IDIMAS

Information Discovery and Integration using a Multi-Agent System with ant colony optimization

P.S. Grasdijk
Artificial Intelligence
University of Groningen, the Netherlands

Internal supervisor:
Rineke Verbrugge (Artificial Intelligence, UG)

August 10, 2017

Abstract

With the abundance of unstructured data on the Internet, retrieving and extracting relevant high-quality data has become a major challenge. Relevant data is diffuse with different unknown windows of availability, meaning that data can already be removed before crawlers are able to find it. The data is acquired through text mining, which is the process of information discovery, extraction and integration of text.

Text mining contains many sub-fields. Information retrieval focuses on finding which pages contain relevant information. Information extraction extracts relevant information from those pages. Information normalization normalizes the information in order to allow the resulting data objects to be created, identified and compared. Finally, the matching data objects are fused or clustered, depending on the overall goal.

In this thesis we describe the design of a multi-agent system that discovers, extracts and integrates information about books from vendors. Part of the pre-processing is done outside of the model in order to reduce complexity. This includes the crawling of web pages, its extraction of book information and finally, part of the normalization of information. The quality of the books is used as a basis for ant colony optimization, in order to assess the quality of the vendors and to decide which vendors will be used for further information retrieval. A variety of parameters and algorithms have been tested and compared, using three types of algorithms, namely Ant System (AS), Rank-based Ant System (RAS), and random walk. The parameters that have been tested are the number of crawler agents, the pheromone amount update, the evaporation rate and the importance of various book categories for the calculating of the book quality. Secondly, an experiment has been conducted in which the agents were allowed to request feedback on books that have already been retrieved.

The experiments show that all parameters except the pheromone amount update heavily influence the effectiveness of an algorithm, as expected. Especially the number of crawlers relative to the size of the database is of large influence. Agent feedback also has a large impact on the results, creating a smaller, but more up-to-date knowledge base. In general, the opposite of the hypotheses occurred. Random walk outperformed Ant System and Rank-based Ant System for many parameters in results such as the number of books in the knowledge base, number of unique author-title combinations, number of book duplicates and how up-to-date the knowledge base was. Further research could be done with other swarm intelligence algorithms such as the bees algorithm, an algorithm that takes long-term vendor history less into account.

Contents

Chapter 1

Introduction

During the past few decades, the world wide web has become an integral part of our lives [5]. The amount of freedom and power that a user or corporation has on the Internet is unparalleled in this world. Everyone can create a website and claim their place on the world wide web. The issue then is to become noticed, as the Internet is an ever increasing large decentralized system, making it impossible to be fully documented [1].

To automate the process of documenting the Internet, crawlers for search engines were invented [26]. These automated agents crawl the Internet to find websites and their content. These pages then get classified and indexed in order to be part of the documented Internet from that search engine. This classification happens on a more general level in order to depict the type of web page, but has recently also been focusing on the actual contents of the web pages. This is done through web content mining and subsequently through information extraction.

As the Internet has become such an important resource, the information which flows from it has become important as well. Buying clothes, electronics or even groceries online has become normal business. When planning holidays, trips to family or even the route to a friend in the same city, an online map and route planner will guide you to your goal.

Information therefore dictates most of what we do in our lives. Because a lot of information is currently findable online, the problem becomes not only where to find relevant information, but also how correct that information is. For instance, it could be very handy to know that a store says that it is selling a rare edition of a book that you really want. You arrive at the store, using the Internet for a route description, only to find out that the book was sold weeks ago and that they haven't yet updated their website. This example illustrates the point that knowing that something is true at some point, does not mean that it is still true a few weeks later. This way of looking at information is defined as the *freshness* of information, and depicts how old the information is.

Knowing how old the information is still does not guarantee the correctness of information. It generally says something about the likelihood of the information still being correct. Even this depends on the context of the information which a person is looking for, because very fresh information does not necessarily mean that it is true: just look at media, for example.

Finding relevant, up-to-date and correct information therefore has become of paramount importance on the Internet. But as the Internet has become a whirlpool of information, misinformation,

facts and lies, it has become more important than ever to be able to objectively know the correctness of what is being said. However, the Internet is unstructured and constantly changing, making it a very complex problem [94].

A Multi-Agent System (MAS) can help index, classify and extract information, as a MAS in itself also is decentralized. Furthermore, it is self-sufficient and can reach far without human interaction, allowing to follow many threads simultaneously. It can also run a range of tasks, not just finding and indexing new information, but also through looking at older web pages and checking if the information has changed.

However, since applying the concept to every domain is too complex, a selection has to be made as to what information the model focusses on.

1.1 Application domain

The concept should function for every domain, due to its generative nature. However, for the research purposes, some domains are more suitable than others. This is mainly due to the nature of the information in the domain involved.

The domain chosen is about books, specifically about rare or antique books. Rare or antique books aren't available in large supply, which means that the books can appear and disappear from trader websites. This adds an importance to find the books as they appear, but also to note the date when they disappear. When a person is looking for a specific book, it would be beneficial to only view the versions of that book which are still available. The versions of the books that were available at one point can be used to view the history of the pricing of the book, which provides information on which books are worth their value.

Currently not a lot has been done on this domain, other than aggregating information from the data feeds from various sites, which are then compiled based on authors and how well the search terms fits the title of the books. For other domains, in order to keep information up to date, actual people are used to ascertain the relevance of the information within that domain. Using a decentralized, autonomous model for the acquisition and integration of information, I can hopefully create a method for which the cost and speed of obtaining this information is a lot cheaper and faster.

1.2 Research Questions

Therefore, the problem can be described using a number of research questions. A bridge is also attempted to a practical approach, via the last research question.

1. How does a Multi-Agent System provide a suitable solution for the issues of information discovery and integration in unstructured data?
2. Can a proof-of-concept be created that is a complete model for information discovery and integration?
3. Can Ant Colony Optimization be used for learning the quality of databases?

   (a) Does Ant Colony optimization (ACO) (Ant Colony optimization) provide a more suitable alternative than a random crawler?

4. What are requirements for a real-world implementation of a multi-agent system for information discovery and integration?

1.3 Thesis structure

Since the thesis covers a large number of different fields with their own issues and considerations, different chapters touch different fields and next specifically the topics which are related to the thesis.

Part I contains Chapters 2 and 3. In Chapter 2, a brief introduction to MAS is given, as well as the notions of trust and reputation are discussed. These are put in context of the model Information Discovery and Integration using a Multi-Agent System (IDIMAS) that has been developed in this thesis research project. In Chapter 3, Ant Colony optimization is introduced, with also the introduction of various variants of Ant Colony optimization, such as Rank-based Ant System and Max-Min Ant System.

Then Part II contains Chapters 4 and 5, which head in a more practical application of the current solutions and strategies for the fields of information discovery and integration. Chapter 4 discusses crawlers and data mining, while Chapter 5 discusses the issues and solutions for data, object identification and fusion.

Finally, there is Part III, which contains the proof-of-concept IDIMAS. Chapter 6 discusses the data, how it was retrieved and normalized in order to be usable by the model. Chapter 7 introduces the model IDIMAS. Chapter 8 contains the setup and hypotheses for a set of experiments. The results from these experiments are shown in Chapter 9, and are discussed in Chapter 10. Finally, some conclusions are drawn in Chapter 11, which also discusses potential further work.

Part I

Theoretical Background

Chapter 2

Dialogues, trust and reputation in a multi-agent system

2.1 Multi-agent systems

A MAS is a computational system where a group of autonomous agents work together in order to solve a given problem [97]. The problem is too large for a single agent, thus requiring multiple agents. An autonomous agent is defined as follows by Jennings et al [62].

> An agent is a computer system, *situated* in some environment, that is capable of *flexible autonomous* action in order to meet its design objectives.

The MAS is a distributed or concurrent system, where agents are self-interested, meaning that while they still have an interest in improving the performance of the overall system, they have their own interests and goals. The concept of a MAS is also already used in the context of the Internet, because it is suited for the non-structured, decentralized Internet [66]. A MAS is able to be self-reliant for the broad spectrum of interactions that is required online.

2.1.1 Dialogue

As with humans, various forms of dialogues are possible between agents, each with their own, distinct, purpose. A classic framework from which to define and classify dialogues stems from 1995, by Walton and Krabbe [95]. They define and discuss six types of interactions, ranging from persuasion to eristics. A final seventh dialogue type, called discovery, was added several years later [75].

Information Seeking

This type of dialogue occurs when an agent is missing information and attempts to find it from or using other agents. So the initial state is that information is needed, for which the participant's goal is to acquire or give information, depending on the role in the interaction. Finally the goal of the dialogue is to exchange information between the participants [95]. The contrast with inquiry, is that the attainment of proof is not a necessary requirement for information seeking [37].

Deliberation

The dialogue type deliberation occurs when there is an open problem that needs to be solved. The premise is that it concerns itself with future information and starts from a need for action. The different agents participating in the solving of the problem at hand can attempt to look at it from different angles, depending on their beliefs and goals. An important note is that the optimal solution to the problem for the group as a whole may not necessarily mean that it is also the optimal solution to any of the individual agents. Furthermore, agents are also expected to share their complete information and preferences [74]. For a model with dialogues that also include persuasion, this property does not hold, as agents are inclined to only present the information that furthers their cause.

Persuasion

The dialogue method persuasion occurs when there is a conflict of opinions between agents. The goal of a participant is to persuade the other party of his/her view, while the goal of the dialogue is to resolve or clarify an issue. It may be that one agent believes a certain fact and that another agent believes a different fact, where the two facts are inconsistent [37]. It may also be that an agent believes a fact and that the truthfulness of the fact is put into doubt.

Another possibility within a MAS setting, is persuasion with respect to motivational attitudes [37]. The difference for this type of persuasion, as opposed to the earlier mentioned type, is a conflict of intentions. The conflict arises from the intentions between agents where one attempts to achieve a certain goal, while the other agent attempts to reach a different goal, where the two goals are inconsistent. Another possibility remains the situation where the goal an agent attempts to achieve isn't seen as a worthwhile goal.

2.2 Trust and reputation

Due to the complexity of the terms trust and reputation, not one single definition is defined for the terms. These variances in meaning for trust and reputation are not only present in everyday life, but also appear when used in the context of a MAS. Trust and reputation are seen as both personal, subjective qualities for an agent, as well as collective quality. In a broad context, this means that an agent might have a bad reputation, which would be based on the interactions of all agents with that particular agent, while a specific single agent trusts that agent, due to their personal interactions.

When approached in the context of a MAS, there appears to be consensus that models that use reputation also use witness information. This means that information from the observations of other agents are used [78, 83]. A paper from 2005 defines two types of trust, namely reliability trust and decision trust [64]. Reliability trust is seen as a subjective probability for which an agent defines its trust in the likelihood of another agent performing a task. Decision trust is defined as the willingness of an agent to depend on another agent, given a certain situation. Reputation, however, is seen as a more general qualification of an agent [78]. It is even seen as a component of trust, instead of it being a stand-alone term [83].

Trust and reputation can be seen as either global or subjective properties. Here global is defined as the sum of the interactions between all agents. A particular agent controls the values of trust and reputation about himself. Using trust and reputation as subjective properties means that between every two agents their own sets of values for trust and reputation are formed. Depending on the

goal as well as the size of the model, one or a combination of the two properties work. For instance, on community vendor websites, one might not be familiar with the vendor of a book. Therefore, it might be preferable to choose a vendor based on the experiences of previous buyers, using their collective experience as a marker for the quality of the vendor. However, a vendor can be seen as very trustworthy when it has a high reputation, but a single personal experience or experience from a close or trusted friend or relative can mean that you don't trust the vendor.

2.2.1 Traditional approaches to interactions in a Multi-Agent System

There have been three traditional approaches in an open MAS to control the interactions between agents [83]. Since in an open MAS, behaviour of agents is uncertain, mechanisms are required to control the interactions between agents in order to protect so-called good agents from malicious ones. These three approaches are known as the security approach, the institutional approach and finally, the social approach.

Security approach

For this approach, basic structural properties are accounted for [83]. These are building blocks such as the integrity and authenticity of messages, agents, or their privacy. This is guaranteed through the usage of tools such as cryptography, digital signatures or electronic certificates. However, it does not give information about the quality of the information that is passed around.

Institutional approach

Here a more centralized approach is chosen, where a central authority observes, controls and if necessary, enforces agents' actions [83]. The central authority could even punish behaviour or actions that are undesirable. The drawback of this approach is that it requires a centralized hub from which all interactions can be monitored. Once again, the quality of the interaction is not monitored, since the quality of the interaction is subjective to the agents present at the interaction, based on their goals and beliefs.

Social approach

Finally there is a social approach, where trust and reputation are placed. It is far more decentralized than an institutional approach, since agents themselves are capable of punishing behaviour that is undesirable [83]. This can, for instance, be achieved through deciding not to interact with an agent. The method for achieving this behaviour is by requiring agents to model the behaviour of other agents. To achieve this, computational models of trust and reputation are necessary, along with a multitude of other bases of knowledge. These include the generating of social evaluations after interactions, how agents use trust and reputation to select other agents, including how this knowledge spreads along the MAS and finally how agents process this information.

2.3 Trust and reputation classification dimensions

In a multi-agent system, information is distributed in a decentralized way, meaning that all the information is not at a single location but spread out amongst the entities of the system. These are generally known as agents. A single agent, depending on its task-set, will therefore have access to different kinds of information, and will require different kinds of information in order to function.

An agent can gain or lose information based on interactions with other agents.

For these interactions between two agents, both agents have to make decisions, which influence the outcome of the interaction. An agent does not know how another agent will respond in the future, so decisions are made based on uncertain and perhaps incomplete information. Trust and reputation are means that an agent can use in order to predict the most likely response for the current interaction. These are based on earlier interactions with the environment, other agents or earlier known information. Trust and reputation are indispensable for a learning multi-agent system, because they give the agent a reasoning behind its actions and allow the agent to choose the most suitable action for the current situation. This allows the agent to learn from past interactions, creating a situationally optimal agent.

Like the definitions of trust and reputation, several classification dimensions are defined, as there does not seem to be a consensus on which dimension appears to be preferable or superior.

2.3.1 Sabater et al.'s classification.

Paradigm (Par)	Numeric (N) Cognitive (C)
Information Source (InS)	Direct Interaction (DI) Direct Observation (DO) Witness Information (WI) Sociological Information (SI) Prejudice (P)
Visibility (Vis)	Subjective (S) Global (G)
Granularity (Gra)	Context Dependent (CD) Non context dependent (NCD)
Cheating Assumptions (Che)	No cheating (L0) Bias information (L1) Cheating (L2)
Model type (Type)	Boolean (B) Continuous (C)

Table 2.1: The visualization and abbreviations for the dimension model [78].

Several ways have been defined to present the types of informations sources [78, 77]. Information sources are defined as sources from which an agent can learn. These are direct experiences, witness information, sociological information and prejudice. Direct experiences are seen as the most reliable source of information, and come in two forms. Firstly, they can be based on direct interactions with another agent, or secondly, they can be based on the observations of other agents interacting, known as direct observations. Witness information, or indirect information, is information that is obtained from other agents. The information gained is not from observation, but from what the other agent shares, based on its own direct experiences or witness information. This information can be incomplete or even false. Sociological information is knowledge based on social relations between agents and the role that an agent has in the society. Agents with a higher standing in the society can change their behavior or their interactions. However, it is a type of information source that requires rich interactions between agents. Finally there is prejudice, which is a way of assigning properties to an agent, based on information that assigns the agent to a part of a group.

The availability of trust and reputation can be defined in two ways; either as a global property

or as a subjective property. If trust and reputation are defined as a global property, it means that the information is available to all agents. If trust and reputation are defined as a subjective property, it means that the agents form their own trust and reputation about an agent, based on their own observations.

Next, Sabater et al. define granularity as the context-dependence of trust or reputation information. This can be seen as the question whether trust or reputation is limited to a single, concrete type of cases in multi-context. For instance, the trust and reputation for a person flying a plane might be different than for when he is driving a car, even though they both are about steering a vehicle.

Fourthly, Sabater et al. also define various levels of cheating behaviour, ranging from 0 to 2. This defines the models' assumptions regarding interactions of agents.

- Level 0: Agents cannot cheat.
- Level 1: Agents cannot cheat, however, they can hide or bias communicated information.
- Level 2: Agents can cheat.

Finally, the type of exchanged information, is defined as a dimension. This can be represented in two different ways: as boolean information or continuous information. Generally, models that work with probabilistic networks use boolean information, while models that are based on aggregation methods use continuous information.

2.3.2 Pinyol and Sabater-Mir's classification

Trust	Present (✓) Hybrid (∼) Absent (-)
Cognitive	Present (✓) Hybrid (∼) Absent (-)
Procedural	Present (✓) Hybrid (∼) Absent (-)
Generality	Present (✓) Hybrid (∼) Absent (-)

Table 2.2: The visualisation and abbreviations for the dimension model [83].

Pinyol, in cooperation with Sabater, created new classification dimensions, which are agent-oriented [83]. Instead of the earlier five dimensions, this classification has four, and furthermore they have not differentiated between trust and reputation, but have just called the combination trust.

The first dimension they use is called trust. Reputation has explicitly been left from the classification of the dimensions, because they believe that there is no consensus on the definitions of trust and reputation. Since there is no a clear consensus on the definition, it makes dimension classifications subjective. Based on several sources, they define trust as [40, 54, 23]:

> A process of practical reasoning that leads to the decision to interact with somebody.

Here, the models are still defined with respect to two categories, where one provides information such as rates or scores for every agent in order to assist the decision making agent. On the other hand, some models do specify how the decision should be made.

The second dimension is called the cognitive dimension, where Pinyol et al. differentiate between models that have clear representations of trust, reputation or image in terms of cognitive elements. These cognitive elements are elements like goals, beliefs, intentions, desires. This gives a clearer insight in the reasoning behind the actions of an agent.

Thirdly, there is a dimension that is called procedural dimension. It is based on the fact that models have good representations and ways of dealing with trust and reputation, however they don't explain how they bootstrap. A distinction is made between cognitive and non-cognitive models, in the sense that cognitive models explain quite well about the internal components of trust and reputation, but skip how such components are made. On the other hand, non-cognitive models sometimes do not quite explain well how their evaluations are calculated.

The fourth dimension is the generality dimension. Here, is there differentiation between models that are general versus those which have a more specific purpose. How general a model is, depends on how wide the range of domains or applications is that it can be applied to.

2.4 Classification models

Based on the above-mentioned dimensions, some models are showcased that are in line with what is relevant for IDIMAS, the model developed in this thesis. A model that is quite opposite to the IDIMAS is also shown for contrast.

2.4.1 Castelfranchi and Falcone

Natural cognitive agents' (e.g. humans') doxastic and motivational dynamics are systematically and necessarily integrated. After a few preliminary remarks on the conceptual status of the notion of goals as used here, we outline its structural correlation with beliefs. In particular, we focus on the role of supporting beliefs in goal dynamics, showing that they are necessary to regulate goal processing, determine different goal types and initiate processes of intention revision. We describe both a taxonomy and a dynamic model of belief-based goal processing, and discuss its impact for a reformulation of a theory of intention including critical comparison with some standard analyses of intending, e.g. Bratmans planning theory of intention.

Castelfranchi and Falcone introduced a model wherein a definition of trust is given [40]. They define it as both a mental state as well as a social attitude and relation. The focus of their research is on the role of beliefs in goal processing. They believe that trust is necessary in order to initiate processes of intention revision, regulate goal processing and determine different goal types [17]. The research results in the development of a model, which includes a number of features [16]. The list of features for this model can be seen in Table 2.3:

The concept of cognitive means 'mental', referring to explicit attempts at mental representations. These include motivational representations. Even though the model attempts to be both analytic and explicit, it also has implicit forms of belief and trust. These are either routine-based, mindless, automated, or felt and affect-based forms. The same constituents are potentially present, but as primitive forerunners of the explicit advanced representations. It should be an elaborate model that has a layered perspective on the concept of trust, with a fair relation between notions.

Model features
integrated
socio-cognitive
analytic and explicit
multi-factor and multi-dimensional as well as recursive
dynamic
structurally related notion
non-descriptive

Table 2.3: The preferred features for the model [16].

