5</td>
<td>1</td>
<td>120.0</td>
<td>151.9</td>
<td>51</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/5</td>
<td>2</td>
<td>120.0</td>
<td>151.9</td>
<td>113</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/5</td>
<td>3</td>
<td>123.0</td>
<td>151.9</td>
<td>68</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/5</td>
<td>4</td>
<td>106.0</td>
<td>151.9</td>
<td>167</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/5</td>
<td>5</td>
<td>120.0</td>
<td>151.9</td>
<td>147</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/10</td>
<td>1</td>
<td>113.0</td>
<td>151.9</td>
<td>60</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/10</td>
<td>2</td>
<td>113.0</td>
<td>151.9</td>
<td>91</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/10</td>
<td>3</td>
<td>120.1</td>
<td>151.9</td>
<td>146</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/10</td>
<td>4</td>
<td>116.0</td>
<td>151.9</td>
<td>54</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/10</td>
<td>5</td>
<td>120.0</td>
<td>151.9</td>
<td>77</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/25</td>
<td>1</td>
<td>113.0</td>
<td>151.9</td>
<td>58</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/25</td>
<td>2</td>
<td>120.0</td>
<td>151.9</td>
<td>57</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/25</td>
<td>3</td>
<td>113.0</td>
<td>151.9</td>
<td>64</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/25</td>
<td>4</td>
<td>120.0</td>
<td>151.9</td>
<td>84</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/25</td>
<td>5</td>
<td>130.0</td>
<td>151.9</td>
<td>148</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/50</td>
<td>1</td>
<td>113.0</td>
<td>151.9</td>
<td>141</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/50</td>
<td>2</td>
<td>106.0</td>
<td>151.9</td>
<td>100</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/50</td>
<td>3</td>
<td>127.1</td>
<td>151.9</td>
<td>105</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/50</td>
<td>4</td>
<td>113.0</td>
<td>151.9</td>
<td>51</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>1/50</td>
<td>5</td>
<td>123.0</td>
<td>151.9</td>
<td>113</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Table Appendix-4 Summarized result GA fixed mutation rate