2.4.2 Online Reputation models

The basis of online reputation models is quite simple. Agents, or users in this case, give feedback on transactions, and by extension, on the vendors facilitating those transactions. This happens on a scale of zero to five stars. The higher the score, the better the transaction pleased the user. This system is then translated to a negative (zero to two stars), neutral (three stars), or positive (four to five stars) feedback rating. The sum of these ratings gives an insight into the reliability and likability of a vendor [38].

The rule of thumb in that situation is: the higher number of points a vendor has, the more reliable it is. However, this system does have its drawbacks. It does not take into account false reports, or actual reliability measurements. A vendor which has been present for a very long time with quite a positive reputation, could theoretically start raking in negative reviews without this overall influencing its reliability. Some websites have started taking temporal reviews into account, by offering multiple ranges of reviews in order to give an insight into the overall reliability of a vendor. Amazon shows various times of ratings, which are one month, three months, a year and finally lifetime [2].

2.4.3 Sporas and Histos

The idea of using recent feedback was already suggested by Sporas and Histos [99]. Their model only uses the most recent feedback, where they also take the current level of reputation for an agent into account. This means that an agent with high reputation will see a relative smaller reputation change for feedback than an agent with lower ratings.

An interesting conclusion of their research also was that human users were reluctant to give negative feedback to their trading partners, in the case of eBay, due to possible retaliation from their trading partner. The fear for this kind of feedback diminishes the value of a feedback system, as the scores therefore were artificially high. Furthermore, Sporas and Histos developed an algorithm which is a reputation service, in order to give their measurements for the reputation value of a user.

2.4.4 Carter et al.

The model defined by Carter et al. is a model which is based on the formalization of reputation [15]. A practical definition of reputation is used from a sociological context. The expectations placed upon an agent within a MAS are formalized, where the agents are part of an information-sharing community. According to Carter et al., the society defines a set of possible roles, which

in their case is a set of five possible roles [15].

Reputation therefore is defined using five roles. These roles are defined as: social information provider, interactivity role, content provider, administrative feedback role, longevity role.

$R = (\Gamma, \Omega, \Upsilon, \Theta, \Psi)$

Γ stands for Social Information Provider, meaning that users of a society should regularly contribute new information of their friends to the society. Ω stands for Interactivity Role, which means that users are expected to regularly use the system and maintain some form of interactivity. Υ stands for Content Provider, meaning that users should provide a society with knowledge objects that relate to their own areas of expertise. Θ stands for Administrative Feedback Role, so users should provide feedback for the functionality of various aspects of the system and finally Ψ stands for Longevity Role, meaning that users should maintain an average reputation which is positive within a society.

The reputation of an agent is defined by a weighted aggregation for the fulfillment for each of the five roles. This limits the way of calculating reputation to the context of the society. The weights used for the five roles are based on the social values within a society. Therefore are they varied, based on the wishes of the society. The final calculation is done through the average of the relevant times the values were measured, for each of the five roles. Finally, Carter et al. suggested a central authority that controls the transactions.

2.5 Trust, reputation and dialogues in an information-based MAS

Taking the previously introduced approaches and classification dimensions and applying them to the model that is being created in this research, gives insight as to the type of decisions which are made and why they are made. It further eases the design choices to be better suited.

Looking at the three approaches introduced in Section 2.2.1, there are two approaches which can be applicable to the model, of which only one eventually has been chosen, since it is the only one that is applicable. The two approaches that are relevant are the institutional and the social approach. The approach that is most applicable is the social approach. Both are discussed in order to show the distinction.

The institutional approach applies in the sense that there is a agent that keeps a list of assignments. These are requests from the another class of agents as to which books need to be verified whether they still exist or whether they have been sold. The reason this is not applicable, is that while there is a central agent that controls tasks, it does not actually punish bad behavior or actively influence the model. Agents interact with the list-keeping agent based on whether the agent needs the list-keeping agent, the list-keeping agent itself is a purely reactive agent.

Therefore, the most applicable approach is the social approach. Trust and reputation, which are quite important in our model, play an important role in the social approach. Furthermore, agents are capable of punishing undesirable behavior, since agents can decide not to interact with vendors that have a very low reputation score. A large difference is that the agents themselves are not directly evaluated on their interactions, but instead the internal match and merge of an agent decides the feedback loop towards the increasing or decreasing of a reputation score.

By taking the classification dimensions and applying them to the model that is made in this research, the following classification dimensions are found.

2.5.1 Sabater et al.'s classification

The explanation for the classification dimension of Sabater et al. can be seen at Section 2.3.1.

Paradigm (Par)	Numeric (N)
Information Source (InS)	Sociological Information (SI)
Visibility (Vis)	Global (G)
Granularity (Gra)	Context Dependent (CD)
Cheating Assumptions (Che)	No cheating (L0)
Model type (Type)	Continuous (C)

Table 2.4: The values for the experiment using the dimension model [78].

The classifications which are applicable to the model can be seen in Table 2.4. The learning of the model is based on an algorithm known as ant colony optimization. It is an algorithm where reputation and trust are defined in numeric values, so the paradigm type is numeric. There is a limited depth to the interaction between agents, since there is no negotiation between agents, but instead are interactions used as a method in order to transfer data, or give or receive assignments. The rank that a book receives within a bookkeeper, who specializes in a specific title, is key in deciding the quality of the vendor.

The information source used is sociological. The model and its agents learn based on the result of integrating information within a specific type of agent, but not based on observations of interactions. The model also doesn't learn from interactions themselves, since interactions are limited to the transferring of data or giving or receiving of assignments, and no negotiation is done. A difference from the general application of sociological information sources, is that agents don't have different standings in society, since the agents themselves don't have trust or reputation.

The visibility paradigm is global, as this is vital for the efficient running of the model. Every agent should know the reputation values of the vendors in order to make a calculated decision as to which vendor is most likely to provide the highest quality books. These values for trust and reputation for a vendor are different, dependent on the category.

The model differentiates between different categories of books. Therefore, the granularity is context dependent. Vendors can potentially have different specializations, so therefore it would be sub-optimal to create a single value defining the quality of a vendor. A vendor can have very high quality information in one category books, but potentially very low in a different category of books.

The model thrives on the correctness and clearness of information, so there is no cheating in the model. This would only have a negative effect on the quality of the model, since the reputation of vendors would not be represented correctly.

The information type used is boolean, since the model is based on probabilistic networks. Which vendors are chosen for book information are based on reputation using the Ant Colony optimization, which increases or decreases the likelihood of paths chosen based on how well traveled the path is.

2.5.2 Pinyol and Sabater-Mir's classification

It is important to note that this classification model is very agent-driven, which is something the model in the experiment is not. The classification model is nevertheless added to depict the difference between IDIMAS and these types of models.

Trust	Absent (-)
Cognitive	Absent (-)
Procedural	Hybrid (∼)
Generality	Present (✓)

Table 2.5: The values for the experiment using the dimension model [83].

Pinyol and Sabater-Mir define trust as a process of practical reasoning, see Section 2.3.2. As mentioned in Section 2.5.2, trust is seen as a numeric value that is mutated based on an algorithm and not on practical reasoning. Furthermore, interactions in our model are limited to the transferring of data or assignments, and where there is reasoning, such as deciding the rank of a book, it is done internally in the agent.

The model is not cognitive, due to the nature of the interactions, as explained in the previous paragraph. Trust and reputation are defined by actual values, as opposed to reasoned arguments. So trust and reputation are not defined in terms of cognitive elements such as goals, beliefs, intentions or desire.

The procedural dimension is a hybrid, since there is some explanation as to how the model bootstraps, but not a full explanation for every aspect. Furthermore, they are not contained within an agent, but instead stored as global values which are accessible by every agent.

Finally, the basic principle of the model is very generally usable, since it can easily be focused on a general domain for which one might want to use it. The principle of the model does not change based on which domain is chosen, it still is discovering and integrating information from unstructured big data.

Chapter 3

Ant colony optimization

3.1 Swarm Intelligence

Swarm intelligence is a behaviour where individuals perform as a distributed system. Local interactions result in a highly structured social organization [36]. The systems are self-organized, meaning that a type of general order arises from locally initiated interactions. Agents are not familiar with global behaviour.

Swarm intelligence is a concept which has interested not just computer scientists over the years, but also biologists, naturalists and many more. The interesting dynamics of colonies and the behaviour of various groups within those colonies is fascinating, because the members that make up these groups are not complex. The individual members of a group or even a colony, express a behaviour that has been described as complex, but not complex enough in itself to explain the complexity that a colony as a whole achieves. The insects are, as a group or several groups, able to perform and adapt to circumstances beyond what is expected of their capabilities [10]. The colonies show high robustness, since they are able to perform complex tasks even if large parts of a colony are wiped out, despite the members having simple local rules without any global knowledge [11] .

There are even social insect colonies known where specialized ants are able to dynamically respond to the loss of a different large specialized group. The ants then quickly adapt and perform the tasks that the large group was performing, in order to upkeep performance for the colony. It is also believed that having specialized groups of insects that perform tasks in parallel, is more effective than when the tasks are performed by unspecialized individuals.

An interesting aspect that can be taken from social insect colonies, is that complex behaviour of the group appears from seemingly simple interactions. Take for instance ants, since their behaviour is used as a basis for IDIMAS. There are certain aspects that determine the type of tasks that an ant is going to be performing. These can be physical features such as the size of the ant. One of the largest selling points of swarm intelligence is, while the basis of the algorithms is simple, the systems as a whole are flexible and versatile, as well as capable of solving non-linear design problems [98].

3.2 Real-life ant colony

The concept of Ant Colony optimization stems from observations of ants. Ants live in colonies, where they work and live together, as they are social creatures. The behavior of ants is based on the survival of the colony, and not on the survival of the individual. For this purpose, there exist several different castes of ants, each with their own special tasks and specializations.

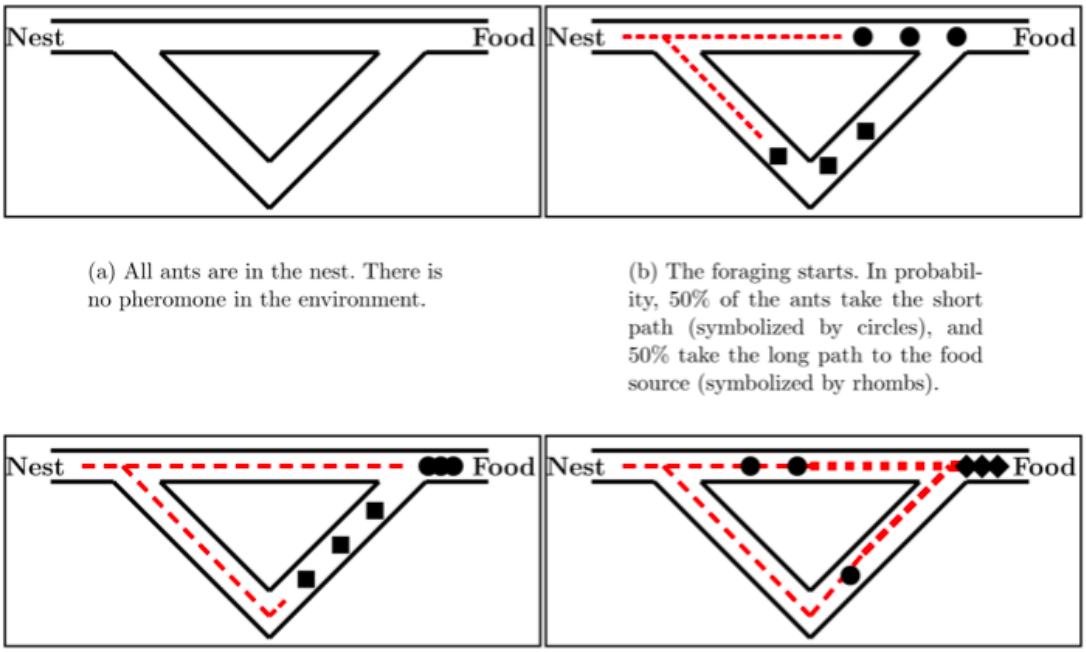
Worker ants form the basis for the ant colony optimization algorithm, due to their behavior when searching for food sources. Initially, the ants walk in a random pattern away and around the colony, in search of food sources. While an ant is moving, it leaves pheromone on the ground, in order to mark the path that it has taken [28]. This pheromone can be smelled by other ants. Once an ant finds a food source, it evaluates the quality and quantity of the food sources, after which it takes a small part back to the colony. Ants which are moving choose their way probablistically based on the strength of the pheromone path. So depending on the the quality and quantity of the food source, the ant that is making the return trip, leaves a pheromone trail. The better the quality and quantity of the food source, the stronger the pheromone trail will be. Other ants which cross this path, could follow the trail to the food source and in turn leave their own pheromones on the ground. The principle of a trace left in the environment by an action, which stimulates the action of a next ant, or the same ant, is called stigmergy [32].

3.3 Double bridge experiment

The principle of stigmergy for an ant colony was shown in 1989 in an experiment called the double bridge experiment [48]. The experiment features two locations with two possible pathways between the locations. One location was the Argentine ant nest of the colony, while the other location was a food source. The two pathways were different in length, as can be seen by Figure 3.1.

At first, the ants would randomly choose one of the pathways in their search for food. Upon finding a food source and then returning to the location, the ants choose their return route probabilistically based on the strength of the pheromone trail. Since the shorter route has a stronger trail, due to ants in the other path not yet having arrived yet, it is more likely that the ants will return through the shorter path. Figure 3.1 shows an example where three ants are returning from the food source to the colony. Two choose the path that they just came from, while one returns through the longer route. Since the two ants that use the shorter route will return sooner, this further strengthens the pheromone trail left by the ants, which increases the likelihood that the path is used by later ants, and that creates a positive feedback loop. This results in the shortest path being found. This experiment formed the basis for an algorithm that finds the fastest path between locations.

In their experiment to deduce whether the shortest path would always be found, the researchers repeated the first experiment, however, they only presented the longest path at first [48]. They let the ants scavenge food for a time in order to stabilize the pheromone trail, after which they added the short path as an alternative option to the food source. The ants stayed with the long path, confirming their expectation that the ants did show a preference for the strongest pheromone trail.



(a) All ants are in the nest. There is no pheromone in the environment.

(b) The foraging starts. In probability, 50% of the ants take the short path (symbolized by circles), and 50% take the long path to the food source (symbolized by rhombs).

(c) The ants that have taken the short path have arrived earlier at the food source. Therefore, when returning, the probability to take again the short path is higher.

(d) The pheromone trail on the short path receives, in probability, a stronger reinforcement, and the probability to take this path grows. Finally, due to the evaporation of the pheromone on the long path, the whole colony will, in probability, use the short path.

Figure 3.1: An example of an experiment setting to demonstrate the usage of the double bridge experiment. Two routes with different lengths are shown. The thickness of the red striped line depicts the strength of the pheromone trail. [7]

3.4 Ant colony optimization

3.4.1 Ant agents

Ants in an Ant Colony optimization algorithm are seen as stochastic constructive procedures that build solutions by moving on a construction graph. The problem constraints are built into an ant's constructive heuristics. Dependent on the role of the model, it can be possible for ants to also construct infeasible solutions. Ants move on a construction graph $G_C = (C, L)$, where set L completely connects components C. The components and connections have a pheromone trail τ, which is τ_i if associated with components, while it is τ_{ij} if associated with connections. Secondly, they also have a heuristic value η, which is η_i if associated with components, or η_{ij} if associated with connections. This is generally known as the *heuristic information* [36].

The pheromone trail contains long-term memory of the entire ant search process, which is updated by the ants themselves. On the other hand, the heuristic value has run-time information or prior information about the problem, which is provided by a source other than the ants themselves. The *heuristic information* generally can be seen as the cost, or estimated cost, of adding a component

or connection to the solution. These are the values used by ant's heuristic rule to decide which path to add and therefore make a probabilistic decision on how to walk on the graph. Important is that ant agents can only walk using paths that are neighboring to the current location. A complete path is defined as a tour.

3.4.2 General ACO heuristic

The algorithm is based on real ants, with a few key differences. Firstly, the pheromone is dropped when an agent finds a food source and is on its way back to the base. Secondly the agents in a model walk synchronously, while real ants move asynchronously. The principle described in Section 3.3 was first described as an algorithm in the 1990's [70, 34]. The algorithm was updated and Argentine ants were renamed to forward ants at a later stage [36].

In general, a basic Ant Colony optimization meta-heuristic functions like in 1.

Algorithm 1 Ant Colony optimization Metaheuristic [36]

```
procedure ACOMetaheuristic
    while ScheduleActivities do
        ConstructAntSolutions()
        UpdatePheromones()
        DaemonActions()
```

The ACOMetaheuristic is built around three functions; firstly there is *ConstructAntSolutions*, secondly *UpdatePheromones*, and thirdly *DaemonActions*.

ConstructAntSolutions is a function that manages a colony of ants where they build solutions by concurrently and asynchronously visitting neighboring nodes of the current node in the problem's construction graph G_C. The decision as to which node to visit is based on the stochastic local decision policy, by using the pheromone trail τ as well as the *heuristic information* η. Through this method, the ants will incrementally build solutions for the optimization problem. When building a solution, or when having a (partial) solution, the ant will evaluate the (partial) solution in order to decide through *UpdatePheromones* how many pheromones have to be deposited.

UpdatePheromones manages the updating of the pheromone trails. One of two actions can occur. The pheromone values increase due to the ant dropping pheromones on the trail, or the pheromone value for that trail decreases due to pheromone evaporation. The increment in pheromones for a trail can only occur up to a certain level, since there exists a possibility where the addition is equal to the evaporation. Increasing the pheromone value for a trail increases the likelihood for other ants to use that trail. The pheromone value increase can happen in two ways: either a single ant deposits a large number of pheromones on the trail, or a large number of ants drop a small number of pheromones on the trail. Pheromone evaporation decreases the likelihood of the trail being used by other ants, and is a useful method for forgetting, thus preventing a suboptimal solution from being found, as it encourages the search for other solutions.

DaemonActions are actions which cannot be performed by single ants. These can be actions such as adding additional pheromones to a trail in order to bias the algorithm. The daemon can observe the actions of every ant in a colony, so a general example of a daemon action is one where it selects the best performing ants from a colony and allows them to drop additional pheromones. These are ants which, for example, build the best solutions for the current iteration.

One key point of interest is that the ACOMetaheuristic does not specify the order or form in which the functions are executed. Therefore, it is possible that the functions can be executed completely in parallel and independently, or if some form of coordination is required between the functions. That this is not specified, means that the developer is free to implement the algorithm as deemed most appropriate for the current problem set.

3.4.3 Pheromone trail choice

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha}{\sum_{l \in N_i^k} \tau_{il}^\alpha} & \text{if } j \in N_i^k; \\ 0, & \text{if } j \notin N_i^k; \end{cases} \tag{3.1}$$

Equation 3.1 gives the likelihood that an ant will chose a path from the current location. The ant k calculates the likelihood of it moving from current location i to location j, using the pheromone trails τ_{ij} [36]. All the possible locations reachable from current location i are contained in neighourhood N_i^k. As is obvious, locations not in neighbourhood N_i^k are not reachable from current location i.

Also, N_i^k does not contain the location from whence the ant just came. This is done in order to prevent the ant from immediately returning to the previously visited node. However, if neighbourhood N_i^k is empty, then the current location i reaches a dead end and then the previously visited node is included in the neighbourhood. However, this can easily lead to loops, if clusters of locations with many dead ends are connected to each other.

3.4.4 Pheromone trail evaporation

$$\tau_{ij}^n = (1-\rho)\tau_{ij}^n, \forall (i,j) \in A \tag{3.2}$$

As part of the daemon actions, part of the pheromone trail evaporates. The strength of this is decided through ρ, which is a parameter between 0 and 1. This is done for all the paths in the model A. The height of the parameter ρ depicts the quickness with which the pheromones evaporate. The higher it is, the faster the pheromones evaporate.

3.4.5 Pheromone trail update

$$\tau_{ij}^{n+1} = \tau_{ij}^n + \sum_{k \in K} \Delta\tau_{ij}^k \tag{3.3}$$

After the pheromone trails have been evaporated, $\Delta\tau^k$ is added to the pheromone trail values of τ_{ij} per ant $k \in K$. The more ants that have walked on that path, the higher the addition will be. An important part for the variations of Ant Colony optimization is what is chosen as $\Delta\tau^k$. In a basic case, it is a constant value for all of the ants. For path optimization problems, only a differential path length can then help in finding the fastest path. For instance, the shorter a trail is, the more pheromones are dropped on that path by an ant.

3.5 Overview of ACO algorithms

Due to the number of variations that exist, an overview of Ant Colony optimization algorithms is given below in Table 3.1, while a few relevant ones are expanded on.

Algorithm	Authors	References
Ant System (AS)	Dorigo et al.	[31, 34]
Elitist AS	Dorigo et al.	[31]
Ant-Q	Gambardella & Dorigo	[46]
Ant Colony System (ACS)	Dorigo & Gambardella	[33]
Max - Min AS (MMAS)	Stützle & Hoos	[91]
Rank-based AS (AS_{rank} or RAS)	Bullnheimer et al.	[13]
ANTS	Maniezzo	[71]
Best-Worst AS	Cordón et al.	[25]
Population-based ACO	Blum et al.	[51, 50]
Hyper-cube Framework(HCF)	Blum and Dorigo	[9]
Beam ACO	Blum	[8]
UACOR	Liao et al.	[69]

Table 3.1: A non-exhaustive overview of ant colony optimization variants.

3.5.1 Ant System

The Ant System that we use in our research closely resembles the general ACOmetaheuristic described in Subsection 3.4.2, with the basic pheromone update rule as described in Subsection 3.3. It was first described by Dorigo in 1992, with a comprehensive addition done a few years later [31, 34].

The Ant System was first implemented for the Travelling Salesman Problem (TSP), a route optimization problem in which an agent has to visit a number of cities in most optimal route, without visiting previously visited cities. To define the problem in an abstract manner: there is a complete, undirected graph $G = (V, E)$, which contains the nodes V and edge weights E. The goal is to create a construction graph G_C that contains all the nodes only once with a minimal length. The search space S of this problem can be defined as all the potential construction graphs [7], with an objective function value $f(s)$, where $s \in S$ is calculated as the sum of all edge weights $E \in G_C$ for s.

The equation used for the path choice, as described in Equation 3.1, is used for this formula. Furthermore, also the same equation for Equation 3.2 is used [36]. The coefficient ρ must be a value below one in order to prevent a permanent accumulation of pheromones. For the updating of the pheromone trail, Equation 3.3 is used, with a special equation for $\Delta\tau_{ij}$ [34].

$$\Delta\tau_{ij}^{n} = \sum_{k \in K} \Delta\tau_{ij}^{k} \tag{3.4}$$

where $\Delta\tau_{ij}^{k}$ is a quantity per unit, dependent on the distance. It is laid on the edge (i, j) by the $k - th$ ant between time n and n+1. This value is given by the following equation:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if } k - th \text{ ant uses edge } (i,j) \text{ in its tour;} \\ 0, & \text{otherwise;} \end{cases} \tag{3.5}$$

In Equation 3.5, Q is a constant and L_k is the tour length of the $k - th$ ant. Each ant also has a *tabu* list, which contains the cities that an ant has already visited and is forbidden from visiting until n iterations have been completed. When a tour has been completed, the *tabu* list is used to calculate the ant's current solution, such as the number of cities visited and the total distance traveled.

Afterwards, the shortest path is saved and all of the *tabu* lists are emptied, after which the process repeats. The algorithm stops after a threshold n number of iterations or if all of the ants follow the same route, which is known as a *stagnation behaviour*, as the model stops searching for alternative routes.

3.5.2 Elitist

$$\tau_{ij}^{n+1} = \rho\tau_{ij}^{n} + \Delta\tau_{ij} + \Delta\tau_{ij}^{*} \tag{3.6}$$

$$\Delta\tau_{ij}^{*} = \begin{cases} \sigma\frac{Q}{L^{*}} & \text{if edge } (i,j) \text{ is part of the best current solution;} \\ 0, & \text{otherwise;} \end{cases} \tag{3.7}$$

Elitist ant system is the first improvement for Ant System (AS) (Ant System), introduced by Dorigo and expanded on by Dorigo et al. [31, 34]. The underlying concept is to give additional pheromones to the pathways found for the best tour so far. This would then increase the speed for which an optimum is found, since the pathways in the best current tour receive additional pheromones. The update rule for the elitist ant system is therefore different, with a new delta addition $\Delta\tau_{ij}^{*}$ that can be seen in Equation 3.7. Using the current best tour is an example of a daemon action, as there is no ant that can know whether their route was the fastest. The rest of the algorithm functions in the same way as described in Section 3.5.1.

This is a very successful algorithm for the route optimization problem, however, it is unsuited for IDIMAS, because the expected results are different. Elitist ant system attempts to find the single best route, while the IDIMAS works under the premise that there is no single best route. It can also be seen as the primitive form of Rank-based Ant System (RAS) (Rank-based Ant System), see Section 3.5.3, since elitist Ant System only takes into account the current best ant's route for extra pheromones, while Rank-based Ant System also takes into account the paths of the ants below that, according to the quality of their solution [35].

3.5.3 Rank-based Ant System

The Ant System has also been extended with the inclusion of ranking [13, 14]. The solutions of the ants are sorted by their tour length L_k for $k \in K$, and the contribution of the $k-th$ ant to the trail update in Equation 3.5 is dependent on the rank μ of the ant. Furthermore, also only the best ω ants are considered for the trail update [36]. This can be seen as an extension on the Elitist Ant System, seen in Section 3.5.2, since it takes more than just the single best ant route into account [35].

The underlying concept is that the over-emphasized pheromone trails that are caused by many ants using sub-optimal tours are avoided, preventing the model from reaching a local optimum. The weight of the trail σ is set at a minimum of one, and at $(\sigma - \mu)$ for the $\mu - th$ best ant. The value for ω is set to $\sigma - 1$, which results in the fact that the number of ants considered is exceeded by the number of ants by one [13].

$$\tau_{ij}^{n+1} = \rho\tau_{ij}^{n} + \Delta\tau_{ij} \tag{3.8}$$

This results in several variations in the pheromone update equations. The general update rule of Equation 3.8 is used. However, the update rule for $\Delta\tau_{ij}$ differs. These are described in Equations 3.9 and 3.10.

$$\Delta\tau_{ij} = \sum_{\mu=1}^{\omega} \Delta\tau_{ij}^{\mu} \tag{3.9}$$

$\Delta\tau_{ij}$ is calculated using only the best ω ants.

$$\Delta\tau_{ij}^{\mu} = \begin{cases} (\sigma - \mu)\frac{Q}{L_{\mu}} & \text{if the } \mu - th \text{ best ant uses edge}(i,j); \\ 0, & otherwise; \end{cases} \tag{3.10}$$

and finally $\Delta\tau_{ij}$ is calculated through the same general principle as in Equation 3.5, with the deviation that the pheromone addition is influenced by the rank of the ant.

The rest of the algorithm functions in the same way as described in Section 3.5.1.

3.6 Application to quality measurement systems

Since Ant Colony optimization is generally used for path optimization problems or general optimization problems such as task ordering, it is interesting to see where else the concept can be applied.

If one would change the perspective from finding the fastest route towards an end goal, like food for a real-life ant colony, to measuring the quality of the food based on how well it is received back at the colony. This can then be used as a basis to influence the likelihood of ants returning to that food source. In our model, the vendors are the food sources and the books are the food in this thesis.

The quality of the books then define the quality of the food source, and thus how likely it is that an ant will make a return visit, or how likely it is for a different ant to visit. Using swarm intelligence, the system is then theoretically capable of rating vendors, by seeing which vendors have high pheromone values associated with them. A high pheromone value means a well-visited or high-quality vendor, which can be relevant if an ant is in search of new books. An individual agent would not know the value of that specific vendor, however, the colony as collective would know which vendors are interesting to visit, due to their high likelihood thanks to pheromone values.

This can be relevant in several ways. If a knowledge base is constructed and is required to be maintained, then the agents can learn which food sources provides qualitative information that provide an improvement to the existing knowledge base. Furthermore, the knowledge base with its agents can also be used to assess the relevance and truthfulness of the information that it holds, since high-quality food sources will more likely be visited than less high-quality food sources, keeping the knowledge base more up to date, albeit smaller.

Part II

Text mining

Chapter 4

Information discovery

4.1 Issues

In the field of data mining and it sub-field of crawlers, there are still on-going issues. Because there is such a large variety in the information that can be crawled as well as methodologies that can be used, this section will name some of the relevant issues for this thesis [52].

4.1.1 Mining methodology

The approach required for the issue is dependent on a number of factors, namely the domain where information is being retrieved from.

- Mining knowledge in multidimensional space;
- Interdisciplinary data mining;
- Boosting the power of discovery in a networked environment;
- Handling uncertainty, noise or incompleteness of data.

One of the issues within mining methodology is mining knowledge in a multidimensional space. This entails finding patterns at various levels of abstraction of the database, as well as using various combinations of variables. This is also known as (exploratory) multidimensional mining and is useful in order to reduce the complexity of the data.

Next, there is interdisciplinary data mining. The power and performance of data mining can be enhanced using a method for information retrieval that is specific to the domain or discipline of the data. For example, if the data contains natural language, Natural Language Processing (NLP) can be used in order to help with data mining.

In networked environments, information is split amongst various databases or websites with semantic links between them. Knowledge gained from one database could be used to to enhance discovery of a linked database.

Finally, there is handling of uncertainty, noise or incompleteness of data. Since the data that was obtained for this thesis often has been created by humans, mistakes can be made, causing the data to be either noisy or incomplete. An issue remains whether the data found is correct beyond reasonable doubt.

4.1.2 User interaction

Overall, user interaction is not that relevant for the current project, due to the nature of the storage of the data. However, there is one aspect that is relevant, which is discussed in [52]:

- Incorporation of background knowledge.

The incorporation of background knowledge ties in with the point made earlier about interdisciplinary data mining. However, the difference is that the knowledge is not based on data mining algorithms or information retrieval methods specific to that domain, but instead is based on experiences of a user with that domain.

4.1.3 Efficiency and scalability

Again, only the relevant aspects are taken into account, based on the issues as stated by Han [52]:

- Ẽfficiency and scalability of data mining algorithms;
- Parallel, distributed, and incremental mining algorithms.

A performance-based issue is the efficiency and scalability of data mining algorithms. So the running time of algorithms has to be short, predictable and acceptable by applications. This is necessary because of potentially huge amounts of data in data storage or dynamic data streams.

Since databases can be large with a wide distribution, this could increase the computational complexity required of data mining methods. In order to keep data mining scalable, it is necessary to create parallel, distributed, or incremental mining algorithms. These algorithms can effectively partition the data into processable-sized parts, which can then be processed in parallel. Afterwards, the found patterns can be processed and merged back into a final database.

4.1.4 Rapid data turnover

Finally, there is one issue that is relevant for the type of data that is being used in this research.

- Limited availability of data before disappearance.

There is no fixed window of time that information will be available for the data in this thesis. The data resembles that of data streams, however, there is a difference with data streams. Data streams process information by looking at changes within a specific window of time, or process the information on a keep or drop basis [45]. This means that information is presented once, and then a choice is made if the data is used, otherwise it is lost forever.

The domain of IDIMAS contains information that is available for the duration of the sale of the object. So there is a window of time for how long information is available. An example of information disappearing is when the book is sold. A vendor will then take it out of its web shop. It does have a large overlap with data streams, because the information is touched, evaluated and then discarded or saved [45]. The difference between regular data streams and the data in the domain of IDIMAS is that the number of streams is infinite, and that they all have their own window of opportunity in which relevant information is present. However, it is impossible to tap into all these data streams at the same time, nor is it known beforehand how long information will be available.

4.2 Information retrieval

Information retrieval for this thesis is done mainly through the use of crawlers. Crawling is described as the process of automatically discovering and exploring websites. This is normally done from the client-side of a website, where the crawler is simulating the actions of a user. In the early days of crawlers, they were used to collect statistics of the web [79]. However, the Internet has grown explosively due to the amount of information that is available. This growth in data comes with its own issues. It has made it impossible for users of the Internet to stay up to date with every change and addition to the information that interests them. This has caused web crawlers to become an indispensable tool to search engines and database owners, because it allows them to automatically update their databases. Because a website can only be found in a search engine if that search engine is aware of its existence, it is important to index as many websites as possible.

4.2.1 Web mining

Since the Internet contains unstructured information, the goal for information retrieval is to correctly evaluate and index web pages [24]. Web mining for information retrieval is distinguished into three categories, which are: web content mining, web structure mining, and web usage mining [90, 53, 92].

Web content mining is the recognition and analysis of the contents of a web page. It focuses on all types of content for mining, from text to images to sound. It can be used to cluster pages based on topics or types of information. Clustering can range from clustering pages that are about SpaceX to differentiating between product overview and product detail pages. This information can then be used to direct crawlers or extract relevant information from a website. Historically, this type of work has been very labor expensive. These websites could have frequent updates, changing the structure of information presentation of a website, and that resulted in the Traditional crawler (Trad crawler) requiring an update in order to cope with the new website. This shows the issue of scalability, as described in Section 4.1.3. If every website required its own specialized crawler, increasing the number of websites that are crawled is an almost impossible task, because at one point more maintenance is required in order to deal with changing sites than expanding the network of crawlers.

Web structure mining is also useful in directing crawlers, because it analyses the hyper links, forms or edges between web pages in order to learn about the structure of websites and which pages are important [76]. If the crawlers automatically learn about the general structure of the websites that they visit, then that allows them to be more generalized.

Finally, there is web usage mining, a field that focuses on the behaviour of web users [88, 92]. This field is used to analyze the behaviour of users in order to get a better return from the users visiting their website. This can range from analyzing the sequence of page views and clicks from users, to web shops attempting to recognize what makes users more likely to buy products from their website.

This field focuses on finding relevant information in unstructured data. Finding this data can be through keywords or also by looking at words that are closely connected to what the user is searching for [67]. If there are multiple matches, which is not unlikely when searching for information on the Internet, then a user should get the most relevant web page.

For search engines, multiple issues then arise: how do you define what is most relevant for a user, and how do you differentiate between the found results. For business intelligence, the problem is different. The question changes to finding the relevant information on a page, relative to what is needed. It is a steadily growing field with roots in a wide variety of applications [18].

4.2.2 Crawlers

As the Internet evolved, the methods for presenting information on websites also evolved [94]. Websites became larger, more intensive and more interactive. This has required crawlers to become more intelligent, because information became harder to find. So it means that crawlers had to evolve in order to deal with new ways that information was presented.

Traditional crawler

The traditional type of crawler is known as a Trad crawler. They can be seen as simplistic parsers. These crawlers start from a set of Seed URL (seed URL). The Trad crawler then goes through the web addresses, where it downloads the web pages associated with these web address (URL)s. The linked URLs are extracted from these web pages and added to the queue of the set of seed URL. This process continues iteratively until all of the content reachable from the set of seed URL is reached. Real-life practice limits the crawler to a number of iterations that it was allowed to do. There could also be a classifier on the URLs that were added to the set of seed URL, in order to guide the crawlers. An example of a classifier is to contain new URLs to stay within the current second-level domain, as well as limiting the addition to new URLs. Second-level domains are the part directly left of a first-level domain, so if there is a web address $http://www.rug.nl$, then nl is the first-level domain, and rug would be the second-level domain.

A big issue in the field of crawlers is scalability, as mentioned in Section 4.1.3. In order to decide whether a URL had to be added to the list of an seed URL, it had to be compared to the items that already are in that list, as well as the list of URLs that have already been crawled. Because the number of crawled URLs increases, the process becomes increasingly time-consuming. One of the suggested solutions was through the use of parallel crawlers, because this allows for the parallel retrieval of information [19]. This issue of efficient crawling was one that kept large search engine companies busy for a long time, since it became a bottleneck for guaranteeing the freshness of the pages that users would want to visit [79]. A methodology for ranking web pages was through the use of the number of URLs leading in and out of the website. A more active website is likely to have more links than a lesser one. But if that information was not up to date, then the page could have an incorrect place on the ranking list. This algorithm was known as PageRank, and was used to improve the quality of search engines [12].

Another improvement in the ranking of web pages, was the use of keywords [72, 86]. A more complex method was needed for the correct ranking of pages, because there were a large number of junk websites and those did not provide relevant information for a user.

Deep crawler

However, as the size of the Internet grew, so did the complexity of the web pages. The Trad crawler assumed that all the relevant information was reachable through the URLs. Websites became interactive, where they could act as portals for the user, who could post queries and receive results from a database on the back-end. This meant that crawling no longer was a matter of finding and following the correct URLs, but also that web pages could be generated on the spot,

based on the submissions given by a user. To address this issue, Deep-web crawler (DW crawler) was created.

The main change from a Trad crawler is that edges between web pages, or nodes, were no longer defined as hyper links leading from web page i to web page j. Instead, edges were defined to exist when submitting a form in web page i led to web page j. This new concept of a DW crawler is not flexible, since it is limited by the forms that they can submit.

Rich crawler

The interactions between server and client also became more complex, with more and more actions being performed at the client side. While this does not pose any difficulties for a regular user, it is fatal for a Trad crawler or a DW crawler. The interactive web pages were typically represented in the form of a Document Object Model (DOM), and actions by a user would change the state within the DOM, without changing the URL. As both the Trad crawler and the DW crawler rely on the changing of a URL, any changes in the web page would remain invisible.

Again, a more intelligent approach was required. The new approach was to load the web page into a virtual browser, and to analyze the code of the web page. Several different approaches have been created in order to find an optimal solution [42, 30]. The core concept is that the code of the web page is analyzed for potential events that can alter the DOM. The events which are likely to be useful are then executed and the results of those are also analyzed. This type of crawler finishes if there are no more URLs or states to process.

4.3 Information extraction

Once the relevant pages are found, the problem has not yet been solved. While it is known that the pages most likely contain relevant information, the information still has to be extracted from the page. The information on the page is still unstructured, but may contain clues as to the structure of the page. Relevant information can be in a table, but it can also be in part of a sentence. It is still presented through descriptions, where natural language is used in order to describe the specifics of a product. If one has special access, such as through a data api, it might also be possible that the data retrieved is semi-structured. Then the structure of the information retrieved is known beforehand.

The issue remains that the information retrieved is represented in natural language, for which NLP is required [20]. Natural language processing is required to not only find structure in the sentences, but also meaning [20, 22].

Information that is easy for humans to process, such as cities, person names or more complex information such as task descriptions, is very difficult to process for machines [58]. The information required is context-based and implicit, because it is not labeled.

4.3.1 Named entity recognition

To solve these problems, a method like Named Entity Recognition (NER) is required. This field deals with finding and classifying relevant information in text. A well known and widely used toolkit for NLP and NER is the Stanford CoreNLP toolkit [73]. It is a challenging field due to

the variety of information in text. State of the art research generally is focused on using either user-chosen features or gazetteers [80, 29]. These can be features such as capitalization, length of words, or gazetteers containing keywords that are relevant for the task at hand. Furthermore, NER is generally limited to a specific language. All of this results in low generalization. Added to this is the general structure of language. Words in a sentence have a general structure, where the location of each word is influenced by its neighbors. Therefore, a complex approach is needed that takes sequence labeling into account.

Using a Conditional Random Field (CRF) is a popular approach for NER, because it takes the sequence of labels into account [58, 93]. Popular implementations of CRF methods for NER are linear chain CRF and semi-Markov CRF [87]. A different, more recent approach is through the use of neural networks [21, 22]. A recent study was able to achieve state-of-the-art performance in NER without the use of language-specific knowledge or resources [44].

Chapter 5

Information integration

In an ideal world, the information that is stored in a database is clean and correct. Clean data is data that is up to date and complete. So when a query is sent to a database, the relevant and correct information will always be retrieved.

Unfortunately, this is not the case in the real world. Information can be incomplete or outdated. This is known as "dirty data", meaning that the data does not correctly represent the real world information that it stands for. The reason that dirty data exists, is due to the transience of the data, as well as human-made errors.

Therefore, it is adamant that the information found is tested for quality and if necessary, deduplicated. It is also important that the relevant parts are correctly integrated. The basis assumption for this chapter is that the information is already locally categorized in a database, meaning that the information has been extracted in relation to what was found during web content mining, but not yet compared to the same-level information.

5.1 Data quality dimensions

Data quality is seen as such a serious issue that its dimensions have been defined specifically[41, 4]. These defined issues handle various aspects of data quality and management. It has been standardized by ISO as to what the definition of these dimensions are in 2008 [89].

5.1.1 Consistency

Consistency refers to the validity and integrity of data that represent real world information. It concerns data inconsistencies and conflicts in the database, both intra and inter data objects. Intra data object consistency is the consistency of information within a data object. It can be context dependent however, such as with book information. An example of this is a book's title and author combination. It is important that the author and title combination is possible, and not for example that the author of "Harry Potter and the Goblet of Fire" is "J.R.R. Tolkien" according to the database. Inter data object consistency is concerned with keeping conflicts between data objects to a minimum. Keeping in line with the previous example of "Harry Potter and the Goblet of Fire", if there are two data objects where one had the author "J.R.R. Tolkien", and the other "J.K. Rowling", it is highly unlikely that both data objects are correct.

5.1.2 Deduplication

Deduplication handles the possibility of having several tuples in the database and how to best resolve these tuples. Duplicate information can best be described as data objects that in some form refer to the same real world information.

This can be a difficult issue to resolve, as information in data objects can be inconsistent, meaning that pairwise comparisons of data objects will not give a clear resolution whether two data objects refer to the same real world information. An additional problem is that for large databases, it is too cost-expensive to compare every feature of every data object.

5.1.3 Accuracy

Data accuracy refers to the closeness of the information in the data object to the real world information. This issue is hard to measure accurately, as generally context is needed to assess the accuracy of the information. Keeping with the example given in Section 5.1.1, if it is not known what the correct author should be for a book, then it is difficult to deduce which option is the correct one.

5.1.4 Completeness

Completeness refers to whether information in a data object is complete, meaning that all the information that is required to accurately and correctly answer a question is present. If that is not the case, then the other issues become harder to solve. The problem can extend beyond data objects being incomplete, as complete data objects can be missing. This could lead to querying objects in the database that do not exist.

There are two assumptions in this scenario, theClosed World Assumption (CWA) and the Open World Assumption (OWA). CWA assumes that all the data objects are present, but that some of the values within the data objects are missing. OWA, in addition to values within data objects missing, assumes that complete data objects may also be missing.

5.1.5 Currency

Another dimension of data quality is currency, which refers to timestamps associated with values in a data object. However, not all timestamps are valid, meaning that they do not correctly reflect the time for which the objects they represent are accurate. This can become an issue when one might create a coherent time-line of the changes within a data object. Or a different example, when a data object contains very different timestamps, it raises the question of whether the information itself is still accurate, or that the timestamps require updating.

Another issue is that not all databases use the same formatting for timestamps, making it harder to use and compare timestamps. This is an issue that can be resolved, even without the presence of timestamps. However, it is necessary to make use of context in such a case. Staying in line with the example from Section 5.1.1, it could be important to know which book version is older, a “Harry Potter and the Goblet of Fire” first edition, or a fourth edition book. If the information does not contain timestamps, then explicit knowledge is needed that a first edition is made earlier than a fourth edition.

5.1.6 Believability

Believability describes how users perceive the information that is presented to them. If the information presented at one point in time had issues and caused issues, then that might result in users no longer trusting the information, even though all the bugs and inconsistencies have been resolved. A database may hold the most correct information possible, if the users don't believe in its validity, then the database is worthless. This is a very different issue than the earlier mentioned issues: it has nothing to do with the quality of the database itself, but all with user perception.

5.1.7 Interpretability

Finally, there is the issue of interpretability. This is related to believability, because it deals with the users and not the quality of the content in the database. Interpretability concerns itself with the readability of the information. If the data in the database is of high quality, but it is also very hard to understand, then the interpretability of that database will be low. Users might perceive the database as low quality.

5.2 Object identification

One of the most important aspects of data integration is object identification [4]. This has also been described as the entity identification problem [52]. The field concerns itself with finding and matching the same objects within or between databases. Because there are generally no standard formats for the method of data representation in a field, there is a lot of variation in the way that information can be presented. Consider the field of rare books, when finding and matching books based on author. There are vendors that use the format "J.R.R. Tolkien" to describe an author, while other vendors use "Tolkien, J.R.R.". Both are correct, but how is it decided what constitutes the same name, but formatted in a different way? Furthermore, based on the format that is chosen for the merged database, one of the descriptions is more correct than the other. In this scenario, it is also assumed that the data is already clean. In a real world situation, this is unlikely.

Another issue that arises is when databases have multiple objects that point to the same real world representation. If a vendor has a single book eight times in their database, with almost exactly the same variables for the features, does that mean that the vendor has eight copies of a book, or is the same book token represented multiple times? This is also influenced by dirty data, because information could be misrepresented at various levels of addition [41, 47]. This is a different situation than data exchange, where information is restructured, with possible loss in content [39].

On a basic level, object identification is described as record linkage, where it focuses on matching activity on simple structured data, such as files or relations. It is generally prior knowledge that the information is related in these cases. The goal of record linkage is to reduce multiple representations of objects to the same real-world object to a single representation [52]. Record linkage turns into object identification when the data becomes more complex. At that point, data is represented as semi-structured or complex-structured data, where representations or different structures are used to represent the data. Relevant meta-information, such as keys, can be used in order to improve object identification, because the methods required for object identification are more sophisticated than for record linkage.

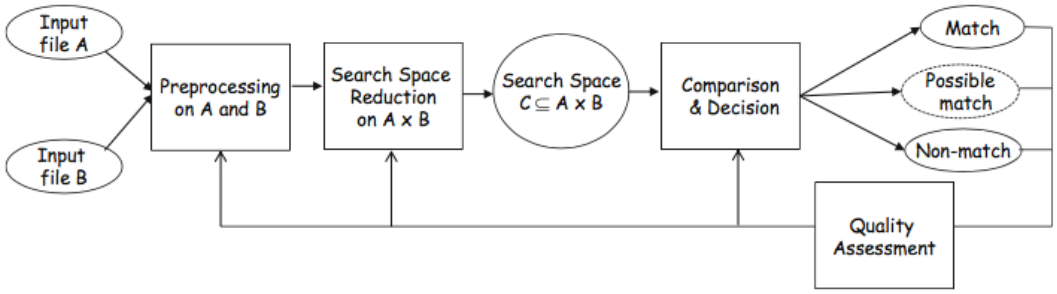


Figure 5.1: A top level process description for the process of object identification [4].

Figure 5.1 describes the general process used for object identification. Input file A and B can also be used to describe objects and not complete files. Afterwards, preprocessing is done in order to correct obvious errors and to standardize the data. Preprocessing can include transformations such as changing text in the data to lowercase, removing characters that are impossible for the specific feature, or more complex issues such as conflict resolution for schemas. These transformations cause that the search space is reduced. Next, the information is compared and merged. There are two possible solutions when objects are found to be matching. It is not always necessary to merge the records if a linkage has been found. Dependent on the goals of the process, it can also be possible that multiple objects are fused together in other to create a single object, which is a process that is called fusion. After the objects have been matched and if necessary, merged, the database is assessed for quality, based on predefined goals [4].

5.2.1 Matching

As seen in Figure 5.1, part of the process is the comparison of and decision about objects. This is done after the search space has been reduced, in order to reduce the complexity of the problem and to remove redundant features. There are several algorithms that can be used to achieve this, ranging from schema mapping to clustering to complex classification algorithms [52].

Schema mapping

Schema mapping is also used for preprocessing in order to standardize the data. In that situation, schema mapping is used for the normalization of the variables of features, such as reducing a book binding to one of a predefined group of variables. It can also be used when matching objects [41]. In that case, it is applied on a higher level, by matching features such as column headers between input files [84]. Major issues of this approach are the labor required to create the schemas and conflict resolution for dirty data.

There are a few possible approaches, because it is not known beforehand which features match. One method is to create a sample or predefined schema, where it is annotated what the relation is between features, and then to let a program finish the rest. This is very specific to the files or objects that are being compared at that moment, and is therefore not scalable. It is also heavily influenced by the level of dirty data that is present in the files or objects. It is assumed that schemas are complete, meaning that the schema is expected to contain all possible options that can be presented. If the data is dirty, this means that there will be more options, causing the effectiveness of the schema to be lower. The other option is that the schema will not be complete,

since there are variants of schema inputs present that do not match. That is why the data has to be cleaned beforehand [47].

Distance-based comparison

Another approach for comparison, especially in the scenario of string comparison, is using distance-based comparison. Dependent on which algorithm is used, a different score is used to measure the difference between two strings. However, the bottom line in every situation is that the higher the overlap, the smaller the number of mutations required to change one string into the other, the better.

One of the most well known techniques is the *Levenshtein distance* [68]. It measures the number of mutations required to go from one string to the other in absolute terms. For instance, going from the word "book" to "books" has a *Levenshtein distance* of 1, because one mutation, the addition of s, has to be done.

$$d_w = d_j + \lambda p(1 - d_j) \tag{5.1}$$

Another approach, the one used in this thesis, is the *Jaro-Winkler distance* [96]. It is an extended version of the *Jaro distance*, for which the equation can be seen in Equation 5.2 [61, 60]. The *Jaro-Winkler distance* takes the prefix of a string into account, giving a higher rating to strings that overlap in the beginning. The equation for *Jaro-Winkler distance* can be seen in Equation 5.1.

$$d_j = \begin{cases} \frac{1}{3}(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}) \\ 0, & \text{if } m = 0; \end{cases} \tag{5.2}$$

For Equation 5.2, the *Jaro-distance* $d_j$is calculated. m is the number of matching characters, using the formula that can be seen in Equation 5.3. The values s_1 and s_2 are the lengths of the two strings, and t is half the number of transpositions required.

$$m = \left\lfloor \frac{max(|s_1|, |s_2|)}{2} \right\rfloor - 1 \tag{5.3}$$

Then for the *Jaro-Winkler distance*, seen in Equation 5.1, Equation 5.2 is used, with the addition of $\lambda p(1 - d_j)$, where λ stands for the length of the common prefix, generally up to four characters. p is a constant scaling factor and depicts the amount of influence that that a matching prefix will have. This value is generally defaulted at 0.1, and can be increased to a maximum of 0.25, in order to ascertain that the resulting score is within the range of zero to one.

Clustering

An issue with handling text in a database, is that it is not formatted in a normalized way. Furthermore, data can be misrepresented or spelled wrong. For instance, a book normally has a binding. That can be a description such as "hardcover" or "softcover". Unfortunately, humans do not write and describe information the same way. The description "hardcover" can also be described as "Hardcover", "hard cover", or simply "h". Transforming the information to the relevant and correct description therefore is key. These types of clustering can be described as data reduction clustering, since they aim to decrease the number of variables that a feature is able to have [52].

An appropriate method to achieve this is clustering, where items are grouped based on overlapping features [59]. The goal is to create subsets, or clusters, of a database, so that the items within a cluster are as similar as possible, while being as dissimilar as possible between clusters. When clustering, the class labels are not used. This is due to the possibility of class labels not being present, or the size of the dataset is not known. Some popular clustering methods such as K-means require to known beforehand how many clusters are available. When dealing with real-world information, this information generally is not known.

When clustering information, generally a mean of a cluster is calculated, and new data objects are compared to the means of every cluster. They are then added to the cluster where they have the best fit. After each iteration, new means are calculated for the clusters, and the process is repeated. However, if the information is not numeric but text, the situation becomes more complex. A mean of a cluster cannot be calculated, because the information used is represented as words. Added to this dilemma is the fact that it is not known beforehand how many clusters are necessary.

Therefore, it becomes important to be able to dynamically create clusters, when the need arises for them. Clusters are important because they create an opportunity to reduce the within-feature dimensionality of a database, therefore reducing the complexity of the database. This makes it easier to compare and classify data objects within a database. Generally, clustering is done for numeric data, but it is also important that is is usable for other types of information, such as categorical, ordinal, graphs, sequences or even documents.

In IDIMAS, a different kind of clustering is used. Some features of a data object can clearly be reduced to a classified few variables, whereas others, such as author or title, are more difficult. Because there are millions of books, with new books appearing every minute, it is impossible to have an up-to-date complete knowledge of the book titles and authors at any time. Therefore, a different approach is required. The problem for this thesis is that information is represented as text. The mean of a cluster is therefore taken by choosing an item from the cluster, known as an exemplar. Exemplars have also been used in order to cluster information such as sentences, based on a method called affinity propagation, that takes as input the similarity between points of data [43]. This defines clusters by passing real-valued messages between the data points, until clusters remain.

5.2.2 Fusion

When going beyond record linkage and object identification and the goal is to to create a minimal and up to date set of two input sets, then records have to be removed, either partially or completely. This can be achieved through either choosing an exemplar, which is described in Section 5.2.1, or using fusion. Data fusion is described as a step in the data integration process, after steps such as matching and deduplication, and focuses on dealing with contradictory items in a dataset [6].

When choosing an exemplar, the most representative object is chosen from a cluster of objects that all depict the same real-world object. The basis on which an exemplar is chosen can differ. One method that is suggested, that also deals with the rapidly changing world of records, uses incremental linkage in order to create and match records [49]. That research uses correlation clustering and db-index clustering in order to identify and fuse objects [3, 27].

This can be done on an object-based level, but also on the features within an object. Then for every feature the most representative option is chosen.

5.3 Information warehouse

When dealing with a lot of information, the way that the information is stored becomes important. Efficient method of storage will lead to fast processes for the retrieval of information.

Since information warehouses are quite a subject by itself, it will not be discussed in IDIMAS. There are several papers that provide excellent starting points for research on the matter, however for IDIMAS is assumed that the storage methods are sufficient [55, 56, 65].

5.4 Discovery and Integrations

Overall, the problems facing information discovery and integration are very diverse. Finding the relevant pages, then extracting relevant information from said relevant pages are only part of the issue. Once that has been done, the information still needs to be normalized and processed in order to detect anomalies and make it presentable for information fusion. In addition to text mining, the information itself and the sources of the information need to be judged based on the quality of what was provided, requiring a learning algorithm that can differentiate between various levels of quality.

The past Parts I and II discussed the theoretical challenges and possibilities within the fields of trust, ant colony optimization, information discovery and information integration. The coming chapters will discuss the actual implementation of such a system.

Part III

Experiment

Chapter 6

Data Acquisition

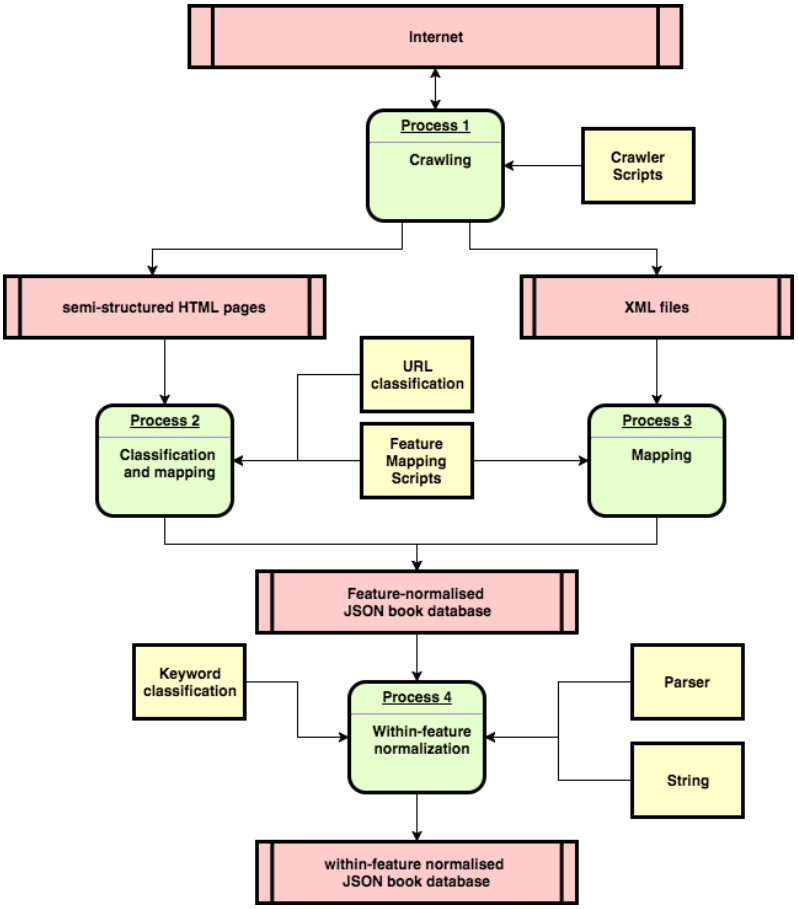


Figure 6.1: The process from unstructured web data to a database.

6.1 Web origin of the data

The database is created using crawlers which retrieve information from two websites. After crawling, the relevant information is extracted using parsers. Since the structure of the websites was

known on beforehand, crawlers and parsers were made specific to these websites.

Because the number of books available online is near immeasurable, a selection of categories as well as locations was made for the books that would be used. For this thesis, books with the categories *fiction* or *literature* were chosen, generally from United States-located vendors.

The two data sources used are Abebooks and Biblio. They are both market-place websites, meaning that they act as intermediate between vendors and buyers. Both of the data sources do not sell books themselves: they both are communities where vendors can sell their books.

The preference of using market-place websites over vendor websites, is that it creates a larger variance in the available vendors. These local vendors have shops in various states, but also offer their books online. This creates a lot of different information sources, which means that a MAS can look beyond the market-place website, and at the individual vendors themselves.

Furthermore, it is likely that vendors that have their own website with books, will also have a presence on market-place websites. This makes the effort of creating crawlers and parsers for specific vendors a costly, and pointless task. The market-place websites were chosen based on their availability of books and range of vendors. Thanks to their variance in available books, from common to quite rare, they have a lot of books from which to choose, creating a varied database.

6.2 Acquisition

The following sections are about the methods that were required to retrieve the information from the two websites. Abebooks allowed access to their data api, whereas for Biblio the actual pages of their website had to be crawled. This required different approaches for information retrieval.

The crawlers found several duplicate books, since several categories of crawlers were run in parallel sessions, without access to a shared queue and history. A full list of crawlers that ran in parallel, along with the final number of books per crawler per week can be seen in Appendix A.2.

6.2.1 Crawlers

The data has been acquired using crawlers. They crawled the pages of the website in the domains of fiction and literature. For Biblio, the seeds that were selected were specific to sub-domains within the website, in order to ensure that the crawler would stay contained within that specific sub-domain until completion. For Abebooks, a different approach had to be chosen. Since the seed URLs were data api requests, the options were stated explicitly in the keywords.

Using site-specific crawlers is an older method, however, sufficient for the issue at hand. Since the number of different websites that was crawled was low, this meant that creating crawlers for the specific websites was less time-consuming than creating a general crawler capable of crawling a wide range of websites.

The implementation has been done using a multi-threaded Python crawler, which ran on several systems. It is based on the implementation by Bucky Roberts, which then was adapted to suit the specific needs for the research in this thesis [85]. When crawling websites, it is important to

take crawler etiquette into account, because overloading a website with requests can get the IP address of the crawler banned. It might interfere with actual users attempting to use the website. Therefore, the crawlers were set up with a limited number of requests per second, as well as set up to crawl from various IP addresses, in order to reduce the workload on the website.

The crawlers for Biblio searched in a breadth-first way, because new URLs are added to the end of the queue, while those in the front are removed. The queue and history of URLs crawled are periodically saved in case of accidental crawler termination. Because the URLs for Biblio are built in a structured format, creating a crawler that only took the URLs of listing pages and saving both the detail and listing pages was simple. The information that is retrieved, is stored in an HTML format.

The crawler ran on eight threads, with a sleep period between two and eight seconds between requests, for an average of five seconds. The actual time between requests per thread usually was higher, because the thread also had to process the information that was retrieved. On average, the website that was crawled received about one request per second.

6.2.2 Data Api

Abebooks provided access to their data api, which allowed for 25 requests per second. Their database would return a maximum of 200 books per request. Due to the number of possible requests; the individual titles, authors and vendors were added as a new query to the queue. Before the queries are added, they are checked to see whether they don't already exist in the queue or have been crawled. The crawler also went in a breadth-first search, because the new URLs were added at the end of the queue.

The advantage of using this approach over web page crawling was the higher number of books that could be found. The market-place of Abebooks also contained more books than Biblio. This had its drawbacks, because the crawler speed is lower as the size of the URL lists are increased. Since there were more request possible, the lists of the queue and URLs that had already been crawled was longer, which took more time.

Another advantage of using a data api, is that the data is returned in a normalized format (xml), allowing for an easier transition to the format that the thesis required.

One large drawback of this approach is that it requires permission from the owner of the api. Because this search was for academic purposes, Abebooks was helpful, however is it generally a much harder resource to get access to, because it is of economic value to the owner. Under normal circumstances, the incentive for an owner to provide access is free of charge not that high.

6.3 Normalization

The data that is obtained, is unstructured, as there are no protocols between vendors on how to write down the information used to describe all of the relevant information for a book. Unfortunately, this means that vendors, or the people filling out the book information, use different styles or writing methods. This adds a layer of complexity to the database, since the information that is crawled needs additional normalization before it is possible to use it for comparison.

Book feature	Type	Normalized
isbn10	Linked	Yes
isbn13	Linked	Yes
author	Linked	No
title	Linked	No
category	Unlinked	Yes
publisher	Unlinked	Yes
publication_date	Unlinked	Yes
pages	Unlinked	Yes
prices	Unlinked	Yes
quantity	Unlinked	Yes
vendor	Unlinked	Yes
vendor organization	Unlinked	Yes
vendor location	Unlinked	Yes
binding	Unlinked	No
edition	Unlinked	No
signed by author	Unlinked	No
book_condition	Unlinked	No
dustjacket_condition	Unlinked	No

Table 6.1: The book template used for the books in the model, including their *Type* and whether they are normalized before entering the model.

This process of normalization therefore is a multiple-fold process, where part of it was done outside the model, and part of it was done inside the model. This was done in order to reduce the complexity of the model. Nevertheless, some of the normalization was done within the model because the amount of normalization required for that feature gave insight into the quality of the book. The retrieved data had a lot of information from one vendor called *TRANSMEDIA HOLDING*, which showed improbable book information of questionable quality. That vendor has therefore been omitted from the model. The number of books that this single vendor provided would also skew the results towards that vendor, as they are a significant part of the database. When different crawler coverages are tested, then it would be likely that this vendor would end up with a high ranking, because of the high number of books available.

The normalization essentially can be seen as a two-fold process, of which parts of the second fold are done within the model. Firstly, the information is reduced and reformatted to the chosen relevant book features. Next the variables within the chosen book features are reduced to a form and size that is usable to deduce the value of those features for that specific book.

6.3.1 Feature reduction

Biblio and Abebooks required different approaches for feature reduction. Because the data from Biblio was in unstructured HTML pages, it required a more extensive information extraction approach than Abebooks, who supplied the book information in an readable XML format.

Biblio parsing

The pages that are crawled are HTML pages, so the structure of a web page is dependent on a website. A parser therefore has been written that, for the chosen features, finds that information and stores it in a JSON format.

Two types of pages are crawled. Either the page is a listing page, or a detail page. A listing page is a page contains an partial overview of relevant information, such as the information for multiple books. This information is not complete and is also to be found on the detail page, that contains more detailed information on a specific book. The listing page, however, does contain links that lead to the detail page of a book, where all of the relevant information for that book is found. The listing pages were discarded for the normalization, only the detail pages were used. All the information was taken from one website, which meant that a specialized parser was sufficient to find the relevant information.

Abebooks parsing

The situation is less complex for Abebooks. The books already have features that have been defined by Abebooks. The only parsing required is to map the features. This was done by running through the XML features, and then by mapping the XML features to the correct book features.

6.3.2 Linked feature normalization

Once the information is formatted to the correct features, it has to be normalized further, in order to reduce the number of variables within the features. For the features, a distinction is made between two types of features: linked and unlinked. Table 6.1 depicts which features are linked and unlinked.

Type features such as title, author and ISBN number are linked to a specific book type. Token features such as binding or edition are general features of any book, as they are independent of a book type. The better normalized the book information is, the easier it is to compare books. It is therefore important to normalize the information, in order to be able to create high quality information comparisons.

The clustering and normalization of linked features is done within the model. The type features are used as a basis to create Bookkeeper agents (Bookkeepers), which is why they aren't normalized until they are assessed within the model. The algorithm on how the author and title are normalized and clustered is given in Section 7.5.4. Clustering is done by measuring the Jaro-Winkler distance between an author or title and the exemplars of the currently existing cluster heads. The author or title is then classified based on a threshold, and a new cluster is created if no good match is found. As the Crawler agents (Crawlers) choose the books at random, the order in which they will find books is different every run. This results in a non-discrete model, where different exemplars can be chosen for clusters on run-throughs.

Unlike what one might expect, the book type features are not clustered by ISBN numbers. This is because an ISBN number is connected to a specific version or variation of a book, other than for reprints. Since the Bookkeepers are specific to an author, title and all its versions, using an ISBN number would not simplify clustering [57]. Furthermore, there are less unique authors than ISBN numbers or titles in the database, seen in Table 6.3. Finally, an ISBN number does not have any form of legal or copyright protection. This has resulted in tokens of titles being the same ISBN number.

Feature	Ranks
Book conditions	unknown, poor, fair, good, near fine, fine, very fine, as new, new
Dust jacket	unknown, none, missing, poor, fair, good, near fine, fine, very fine, new
Binding	unknown, paperback, softcover, leather, hardcover
Edition	other, one hundredth, seventy fifth, fiftieth, thirtieth, twenty fifth, twentieth, fifteenth, tenth, ninth, eighth, seventh, sixth, fifth, fourth, third, second, anniversary, limited, first

Table 6.2: The various options for each of the ranks.

6.3.3 Unlinked feature normalization

For token features, the normalization of information is less complex. Every token feature required a different approach to its normalization, which resulted in a parser being written for every token feature. However, there were some overlapping functions for every feature, such as the removal of irrelevant characters. Different parsers were required because the nature of the token features could be different, seeing as the variables for some features had to be normalized so that they could be ranked, while for others a less complex normalization was enough.

Token features such as edition, book or dust jacket condition required within-feature ranking, in order to depict the edition or condition of the book. The options for within-feature ranking can be seen in Table 6.2. The same algorithm as for clustering the authors has been used here, but with the difference that the cluster headers are pre-defined. The variables are cleaned once again and then based on a threshold matched to the best matching cluster. The threshold chosen is low, between 60 and 70%, in order to differentiate between an actual value or the unknown group, where variables are placed if they do not have a high enough overlap to the other ranks.

The categories, which are fiction and literature, are reduced from the plethora of categories by looking whether the words are present in the variables. Because the books have a lot of categories, the normalization helps the model its learning. As can be seen in Table 6.4, there is a great reduction in types of categories, but also a reduction in tokens.

6.4 Analytics

Over a period of ten weeks the crawlers obtained data. The absolute numbers of the books found per source category per week can be seen in Appendix A.2. It shows the number of books found per website and search category. Every week, an average of 1.65 million books was found, totaling to 16.54 million books. As mentioned before, one vendor in particular had a lot of unreliable and low quality books, so that vendor was omitted from the final model. Furthermore, there were still some duplicates found between crawlers, so these were also removed. This led to an end result of 6.44 million books that have been used for the model.

If all the books are then observed for type and token count of variables within a feature, the following image appears:

Table 6.3 contains the type and token count per feature, before processing. Important features are the title, author and vendor, because they are essential in the model for the clustering of books. The vendor is important for Ant Colony optimization.

Feature	Types	Tokens	Coverage (%)
price	177287	6409559	99.45
title	312063	6409470	99.45
publisher	98463	6257837	97.10
author	143754	6197589	96.16
sku	802008	6175359	95.82
vendor	5328	6175359	95.82
book_condition	17701	6140974	95.28
binding	1445	5976021	92.72
category	202118	5885595	91.32
vendor_location	10421	5191145	80.54
publication_date	20675	4636486	71.94
quantity	506	4627713	71.80
isbn13	239969	4321498	67.05
isbn10	239801	4319263	67.02
edition	11449	1340491	20.80
dustjacket	2	869200	13.49
dustjacket_condition	1270	647141	10.04
pages	3508	173420	2.69

Table 6.3: Raw variables types and tokens for the total database, which is 6444923 books. Coverage is the percentage of books in the database that has this feature.

A few select features are then normalized as part of the pre-processing, in order to reduce complexity for the model. Which features are normalized can be seen in Table 6.1.

Feature	Types	Tokens	Coverage (%)
price	26271	6409559	99.45
vendor	4383	6175359	95.82
category	2	4898885	76.01
publication_date	9963	4598059	71.34
isbn13	239958	4321414	67.05
isbn10	239800	4319258	67.02
organization	5	469632	7.29
signed by author	1	60171	0.93

Table 6.4: The features after they are normalized. Features which are not changed have not been shown.

As can be seen in Table 6.4, the number of variable tokens are reduced without a lot of cost to the number of tokens. It was furthermore possible to create two new categories, *organization* and *signed by author*, from the dirty data.

The largest difference is *category*, which has a coverage drop from 91.32 % to 76.01 %. This difference is due to the large number of inconsistencies that vendors made when writing the category section for their books. A mix of dates, names and even ISBN numbers are used in the category feature to describe the category to which the book belongs. While the search terms used were specific to *fiction* or *literature*, there are still results that do not contain any categories in their categories feature.

Lastly, it can be seen that the vendor count is lowered from 5328 to 4383, without lowering the coverage. This is due to local stores using different additions to their name, or because different

stores add an organization while stating that they are part of to their organization name. A high percentage of vendors contain few books, with a very small number of vendors that have a lot of books.

Chapter 7

Model

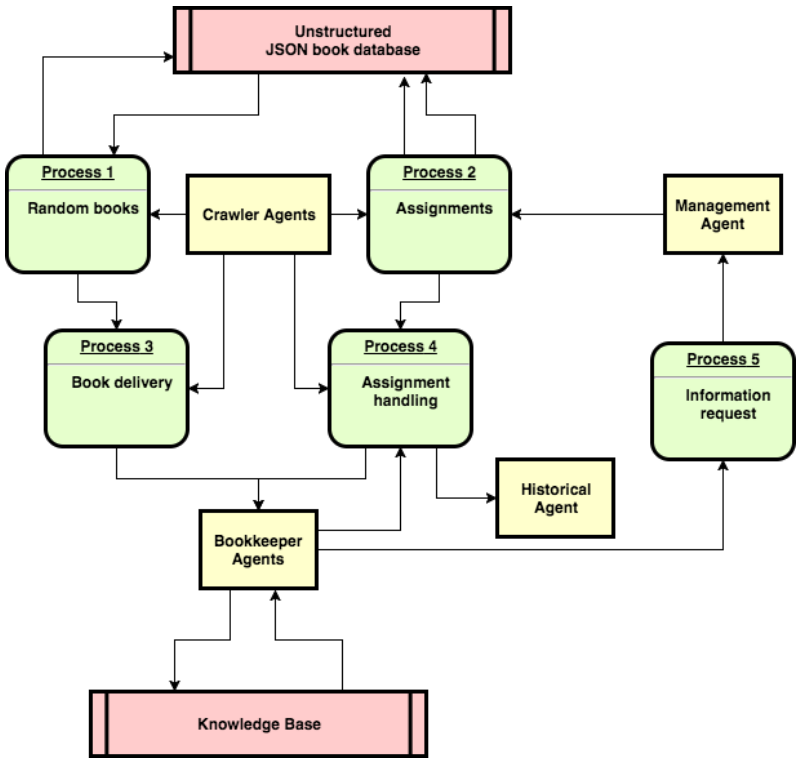


Figure 7.1: The model IDIMAS with assignment feedback.

7.1 Overview

The model contains various types of agents, which have specific different roles and tasks within the model. There is a group of agents known as Crawlers which retrieve books from the database. The database is unstructured and holds all the books per week. These books are retrieved and brought to relevant Bookkeepers, which store the books. Then these books are ranked per Bookkeeper agent (Bookkeeper), by calculating the book quality score (qualScore) and book information score (infoScore), which are compared to other books in their collection.

If Bookkeepers have books in their collection which they haven't had an update from in four weeks, then is it possible that they contact Management agent (Management), which takes assignments and adds them to a queue. This is dependent on whether assignments have been enabled. A Crawler agent (Crawler) can then take an assignment and search for that specific book at a vendor. The Crawler then returns to the relevant Bookkeeper with information about the book. If it does not exist, then the book is removed from the collection and Historical agent (Historical) is contacted. It stores the identifier and Bookkeeper as well as the epoch that the book is removed.

When new books are added to Bookkeepers, they receive two scores, a qualScore and an infoScore. For Ant System, which is explained in Section 7.4.2, if the new book is in the upper half of the ranking list, then the vendor that they're associated with will receive pheromones, increasing the likelihood of them returning. For Rank-based Ant System, which is defined in Section 7.4.3 , a more nuanced approach is taken, where the vendor associated with the book will receive pheromones dependent on the rank of the new book.

7.2 Agents

The model contains four unique agents, the Crawlers, Bookkeepers, Historical and Management. Each agent group has a unique role within the model.

7.2.1 Crawler

The Crawler is the link between the database and the rest of the model. A set number of Crawlers is created at the beginning of the model. The Crawler has four potential actions:

1. It communicates with the Management to check if there is an assignment available.
2. It attempts to retrieve a random book from the database. This book is then removed from the database and put into the local storage of the crawler. This can fail if the vendor no longer has books.
3. It attempts to retrieve a specific book as an assignment, in order to ascertain whether the book is still available.
4. It can deliver a book or the result of an assignment to a Bookkeeper.

It takes one epoch to communicate with Management and retrieve a book. To deliver a book or finish an assignment also takes one epoch.

7.2.2 Bookkeeper

The Bookkeeper stores the retrieved books based on title and author. It stores all the book versions for a single title and author. It also internally ranks the books by two ranking standards, which are qualScore and infoScore.

The number of Bookkeepers is dependent on the number of unique author-titles combinations found. They aren't created until a need for them has arisen. Since the number of unique author-title combinations in the database is unknown, it is therefore not known how many Bookkeepers are necessary.

The Bookkeeper has various actions, but is overall a generally reactive agent:

1. It can communicate with a Crawler if one reaches out to the Bookkeeper. This can either be to fulfill an assignment or deliver a book. If Bookkeeper receives a book, it is then added to the collection of retrieved books and also ranked.
2. If a book has been in the possession of the Bookkeeper for some time without update, it can request an update by sending an assignment to Management.
3. If a book is found to be no longer available, it is removed from the collection. This is done by contacting the Historical for the Bookkeeper and transferring the book to him.

The assignments have a grace period of four weeks before they can be issued. This means that a book has to be in a Bookkeeper's collection for four weeks without update, before assignment for an update can be requested. Since the information is presented on a weekly basis, the finalized time stamps of the books will be stored per week. A month is chosen, because it gives the model time to potentially find the book once again through the use of one of the algorithms described in Section 7.4, if it is still available.

7.2.3 Management

There is one Management agent, and its task is to keep a list of the available assignments. Bookkeepers send assignments to the Management, and Crawlers contact the Management agent in order to see if there are available assignments. The list of assignments is sorted on a last in, last out principle.

7.2.4 Historical

Originally the Historical kept the old records of books, connected to a Bookkeeper, within the database. However, as no analysis is done and it costs a large amount of space to store the old books, an alternative approach is chosen. Now the Historical now is contacted when a book is removed from the knowledge base, where Historical stored the Bookkeeper, book identifier and epoch. Therefore is it a reactive agent that only acts when it is contacted. It also does not have a specific role during an epoch, unless it is needed by a Bookkeeper.

7.3 Ranking

Two ranking algorithms have been implemented. The qualScore ranking is focused on a buyer that is looking for the best quality for price book that it can find for a certain title and author. The infoScore ranking is for buyers that are looking to buy a decently priced book from a reputable vendor.

7.3.1 Categories

Both formulas are based on four categories, namely feature presence, ranking values, Jaro-Winkler distance scores and the price quantity value.

Presence features	
quantity	organization
author	dustjacket_condition
signed_by_author	book_condition
binding	vendor
edition	location
title	sku
pages	price
publisher	publication_date
isbn10	isbn13
category	

Table 7.1: Features which are examined for the presence of a value.

Feature presence

The presence category depicts the presence or absence of variables for features. The features used are the features that a book has; these can be seen at Table 7.1. It does not look at the information contained within the feature, just if there is information present regarding that feature.

$$\alpha = \frac{pres}{pres_{len}} \tag{7.1}$$

For every feature that has a variable, a value of one is added, which ultimately is divided by the number of features which were checked, in order to normalize it. So α is *pres*, which is the number of features with values, divided by $pres_{len}$, which is the total number of presence features. The qualScore has an additional extra potential two points if the book is signed by an author.

Ranking values

Ranked features
book_condition
dustjacket_condition
binding
publication_date
edition

Table 7.2: Features which are ranked. The categories within the ranking can be seen in Table 6.2.

Rank values are a collection of token features which are categorized by rank, such as which edition the book is.

$$\beta = \frac{1}{n}\sum_{1}^{n}\frac{rank}{rank_{len}} \tag{7.2}$$

In Equation 7.2, β is therefore the category for the ranking values, where n is the number of features in the category, for this model *5*, as can be seen in Table 7.2. Then the *rank* divided by the $rank_{len}$ normalizes the value for that feature. The sum is taken and also divided by n, in order to normalize the category to make it comparable to other categories.

The higher quality a variable is for a feature, the more points it will receive. So a $book_{condition}$ variable *new* will receive a value *9*, which is then divided by *9*, as there are nine possible variables for that feature. There is no difference between qualScore and infoScore up until now.

The publication date is dependent on whether the qualScore or infoScore is being calculated. For the qualScore, the idea is that when a buyer is searching for a rare book, those books will overall be older. Therefore, old publication dates will receive higher values than newer, up to a point, where it hits a maximum. It is set to *50* years in this model. For the infoScore, it is the other way around. The more recent the publication date, the higher the score. Once again, there is also is a threshold set here, which is set at *50* years.

Lastly, one addition to the publication date is that it is year-based. The model checks whether a day or month is present. If everything is present, the maximum score will be 1. If the month or the day is missing, the score can reach a maximum of $\frac{2}{3}$. If both the day and month are missing, the end value can be a maximum of $\frac{1}{3}$.

Jaro-Winkler distance score

Jaro-Winkler features
book_condition
dustjacket_condition
binding
edition
title
author

Table 7.3: Features which are measured for Jaro-Winkler distances.

Jaro-Winkler distance scores are the category where the distance of variables to exemplars is calculated and used to deduce qualitative information about the book. For instance, if a variable of the *binding* feature is *hardbound*, it should belong to the rank *hardcover*, as seen in Table 6.2. However seeing as the words differ, so the character mutations needed to go from one word to the other are measured using the Jaro-Winkler distance. The closer the words are to each other, the higher the Jaro-Winkler score is, up to a maximum of one.

The features seen in Table 7.3 are used to measure the distance between the normalized variable and the exemplar or rank that it belongs to. The final value for the category is calculated by dividing all the Jaro-Winkler distances by the total number of Jaro-Winkler distances that are calculated.

$$\gamma = \frac{1}{n} \sum_{1}^{n} dist \tag{7.3}$$

The formula seen in Equation 7.3 functions the same as for Equation 7.2 in Section 7.3.1. For the features present in Table 7.3, the distance, written as *dist*, is calculated. This is done by calculating the Jaro-Winkler distance between the cleaned version of a new book and the exemplar for the relevant category. It is again normalized for the number of features.

Price quantity value

Finally there is the price quantity value, where the *price* and *quantity* are used to create a value indicator. It is based on the premise that the price and availability form a relationship on the rarity of the book.

The relationship between the two features is dependent on the type of ranking that is being calculated. The score has a floor and a ceiling, in order to ascertain that the category value will be a value in the range of zero to one.

7.3.2 Quality score

For the ranking of the quality of a book, the categories seen in Section 7.3.1 are used. These are feature presence α (Section 7.3.1), ranking β (Section 7.3.1), Jaro-Winkler distance γ (Section 7.3.1) and the price quantity value θ (Section 7.3.1). As was mentioned in Section 7.3.1, this category is dependent on the ranking algorithm used.

For the price quantity value, the following formula is used:

$$\theta = min(\frac{p}{q \times s_{max}}, 1.0) \tag{7.4}$$

For this formula, p is the price of the book, q is the quantity available at that particular vendor, and s_{max} is an arbitrary value, set to normalize the value up to an extent. For the model, s_{max} is set to *100*. The price of a book is divided by the quantity available, because this is a good indicator of a probable value of a book. If a book is cheap or there are many copies available, then the rarity of the book can be called into doubt.

The calculation of the qualScore is about the price versus quality ratio. It is meant for a buyer who is looking for a rare book which has a high price to quality ratio. Assumed is that for buyers in this category, there is a preference for rare books with a high quality.

There is also an emphasis on certain categories, in order to further differentiate the qualScore from the infoScore. A practical reasoning behind the differentiation in focus is also the fact that different aspects are important for the two scores. The qualScore is calculated using the following formula:

$$s_{qual} = c_0 \times \alpha + c_1 \times \beta + c_2 \times \gamma + c_3 \times \theta \tag{7.5}$$

The values for the list of c show the importance of the category it is connected with. For the model the following list was used as a base line: $c = \{0.2, 0.4, 0.2, 0.2\}$. There is an emphasis on the ranking category, since it contains information such as the binding, edition and publication date. These are deemed important features for a rare book.

A qualScore between 0.0 and 1.0 is calculated, which afterwards is compared to other books in the collection of a Bookkeeper. The ranking is then based on the qualScore, from the highest to the lowest score.

7.3.3 Information Score

The same categories are used as in Section 7.3.2, however with a different emphasis on the categories. There is also a different method used for calculating the price quantity value. The infoScore focuses more on the type of buyer who is trying to find a decent book for a decent price, and not trying to find a rare book.

The following formula is used for the category price quantity value, as described in Section 7.3.1:

$$\theta = min(\frac{p \times (q-1)}{s_{max}}, 1.0) \tag{7.6}$$

The assumption is that a vendor that has a large quantity of books available for a decent price, will be more reliable. Furthermore, it is assumed that such a vendor has a more equal list of books available. There is also one deducted from the quantity, in order to give single books with a high price don't receive a high score. s_{max} is set to *100*.

The same equation as Equation 7.5 is used, however, the c values are different. For the calculation of the information ranking, the following c values are used as the base line: $c = \{0.2, 0.2, 0.4, 0.2\}$. There is a greater emphasis on the correctness of the information provided in the book features, since the Jaro-Winkler distance has a higher rating. The underlying belief is that vendors that better advertise their books are more reliable and therefore appear to be a better buy for the average buyer.

7.3.4 Vendor feedback

At the end of every epoch, the new books added to the Bookkeepers are retrieved. Before this, all vendors lose a percentage of their current pheromones, depending on the evaporation rate. Dependent on the algorithm, pheromones are added at the vendors that have had new books added to the collection. For Ant System, if a book is in the top half of the collection, then it will receive pheromones. For Rank-based Ant System, it is dependent on its rank within the collection of a Bookkeeper.

There are two rank lists per Bookkeeper, one for the qualScore ranking, as well as another for the infoScore ranking. Only the new books that are not found through assignments are taken into account, and then the sum of the pheromones for both rankings is taken. There is no differentiation between different types of pheromones. So this means that if a book is placed second and first in the rankings, out of a total of 6 books, then for Ant System, it will receive twice the pheromone update amount values.

The higher the pheromone values are at a certain vendor, the higher the likelihood that a vendor will receive a return visit from a Crawler.

7.4 Algorithm

The algorithms implemented are Ant System, which is described in Section 7.4.2 and Rank-based Ant System, which is described in Section 7.4.3. The pheromone values are updated after each epoch.

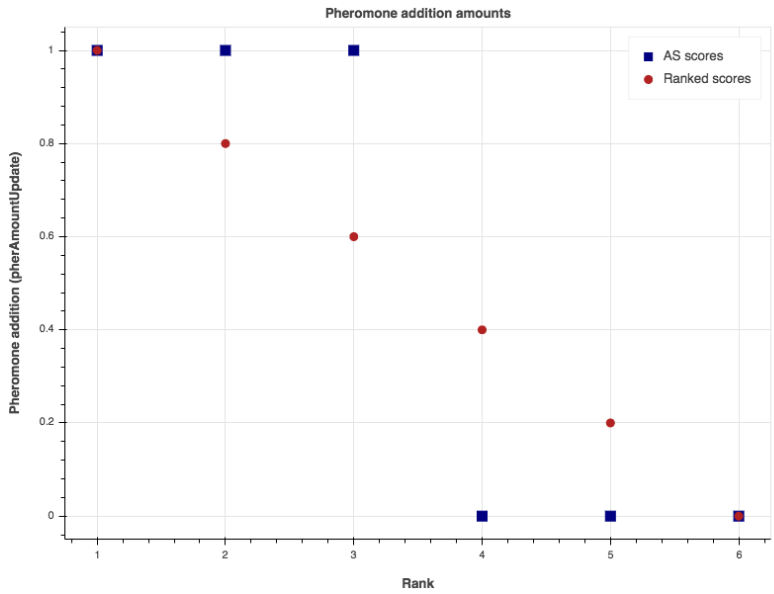


Figure 7.2: Figure showing the difference in pheromone update amounts for the different methods of Ant System and Rank-based Ant System for a collection of six books.

7.4.1 Random walk

For random walk, the model does not learn which vendors produce high-quality books. The Crawlers randomly choose a vendor and a book, and then deliver this book to the relevant Bookkeeper, if that exists. If it does not exist yet, then it is created.

7.4.2 Ant system

For Ant System, the model does learn. This is done based on the books in the collection of a Bookkeeper, as well as whether a Bookkeeper has received any new books.

$$\Delta\tau_{\eta+1} = (1-\rho)\Delta\tau_{\eta} + \sum_{a \in A}(qualScore + infoScore) \tag{7.7}$$

The formula associated with the pheromone update step is seen at Equation 7.7. Here, A is defined as every Bookkeeper that has a new book in their collection from a vendor η. At the end of every epoch, the model visits every Bookkeeper. If a Bookkeeper has received any new books, then the following is done:

If the Bookkeeper only has one book, then half the Pheromone amount update (Q) is added to the specific vendor. Q is a value which is pre-determined, the details of which for this model are discussed in Section 8.2.2. This can be the qualScore or infoScore rank list. The books are checked to see if they are new. If that is the case, then the vendor associated with that book will receive the pre-determined Q. If that is not the case, then nothing happens to that vendor.

If there is an odd number of books in the collection, the balance is tipped in favor of the higher quality half. Therefore for Ant System, in a collection of seven books, only the new books in the top four will be eligible for pheromones.

7.4.3 Rank based ant system

For Rank-based Ant System, a similar approach as in Section 7.4.2 is applied, except that a nuanced approach is chosen. The formula associated with Rank-based Ant System therefore is more complex. The same update rule is used as in Equation 7.7, but the qualScore and infoScore are no longer either the Q value or zero. This creates a more nuanced pheromone update.

$$score = Q - (\frac{1}{ranklist_{len} - 1} \times rank_{index}) \times Q \tag{7.8}$$

Almost all the new books in the collection of a Bookkeeper are taken into account. However, the rank of the book will determine the amount of pheromones that a vendor receives. The formula used to calculate qualScore and infoScore can be seen in Equation 7.8. Q is the pheromone amount update. It is once again normalized for the number of books in the collection. Important to note is that the index value is taken for $rank_{index}$, and not the actual rank. Therefore, if a book is ranked one in a collection, its index will be zero, as that is its place in the array. The formula creates a linear Q based on the rank of the book, up to a minimum of 0.0 points. For instance, if a bookkeeper has a collection of six books, the vendor connected to the first book will receive 1.0 point, while the second vendor will receive $\frac{4}{5}$ points, the third $\frac{3}{5}$ points, and so on. If a new book is added in last place, no points are received.

7.5 Epoch

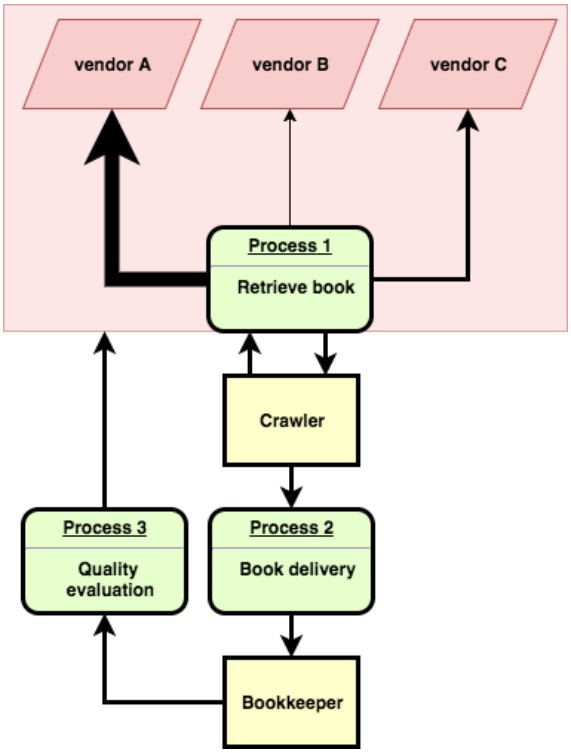


Figure 7.3: A simplified example of an epoch in the model IDIMAS. The thickness of the arrow within the red box depicts the quality of the vendor. The thicker the arrow is, the higher quality the vendor is.

An epoch with assignment feedback goes as follows; First the model checks whether a significant epoch has been reached. If that is the case, then the currentDate is updated to a new value and the information corresponding to that date is loaded into the model. The old information is removed.

Afterwards, a check is done to see if the database still has books and also how many epochs it has been since the model last had books. If it has been without books for too long, then the model is stopped in order to decrease unnecessary runtime.

Then the Crawlers each get a turn. Every Crawler does the following: first it checks if it is busy, because then it has had contact with the database and is heading back towards the Bookkeepers. If that is the case, then it is checked whether the Crawler also has an assignment. If the Crawler has an assignment, then it moves to finish its assignment. If it did not have an assignment, then it will take the book in its possession and find the Bookkeeper relevant to that book. If it does not exist yet, then a new Bookkeeper is called into life. The Crawler delivers the book or finishes its assignment.

If the Crawler is not busy, then it also has two options. First it communicates with the Management in order to see if there are assignments waiting to be done. If that is the case, the first assignment from the list of assignments from Management is taken. The Crawler then crawls through the database to try and find a relevant book for the assignment. If Management doesn't have an assignment, a random vendor is chosen and a random book is taken.

Once all the Crawlers have done an action, the Bookkeepers are run through. They have one possible action, which is to run through their books and check them for freshness. They do this by running through each of their books and looking at when the book was last delivered to them. If it is a book that was found more than a month ago, then they create an assignment requesting an update on that book. These assignments get delivered to the Management, which adds them to the end of the queue. This queue is sorted in order of appearance of the assignments.

7.5.1 crFetchRandomBook

The basic function is crFetchRandomBook, it corresponds with process 1 in Figure 7.3. If there is no assignment and the Crawler currently doesn't have an assignment, then the Crawler goes to a random vendor and takes a random book available. There is a possibility where the vendor is empty, as the empty vendors only get removed at the end of the epoch. Then the Crawler leaves the vendor empty-handed and retries a different vendor during the next epoch.

7.5.2 crFetchAssignment

If the Crawler has an assignment, then that means that the crawler is heading to the vendor specified in the assignment, in order to attempt to find a copy of the book. The result of this, is that the Crawler either has the newer version of the book in its possession, or it has nothing. With newer version is meant that the vendor still has the book from the previous week. This also corresponds with process 1 in Figure 7.3, but with the difference that the strength of the path is irrelevant, and the actual vendor is relevant.

7.5.3 crFinishAssignment

With the given assignment, the Crawler contacts the Bookkeeper which is mentioned in the assignment, and either hands him a book or nothing at all. The model is built to be able to handle

multiple types of assignments, yet currently only one is possible. This is done in process 2 in Figure 7.3.

This assignment is to find a specific book; if the Crawler returns with a book, then this means that the book is still present at the store and therefore the Bookkeeper can keep the book in its collection for a little while longer. The date corresponding to the book the Crawler brought is added to the version in possession of the Bookkeeper, to get an idea of how long the book has been available. If the Crawler returns without a book, then it means the book is no longer available, meaning that it either has been sold or the Crawlers weren't able to find the book. It is removed from the Bookkeeper's database.

7.5.4 crDeliverRandomBook

Algorithm 2 Bookkeeper Matcher

```
procedure BOOKKEEPER FINDER
    Set authorMatch empty
    for bk : bookkeepers do
        authorMatchScore equals bookAuthor overlap bk[author]
        if authorMatchScore is over 95% then
            Set authorMatch to bk[author]
            titleMatchScore equals bookTitle overlap bk[title]
            if titleMatchScore is over 80 % equal then
                Add book to bookkeeper
    if authorMatch is found then
        Create new bookkeeper with authorMatch and bookTitle
    else
        Create new bookkeeper with bookAuthor and bookTitle
```

The Crawler has taken a random book from a random vendor. It attempts to find the relevant Bookkeeper for that book, first based on the author. If an author match has been found, then it looks for a match in the title. If it finds a match, then the book is delivered to that Bookkeeper. If a match is not found, then a new Bookkeeper is created, which receives that book. This process can be seen in the Algorithm 2 and falls under process 2 for Figure 7.3.

Finally the process described in Section 7.3.4 is done, depicted by process 3 in Figure 7.3. The vendors associated with the high-quality books that have just been added to the knowledge base retrieve additional pheromones. Figure 7.3 identifies these high-quality vendors through the thickness of the arrow leading to them.

Chapter 8

Experiments

8.1 Result comparison

The algorithms and parameters will be measured through various groups. These groups are knowledge, books and vendor rankings.

8.1.1 Knowledge

An important result is knowledge; this means the number of Bookkeepers, what their average number of books is, but also the number of Bookkeepers that are currently without books, as well as the number of Bookkeepers that have one book.

The numbers of Bookkeepers with zero, one, or more books, show how much effect the rankings have. They also giving insight into the number of types of books present in the knowledge base, while taking the sum of books in the knowledge base shows how many books are present in the knowledge base at a time.

8.1.2 Book information

The book information is measured in two ways: both the average freshness of books in bookkeeper collection, as well as the fluctuation in books for a collection. These values give information on how up to date a Bookkeeper is, which can be extended to see how up to date the knowledge base is. This is measured at the end of the model run, after 100 epochs.

If the number of Crawler is low and the book fluctuations for the methods are also low, then that means that overall books are being found that already are present in the knowledge base. This also happens if books are being removed.

The average freshness of books is measured by taking the age of the last checked values for the books at a Bookkeeper, at the end of the model. Book fluctuation is measured by the difference in books for vendors between epochs. Since the assignment for book feedback is set at a month, it is expected that the Ant Colony optimization algorithms will have a lower freshness, so a lower average book age than Random Walk (RW) (Random Walk).

The number of duplicates is the the number of times that a unique book within a Bookkeeper is found. If a book is found in weeks 2, 3 and 8, then it will have three duplicates and also a freshness of three.

8.1.3 Vendor Rankings

The vendor rankings are also measured, by looking at the pheromone values. A high pheromone value indicates a more valued vendor, since it is being visited more often than the other vendors. Therefore is it expected to see differences in the top ten vendors across the various algorithms and parameters. These differences can be in which vendors achieve higher rankings, but also in the fluctuations in the rankings.

As mentioned in Sections 8.2.2 and 8.2.3, it is expected that varying these parameters will have a large influence on the rankings. They are likely to influence both the vendors which appear high on the ranking list, as well as the longevity of those vendors. Varying the coverage percentage, which is described in Section 8.2.1, is also expected to have influence on the rankings. Lower coverages are presumed to be more prone to finding local optima, because the first vendors are found randomly and will have a larger influence. Since the creation of new Bookkeepers will give positive feedback to the relevant vendor, it will increase the chance that those vendors receive a return visit.

The number of vendors that still have books in their databases is also measured, along with the number of vendors that are sold out in an epoch. The expectation is that there are clear differences between Random Walk and the Ant Colony optimization algorithms, since the Ant Colony optimization algorithms are more likely to focus on a select few vendors, contrary to Random Walk.

8.2 Parameters

Various parameters are varied in the experiments in order to test the influence of a specific parameter setting. For every parameter setting, the model is run ten times in order to be able to retrieve significant average results. Furthermore, the categories for qualScore and infoScore are already calculated when loading in a weekly database if possible, in order to reduce the size of the database in memory.

8.2.1 Crawler agents

Coverage (%)	# of agents
1	1289
5	6446
10	12890
25	32225
50	64445
75	96674
100	128899

Table 8.1: This table shows the theoretical number of agents required for an average coverage of the database. The numbers are rounded up since number of agents are absolute.

Various book-finding coverage percentages are tested; these can be seen in Table 8.1. Book-finding coverage is defined as the percentage of books that Crawlers theoretically can retrieve from the database before it is renewed. There are an average of 644492,3 books found per week. The number of crawler agents required for a certain coverage can be calculated as follows:

$$cr_{num} = cvr * \frac{db_{books}}{10 * 5} \tag{8.1}$$

In Equation 8.1, cr_{num} is the number of agents required, where cvr is the coverage percentage one wishes to achieve, and finally, db_{books} is the total number of books in the database. The *10* is the number of weeks that the model runs, while the *5* is the number of times the Crawlers can successfully retrieve books from the database in that week. Since a week lasts ten epochs, of which a maximum of five can be used to successfully retrieve information from the database, that means that 128898,5 Crawlers are needed for complete coverage. An overview of the agents required can be seen in Table 8.1.

Hypothesis 1 *An increase in the number of crawlers is expected to decrease the differences between the algorithms of Ant System, Rank-based Ant System and Random Walk.*

The expectations for the variation of the number of crawler agents is defined in the Hypothesis 1.

8.2.2 Pheromone amount update

With the pheromone update parameter, it is set how much positive feedback a vendor will receive. This defines how quickly a vendor will receive pheromones and therefore is likely to rise in the rankings. Dependent on the algorithm, there are several degrees for pheromone update values within a book collection. The formulas and algorithms used can be see in Section 7.4. The pheromone updates vary between 1.00 to 2.50, with increments of 0.75. A large value of 10.00 is also included.

Dorigo found that for the problem of shortest-path finding, the value of the pheromone amount update was of negligible influence [31]. As the number of interactions is very high in IDIMAS, an influence could be more visible for Ant System and Rank-based Ant System.

Hypothesis 2 *Varying the number of pheromones that is deposited by a single agent at a vendor is of negligible influence.*

Hypothesis 2 states that varying the number of pheromones that is deposited by an agent is of negligible influence to the results of the model.

8.2.3 Evaporation rate

The evaporation rate influences the rate at which vendors lose pheromones. It is expected to influence the longevity of positive feedback for vendors. The evaporation rate value is varied between 0.1 to 0.7, with increments of 0.15.

Hypothesis 3 *Increasing the evaporation rate is expected to increase the fluctuations of ranks of vendors in the vendor top ten, since it affects the longevity of number of pheromones that a vendor has.*

It is expected that a higher evaporation rate will lower the longevity of high-ranking vendors, as stated in Hypothesis 3. When high-ranking vendors run out of books, they will no longer receive

pheromones but keep losing them, contrary to their lower-ranked competitors. Also because they have a higher ranking, relatively more Crawlers will retrieve more books, meaning that they run out faster in a week than their competitors. If the evaporation rate is lower, then it is expected that there is less fluctuation in the top five ranking for the vendors, since their pheromones will decrease at a lower rate.

8.2.4 Assignments

As is explained in Section 7.2.2, Bookkeepers have the option of creating assignments, which they can send to a Management, in order to direct a Crawler. A Bookkeeper sends a Crawler directly towards a specific vendor with the task of finding a specific book. The success or failure of this assignment is measured by the presence or absence of the book. If no book is found, then it is no longer in the database.

Hypothesis 4 *The addition of assignments is expected to decrease the freshness of the books in the knowledge base.*

Hypothesis 5 *The addition of assignments is expected to increase the number of duplicates in the knowledge base.*

The grace period for assignments is four weeks; the reasoning behind this is discussed in Section 7.2.2. The addition of assignments will reduce the number of books in the Bookkeepers, because there is no other way of removing books in the knowledge base. Furthermore, it is expected that assignments will reduce freshness of books in the model, defined by Hypothesis 4, because books will periodically get tested for presence. This is contrary to a model without assignments, where books can only get an update they are is found by chance in the following weeks. The likelihood of an Ant System or Rank-based Ant System finding newer versions of the current books in the database is expected to be higher, since the Crawlers are more likely to return to high-ranking vendors, which presumably are the vendors from which the earlier versions of the books came. Hypothesis 5 covers this, by stating that it expects the number of duplicates to increase.

These directed searches will influence the results in various ways, because they actively promote Crawlers to go back to earlier visited vendors to see if that vendor still has a specific book in store. This results in that vendor not receiving any new pheromones, since the books are retrieved through a directed search and not by chance. Secondly, this also means that the vendor will have less books remaining in its database, so that it will run out faster.

Hypothesis 6 *The addition of assignments is expected to decrease the book count of the knowledge base.*

Furthermore, dependent on the coverage of Crawlers, assignment feedback is expected to create a smaller, but more up to date knowledge base. This is stated in Hypothesis 6. Assignments are expected to limit the ability of Crawlers to find new books.

8.2.5 Ranking importance categories

The ranking algorithms qualScore and infoScore are clustered into four categories in order to reduce complexity and improve clarity. The four categories are:

- Feature presence

- Ranking values
- Jaro-Winkler distances score
- Price quantity value

By stressing different aspects of the book information, the rankings will most likely differ. Therefore the positive aspects of different vendors can come forward. The qualScore and infoScore have different weights attached to some of their categories, including a different price quantity calculation. A description of the qualScore can be seen in Section 7.3.2, and for infoScore it can be seen in Section 7.3.3, however, they both use the same Equation 7.5.

Hypothesis 7 *Using only one of the two score measures, qualScore or infoScore, will result in a different top ten vendor ranking. These results are expected to appear when comparing them to the general settings as well as between the two scores.*

Hypothesis 8 *Setting the weights for all the categories to the same value will have a small influence on the top ten vendor ranking, when compared to the general settings top ten vendor ranking.*

Hypothesis 9 *Changing the value of the weights is expected to have an impact on the height of the scores, but not the development of the scores.*

Various setups are tested to see the influence of the different category importances on both the qualScore and infoScore ratings as well as the vendor rankings. The most influence is expected to be seen when just one of the score methods is used, as depicted by Hypothesis 7. Different setups are expected to show consistent different vendors at the top of the rankings, as well as different average ratings. Setting all the weights for the categories to an equal number is expected to have small influence on the top ten vendor ranking, as compared to the standard setting, this is shown in Hypothesis 8. Overall is it expected that changing the weights will have an influence on the height of the scores, which is shown in Hypothesis 9. The setups tested are allEqual, where the categories weights are set to $c = \{0.25, 0.25, 0.25, 0.25\}$, as well as only qualScore and only infoScore.

8.3 Random walk

The model in this implementation Random Walk does not have a learning algorithm. There will therefore never be a preference for Crawlers to retrieve books from a specific vendor, but Crawlers will instead choose a random vendor when searching for books.

Random Walk is the base test case in order to compare the impact of an Ant Colony optimization algorithm on the effectiveness of grouping and ranking books. As there is no learning, there won't be a pheromone update, nor an evaporation rate. The main influence will be the coverage of Crawlers, where it is expected that more Crawlers will mean a more knowledgeable model. A coverage of 100% is expected to yield the same results as using an Ant Colony optimization, because almost all the books will be found every week.

Hypothesis 10 *The random walk algorithm is expected to have a higher number of Bookkeepers with a higher book count in the knowledge base than Ant System and Rank-based Ant System. Furthermore is it expected to have a lower number of duplicates and a higher freshness.*

Overall, Random Walk is expected to have a larger group of Bookkeepers with more books, but with less up-to-date knowledge base, so a higher freshness. Since the focus for the Crawlers is not

on any vendor, it is more likely that they randomly find new vendors, which might have their own unique books. This can be seen summarized in Hypothesis 10. Vendor rankings do not play a role in this algorithm, since the Crawlers have no preference for a certain vendor at any time.

8.4 Ant system

The Ant System algorithm is used in this implementation as another benchmark, since the database itself is newly created.

Hypothesis 11 *Ant System is expected to have a focus on the top couple of vendors in the vendor ranking top ten.*

The Ant System algorithm as a basis focuses on finding the optimal vendor, so it is expected that there will be a single high-ranking vendor, shown through Hypothesis 11. A larger amount of the coverage of Crawlers is expected to increase the number of books in the knowledge base of the model, while keeping the age of the books the same. Furthermore, it is likely that a higher evaporation rate will influence fluctuations in the vendor rankings, because the higher ranking vendors will generally be focused on first, meaning that they will be out of books the fastest, and thus have to wait the largest number of epochs until a new batch of books arrives the next week. They will lose pheromones in the meanwhile, like all the vendors, but since they have a high amount, they will lose relatively the most number of pheromones, while the other vendors can still increase their ranking if they still have books in their database that are being retrieved.

Hypothesis 12 *When comparing Ant System to Random Walk, the expectation is that Ant System will have a smaller knowledge base, but with a lower freshness and higher number of duplicates.*

Compared to Random Walk and Rank-based Ant System, Ant System is generally expected to have a smaller knowledge base, but with a lower freshness, as is depicted in Hypothesis 12. Since Crawlers are more likely to go back to previously visited and preferred vendors, the chances are that they'll most likely find books already in the knowledge base, but from an older week. This prevents them from finding new books, as time is a limiting factor, since the books are only available for ten epochs. For the vendor rankings, it is expected that one or a few vendors will have very high pheromone values compared to the other vendors, due to the Ant Colony optimization algorithm having a strong preference for finding a single best vendor.

Another reason for the expectations of a smaller knowledge base, is due to Crawlers having a preference for returning to previous vendors, even if they have run out of books. Updating the knowledge on which paths are viable is a daemon action, which is performed at the end of every epoch. It is therefore possible that one Crawler takes the last book, and that all the Crawlers after that go to the same vendor, but return empty-handed, as there are no books to take. A higher coverage is thus expected to give a relatively lower increase in the knowledge base, since the spread of the Crawlers for Ant System and Rank-based Ant System will be lower than for random walk.

8.5 Rank-based ant system

Hypothesis 13 *Rank-Based Ant System is expected to have the average ranks of the top vendors be closer to each other, creating a stepwise increase with several vendors at each step in the vendor*

ranking top ten.

The Rank-based Ant System is tested in this experiment, where the rank is the place of the book in the collection. The higher it is placed within the collection of a Bookkeeper, the stronger the pheromone update will be for that vendor. The implementation is discussed in Section 7.4.3. Contrary to the original Ant System, there is less of a focus on finding the absolute best vendor, leaving more room for alternative vendors to also obtain high pheromone values. Therefore Hypothesis 13 states that the top ten vendor rankings are expected to be stepwise, since vendors are more likely to be grouped.

The evaporation rate is expected to influence the longevity of high-ranking vendors, that is already depicted through Hypothesis 3. When the evaporation rate is high, a larger fluctuation in the highest-ranking vendors is expected, since the Crawlers will have a high preference to go to these vendors first, and therefore quickly retrieving the books of a vendor. This means that for the later part of a week, their pheromone value will drop due to the evaporation rate and the fact that their score will no longer increase, since they can't supply new books.

Hypothesis 14 *When comparing Rank-Based Ant System to Random Walk, the expectation is that Rank-Based Ant System will have a smaller knowledge base, but with a lower freshness and higher number of duplicates.*

Rank-based Ant System is also expected to have a smaller database than Random Walk, since the assignments keep the Crawlers focused on vendors that have already been visited before, because they have a higher likelihood of containing books which are already in the database. Therefore, the Hypothesis 14 has a lot of overlap with Hypothesis 12. Since this variant is rank based, a similar effect to Ant System is likely to be observed, however more nuanced with the top ranking vendors being closer together, since other vendors that also produce high quality books will also get positive feedback.

Hypothesis 15 *When comparing Rank-Based Ant System to Ant System, the expectation is that Ant System will have a smaller knowledge base, as well as a lower freshness and higher number of duplicates.*

When comparing between the algorithms of Ant System and Rank-based Ant System, it is expected that Ant System will have a smaller knowledge base with a lower freshness. This is due to the preference for a select few vendors, with the expectation that the vendors will be sold out faster for Ant System, since relatively more Crawlers are likely to visit that vendor. This is hypothesized in Hypothesis 15. More Crawlers visiting an empty vendor means that the efficiency of the crawlers is decreased, resulting in a slower increase in the knowledge base, since fewer books are being retrieved.

Chapter 9

Results

The results for the model is discussed in this chapter. Since there were some unexpected results, new settings for the coverage parameter were tested in order to attempt to explain these results. All of the results were obtained by running the model ten times per parameter and averaging the results, in order to obtain significant results, because the model is not discrete.

9.1 General settings

This section shows the comparison between the various algorithms that have been used. The settings that are used are a *pheromone amount update* of 1.0, *evaporation rate* is 0.25, *coverage* is 0.25, no assignments available, the weights of the categories for infoScore are $0.2, 0.4, 0.2, 0.2$, and the weights for the categories of qualScore are $0.2, 0.2, 0.4, 0.2$. The categories are described in Section 7.3.1.

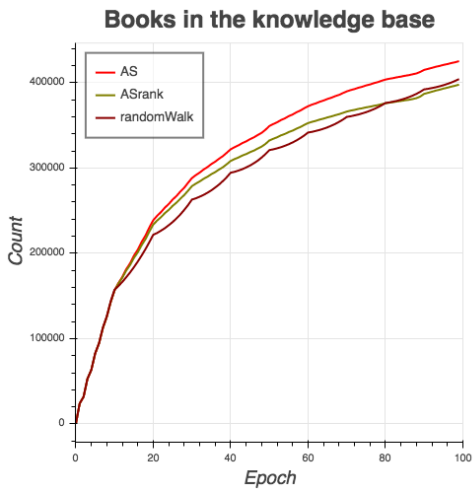


Figure 9.1: The total number of unique books in the knowledge base on average per epoch.

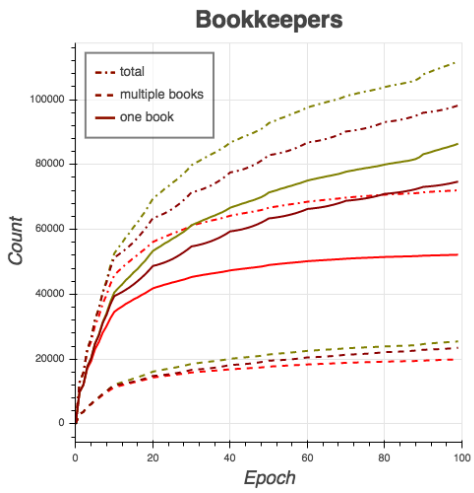


Figure 9.2: The number of bookkeepers in the model for various algorithms. The colours in the graph are the same as for the legend in Figure 9.1.

Figure 9.1 depicts the total number of books in the knowledge base at the end of an epoch. Figure 9.2 shows the number of Bookkeepers in the model at the end of an epoch. It also shows the ratio

of what the Bookkeepers are made up of, meaning how many Bookkeepers have one book and how many have more than one book in their collection. The colours in both graphs depict the same algorithms.

	Count			Knowledge base	
Algorithm	Book	Bookkeeper	Ratio	Duplicates	Freshness
randomWalk	404190 (σ=603)	98186 (σ=108)	4.12	3.83 (σ=2.97)	2.33 (σ=2.59)
Ant System	425207 (σ=3681)	72045 (σ=476)	5.90	3.82 (σ=2.54)	2.52 (σ=2.85)
Rank-based Ant System	397729 (σ=4942)	111856 (σ=2974)	3.56	4.02 (σ=2.81)	2.69 (σ=3.06)

Table 9.1: An overview for the mean and standard deviations of various variables, measured at the end of the final epoch of the model.

Table 9.1 shows the average end state of a model for the various algorithms. The columns under *Count* show the size of the knowledge base for the three algorithm, as well as the total number of Bookkeepers. There are two column headers under *Knowledge base*, which are *Duplicates* and *Freshness*. The column *Duplicates* depicts the average number of times that a book has appeared in the knowledge base, and *Freshness* shows the average distance in weeks between the end state and the last week that the book was found. Books in the knowledge base that are only found in the first week are excluded from the measurement of the *Freshness* and *Duplicates*.

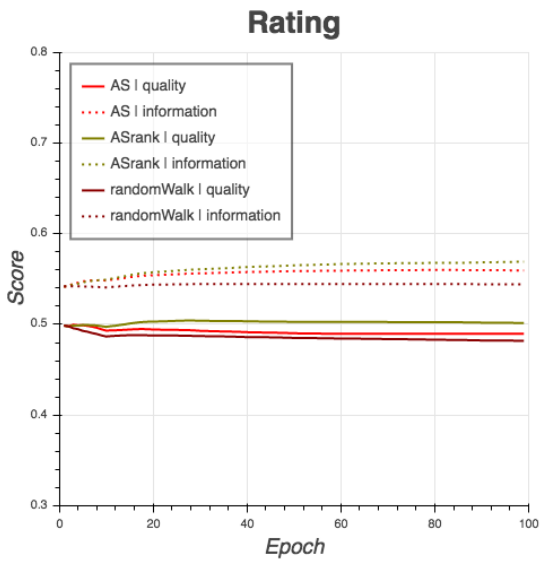


Figure 9.3: The average information and quality rating per epoch.

	Rating	
Algorithm	Quality	Information
randomWalk	0.48 (σ=0.1)	0.54 (σ=0.08)
Ant System	0.49 (σ=0.1)	0.56 (σ=0.08)
Rank-based Ant System	0.50 (σ=0.1)	0.57 (σ=0.09)

Table 9.2: The quality and information rating score averages and standard deviations for the various algorithms.

Table 9.2 contains the qualScore and infoScore ratings for the various algorithms. The development of the ratings over time can be seen in Figure 9.3.

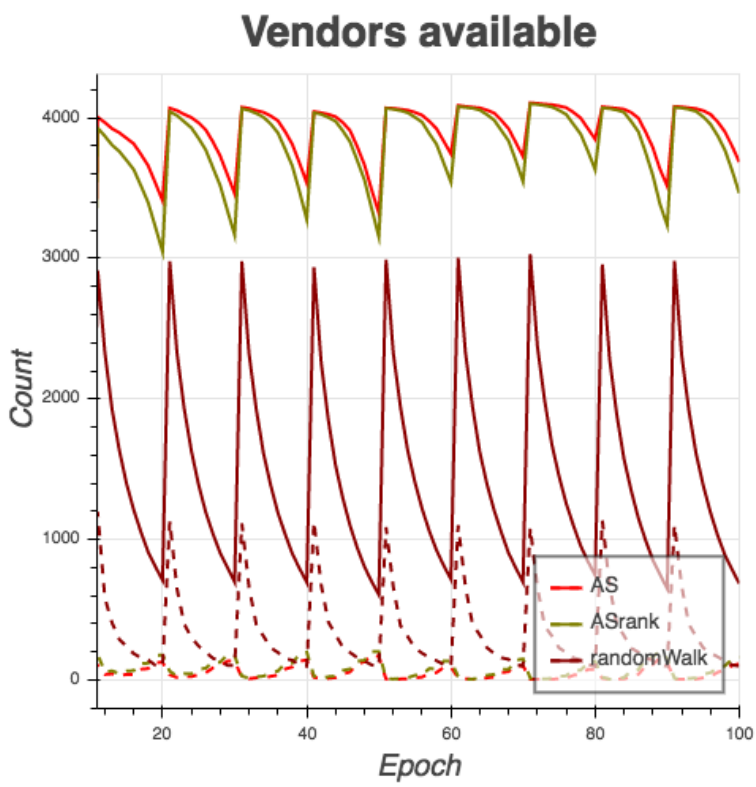


Figure 9.4: The number of vendors with books available as well as the number of vendors removed at the end of every epoch. The dotted line represents the number of vendors that have been removed.

Figure 9.4 show the number of vendors available as well as the number of vendors removed at the end of an epoch. A vendor is removed from the model if it no longer has any books. Random Walk is shown to have a lower number of vendors available than the Ant Colony optimization algorithms.

The Figures 9.5 and 9.6 show the top ten vendor rankings and their fluctuations. Figure 9.5 shows the average number of mutations between epochs. A mutation occurs if a vendor no longer holds the same rank as in the previous epoch. Figure 9.6 is more complex. For every epoch, the top ten vendors receive the inverse of their rank as points. Therefore, the best vendor, meaning that it has the highest number of pheromones, will receive ten points, the number two vendor nine, and so on. The sum of these points, divided by 90, the number of measured epochs, gives an average vendor ranking during a model run. If a vendor has ten points, then it means that this vendor was number one for all the epochs. These numbers are calculated and averaged over ten runs, after which a top ten ranking is made, based on the then highest number of points. If a vendor was high ranking in just one model run, then it is unlikely that it appears in the final top ten. The mean and standard deviation are shown, in order to show the fluctuations of the ranks for that vendor.

This method of display is chosen because it shows a normalized ranking of the vendors. Using the actual pheromone values would create discrepancies, because the total amount of pheromones available in the model is different depending on the epoch.

There is 60% overlap between the two vendor top ten lists for Ant System and Rank-based Ant System, so six out of ten vendors appear in both lists. The mean rank difference is 0.67, with a

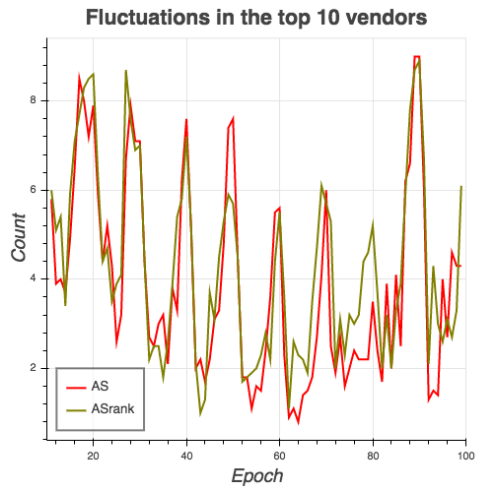


Figure 9.5: The fluctuations in the vendor ranking top ten.

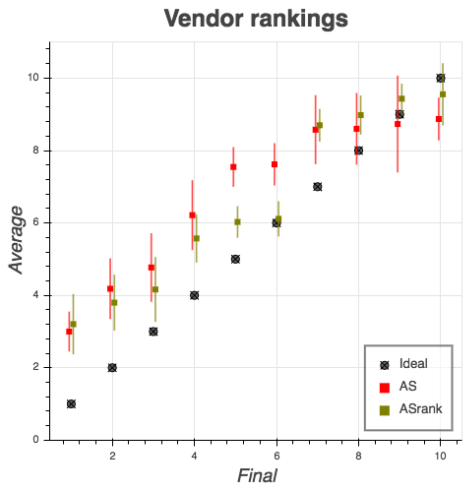


Figure 9.6: The ideal rank and actual average rank. How this score has been calculated can be read it the text below.

standard deviation of 1.11. Therefore, the overlapping vendors are quite close together.

9.2 Evaporation Rate

The evaporation rate uses the same general settings as described in Section 9.1, these are the *pheromone amount update* set at 1.0, *evaporationrate* is 0.25, *coverage* is 0.25, no assignments available, the weights for the categories of qualScore are $0.2, 0.4, 0.2, 0.2$ and the weights for the categories of infoScore are $0.2, 0.2, 0.4, 0.2$, but with the difference that the *evaporationRate* is varied between 0.1 and 0.7, with increments of 0.15.

9.2.1 Ant System

The shown graphs and tables are in the same lay-out as those in Section 9.1. The evaporation rates are tested on the Ant System algorithm in this section.

What can be seen, is that there is more overlap between vendors in the top ten rankings when the evaporation rates are close together. The further the evaporation rates are apart, the lower the overlap is.

Table 9.3 shows that while the book count increases for a higher evaporation rate, the number of Bookkeepers stays the same. The number of duplicates decreases while the the freshness increases.

9.2.2 Rank-based Ant System

The graphs and tables shown below are in the same lay-out as those in Section 9.1. The evaporation rates are tested on the Rank-based Ant System algorithm in this section.

The same effect as for the Ant System is visible here for the overlap between vendors in the top ten. If the distance between the evaporation rates is larger, then there is less overlap between the

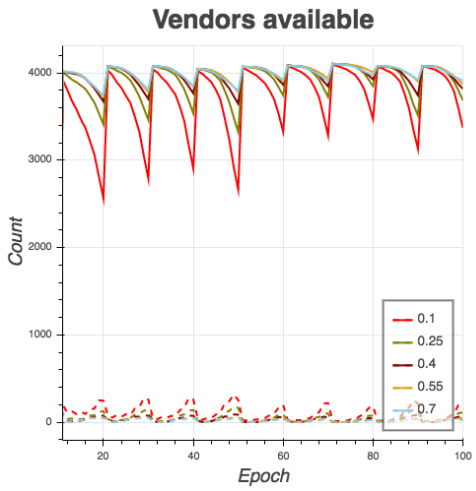


Figure 9.7: The number of vendors with books available at the end of every epoch with Ant System. The dotted line shows the number of vendors that have been removed.

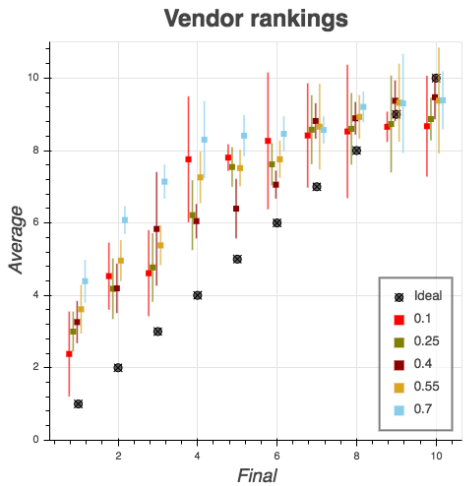


Figure 9.8: The ideal rank and actual average rank with Ant System.

	Count			Knowledge base	
Evaporation rate	Book	Bookkeeper	Ratio	Duplicates	Freshness
0.1	407478 (σ=2366)	75634 (σ=483)	5.39	3.95 (σ=2.67)	2.32 (σ=2.66)
0.25	425207 (σ=3681)	72045 (σ=476)	5.90	3.82 (σ=2.54)	2.52 (σ=2.85)
0.4	440631 (σ=6101)	70668 (σ=643)	6.23	3.70 (σ=2.36)	2.61 (σ=2.93)
0.55	459198 (σ=4116)	70884 (σ=527)	6.48	3.54 (σ=2.06)	2.79 (σ=3.02)
0.70	512685 (σ=15364)	74126 (σ=1188)	6.92	3.11 (σ=1.83)	3.00 (σ=3.06)

Table 9.3: An overview for the mean and standard deviations of various variables, measured at the end of the final epoch of the model while using Ant System.

vendors in the top ten. This means that there is more overlap between vendors when comparing evaporation rates 0.25 and 0.40, than there is when comparing 0.10 and 0.70.

	Count			Knowledge base	
Evaporation rate	Book	Bookkeeper	Ratio	Duplicates	Freshness
0.1	386294 (σ=2138)	103010 (σ=763)	3.75	4.12 (σ=2.90)	2.48 (σ=2.86)
0.25	397729 (σ=4942)	111856 (σ=2974)	3.56	4.02 (σ=2.81)	2.69 (σ=3.06)
0.4	433284 (σ=4238)	136905 (σ=2240)	3.16	3.70 (σ=2.62)	2.75 (σ=3.06)
0.55	523014 (σ=6105)	180073 (σ=5429)	2.90	3.06 (σ=2.08)	2.87 (σ=2.95)
0.70	557775 (σ=15951)	194765 (σ=2015)	2.86	2.89 (σ=1.65)	3.54 (σ=3.14)

Table 9.4: An overview for the mean and standard deviations of various variables, measured at the end of the final epoch of the model using Rank-based Ant System.

Tabel 9.4 shows that both the number of unique books tokens and the number of unique book types increase when the evaporation rate increases. The number of duplicates decreases while the

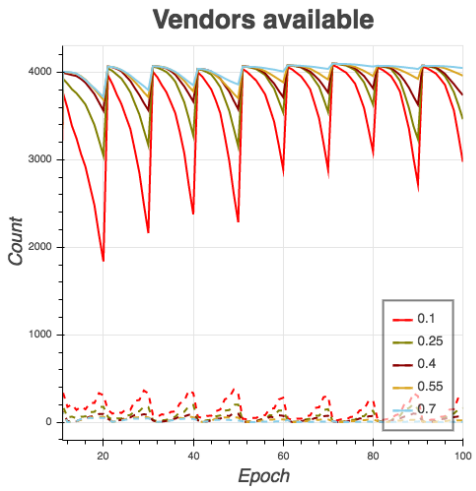


Figure 9.9: The number of vendors with books available at the end of every epoch with Rank-based Ant System.

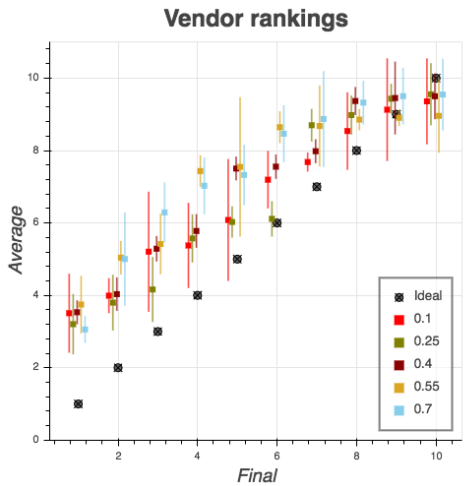


Figure 9.10: The ideal rank and actual average rank with Rank-based Ant System.

freshness increases.

9.3 Pheromone amount update

The settings that are used are a varying number for *pheromone amount update*, the *evaporationrate* is 0.25, the *coverage* is 0.25, there are no assignments available, the weights for the categories of qualScore are $0.2, 0.4, 0.2, 0.2$ and the weights for the categories of infoScore are $0.2, 0.2, 0.4, 0.2$, as described in Section 9.1. Because the differences between the algorithms were small, only Rank-based Ant System is shown here. The results for Ant System can be seen in Appendix A.1.

Pheromone amount update	Count			Knowledge Base	
	Book	Bookkeeper	Ratio	Duplicates	Freshness
1.00	397729 (σ=4942)	111856 (σ=2974)	3.56	4.02 (σ=2.81)	2.69 (σ=3.06)
1.75	392840 (σ=4300)	110611 (σ=2051)	3.55	4.07 (σ=2.81)	2.65 (σ=3.05)
2.50	396563 (σ=4726)	112181 (σ=2161)	3.54	4.03 (σ=2.80)	2.67 (σ=3.05)
10.0	394068 (σ=5184)	113121 (σ=3949)	3.48	4.06 (σ=2.82)	2.64 (σ=3.04)

Table 9.5: The means and standard deviations for the various pheromone amount updates at the end of the final epoch, using the Rank-based Ant System algorithm.

The vendor overlap in the top ten between the different parameters is very high, with all but one setting at 90% and even one at 100%. The mean difference is always under 1.0 and if the overlap is 90%, the vendor that is unique for its list, is in the lower half of the top ten. Table 9.5 also shows that the results vary very little for the various parameter settings.

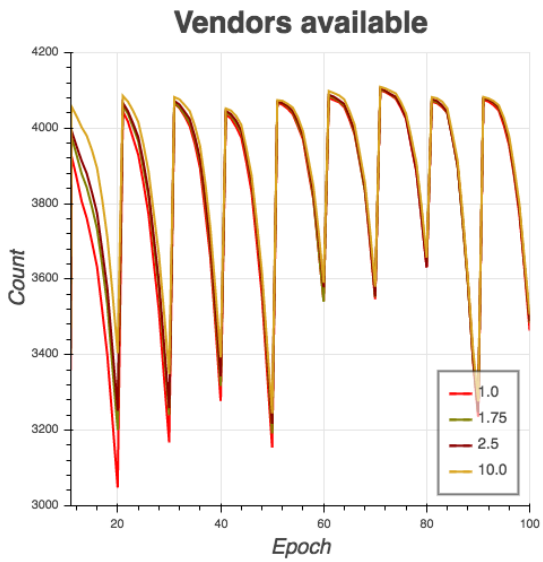


Figure 9.11: The number of vendors with books available at the end of every epoch for algorithm Rank-based Ant System.

9.4 Category importance

The settings that are used are *pheromone amount update* is 1.0, the *evaporationrate* is 0.25, the *coverage* is 0.25, there are no assignments available, but the categories' importances for qualScore and infoScore are varied. The original settings for the weights for the categories of qualScore are $0.2, 0.4, 0.2, 0.2$ and the weights for the categories of infoScore are $0.2, 0.2, 0.4, 0.2$, known as original. The model is tested when just qualScore or infoScore is present, and finally when the category importances for all values is set to an equal value of 0.25, which is called allEqual.

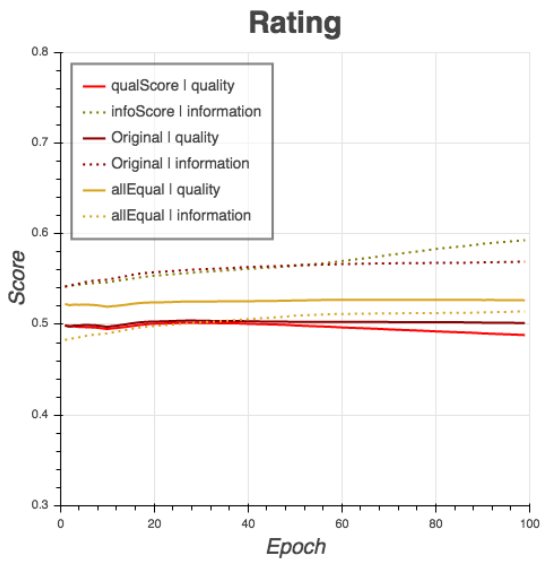


Figure 9.12: The development of the rating scores for the various algorithms with the Rank-based Ant System algorithm.

It is interesting to note that in Figure 9.12, the rating keeps increasing when just the infoScore

is taken into account. This is contrary to the qualScore rating, which is decreasing as the model progresses.

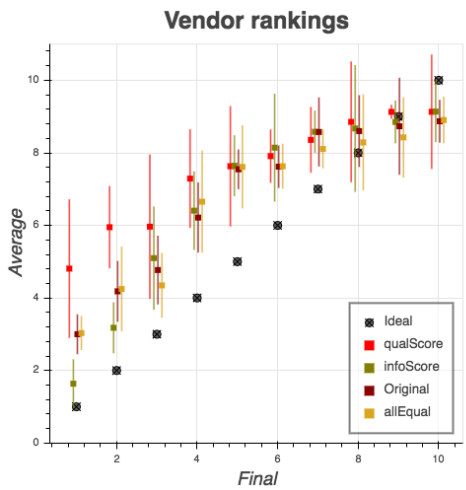


Figure 9.13: The average and ideal rank for the top ten vendors with the Ant System algorithm.

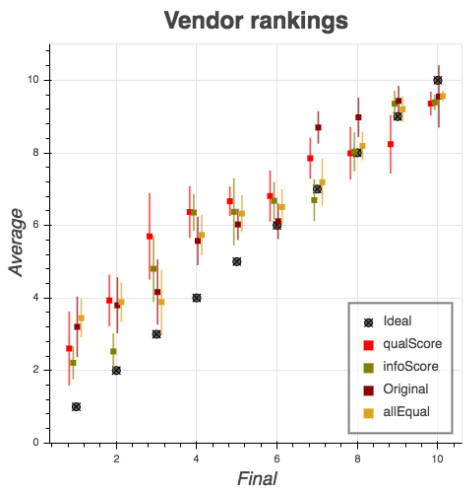


Figure 9.14: The average and ideal rank for the top ten vendors with the Rank-based Ant System algorithm.

When both scores are used, there is a 90% overlap in the top ten vendors, with a mean rank difference of 0.89 and a standard deviation of 1.10. However, if the non-overlapping category importances are used, as shown in the first two rows in Table 9.6, then only 40% overlaps, with a mean rank difference of 3.0 and standard deviation of 2.55.

	Ant System Count			Rank-based Ant System Count		
Category importance	Book	Bookkeeper	Ratio	Book	Bookkeeper	Ratio
infoScore	425007 (σ=5309)	71198 (σ=448)	5.97	474336 (σ=2641)	184737 (σ=1309)	2.57
qualScore	430328 (σ=3979)	72885 (σ=502)	5.90	448718 (σ=4429)	176919 (σ=2080)	2.54
Original	425207 (σ=3681)	72045 (σ=476)	5.90	397729 (σ=4942)	111856 (σ=2974)	3.56
allEqual	426175 (σ=3589)	71920 (σ=672)	5.93	402751 (σ=3449)	110165 (σ=1990)	3.66

Table 9.6: Table depicting the means and standard deviations of the number of books and bookkeepers for Ant System and Rank-based Ant System at the final epoch.

Table 9.6 shows that there is overlap between the first two settings, infoScore and qualScore, as well as between the third and fourth setting, Original and allEqual. The largest differences can be seen for the Rank-based Ant System algorithm.

Table 9.7 depicts the average Duplicates and Freshness. The algorithm Ant System does not show major differences for the various parameters, while for Rank-based Ant System there are two clear groups, one with infoScore and qualScore, and another with Original and allEqual. The second group has a more duplicates as well as a lower freshness.

When both scores are used, there is a 80% overlap in the top ten vendors, with a mean rank difference of 1.0 and a standard deviation of 0.5. The unique vendors are at ranks nine and ten. If

	Ant System		Rank-based Ant System	
Category importance	Duplicates	Freshness	Duplicates	Freshness
infoScore	3.83 (σ=2.51)	2.50 (σ=2.83)	3.37 (σ=2.56	2.92 (σ=3.06)
qualScore	3.78 (σ=2.50)	2.47 (σ=2.78)	3.55 (σ=2.76)	2.90 (σ=3.09)
Original	3.82 (σ=2.54)	2.52 (σ=2.85)	4.02 (σ=2.81)	2.69 (σ=3.06)
allEqual	3.81 (σ=2.51)	2.53 (σ=2.84)	3.97 (σ=2.80)	2.75 (σ=3.09)

Table 9.7: Table showing the means and standard deviations of duplicates and freshness for various category settings. This was measured at the end of the final epoch of the model.

the non-overlapping category importance are used, then there is 60% overlap, with a mean rank difference of 1.17 and standard deviation of 1.34.

9.5 Assignment feedback

For all the relevant plots, the presence of assignments is shown by *true* and absence by *false*. An assignment is given when a Bookkeeper has not had information about a book for four weeks. The settings that are used are a *pheromone amount update* of 1.0, the *evaporationrate* is 0.25, the *coverage* is set at 0.25, assignments are available, the weights for the categories of qualScore are $0.2, 0.4, 0.2, 0.2$ and the weights for the categories of infoScore are $0.2, 0.2, 0.4, 0.2$. Furthermore have the same style of plots as well as tables as in Section 9.1 been used.

Table 9.10 contains the percentage of assignments that were sent out, based on the size of the knowledge base.

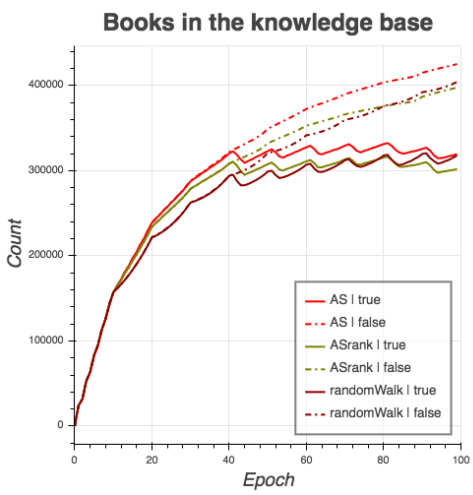


Figure 9.15: The number of books in the knowledge base with and without assignments.

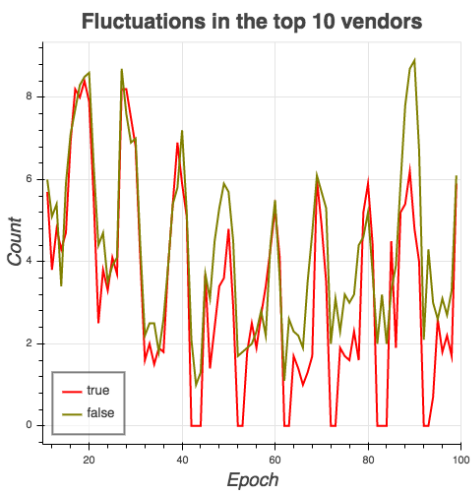


Figure 9.16: The average number of fluctuations in Rank-based Ant System for the top 10 vendors.

The Figure 9.15 shows the number of books in the knowledge base, for both the absence and presence of assignments. The top ten vendor lists contained fluctuations. For Ant System there is 70% overlap between the model with *true* and *false*, with a rank difference mean of 1.71 and standard deviation of 1.71. For Rank-based Ant System there is 90% overlap, with rank difference mean of 0.55 and standard deviation of 0.50.

Algorithm	Count			Knowledge base	
	Book	Bookkeeper	Ratio	Duplicates	Freshness
randomWalk	317662 (σ=147)	92612 (σ=158)	3.43	4.24 (σ=2.71)	1.09 (σ=1.09)
Ant System	319456 (σ=2767)	68480 (σ=336)	4.66	4.29 (σ=2.25)	1.07 (σ=1.09)
Rank-based Ant System	301815 (σ=1695)	99487 (σ=985)	3.03	4.54 (σ=2.41)	1.03 (σ=1.09)

Table 9.8: An overview for the mean and standard deviations of various algorithms with assignments available. This was measured at the end of the final epoch of the model.

Algorithm	Rating	
	Quality	Information
randomWalk	0.49 (σ=0.10)	0.54 (σ=0.08)
Ant System	0.49 (σ=0.10)	0.56 (σ=0.08)
Rank-based Ant System	0.51 (σ=0.10)	0.57 (σ=0.09)

Table 9.9: The rating score averages and standard deviations for the various algorithms with assignments available.

Algorithm	Epoch								
	10	20	30	40	50	60	70	80	90
randomWalk	0	0	0	12,47%	11,75%	12,73%	14,47%	14,45%	14,84%
Ant System	0	0	0	10,29%	9,08%	8,74%	10,18%	12,72%	10,94%
Rank-based Ant System	0	0	0	15.23%	12.93%	11,83%	12,84%	16,16%	14,86%

Table 9.10: The percentage of books in the knowledge base for which an assignment was given.

9.6 Coverage

The settings that are used are *pheromone amount update* is set at 1.0, the *evaporationrate* is 0.25, the *coverage* is 0.25, there are no assignments available, and the weights for the categories of qualScore are $0.2, 0.4, 0.2, 0.2$ and the weights for the categories of infoScore are $0.2, 0.2, 0.4, 0.2$. The same style of plots as well as tables as in Section 9.1 are used. The *coverage* is varied. Because early analysis showed results inconsistent with the hypotheses, a second run of the model was conducted with different *coverage* settings.

9.6.1 First parameter sweep

For the first run, *coverage* was varied between 0.25 and 1.0, in increments of 0.25.

Coverage (%)	randomWalk Count			Ant System Count			Rank-based Ant System Count		
	Book	Bookkeeper	Ratio	Book	Bookkeeper	Ratio	Book	Bookkeeper	Ratio
0.25	404190 (σ=603)	98186 (σ=108)	4.12	425207 (σ=3681)	72045 (σ=476)	5.90	397729 (σ=4942)	111856 (σ=2974)	3.56
0.5	580873 (σ=968)	129835 (σ=139)	4.47	630492 (σ=2709)	106321 (σ=566)	5.93	699926 (σ=4900)	201793 (σ=2756)	3.47
0.75	747664 (σ=1362)	160567 (σ=150)	4.66	799998 (σ=4882)	141523 (σ=3206)	5.65	789268 (σ=3124)	216928 (σ=250)	3.64
1.0	910669 (σ=1182)	202510 (σ=214)	4.50	951378 (σ=2966)	220020 (σ=261)	4.32	938323 (σ=4254)	223971 (σ=259)	4.19

Table 9.11: An overview for the mean and standard deviations of various algorithms for different coverages. This was measured at the end of the final epoch of the model.

Table 9.11 shows that an increase in coverage means that the ratios, which are the average sizes of the collections of the Bookkeepers, come closer together. Both Ant System and Rank-based Ant System have the highest number of Bookkeepers for the highest coverage, with Ant System having the strongest increase in Bookkeepers.

	randomWalk		Ant System		Rank-based Ant System	
Coverage (%)	Duplicates	Freshness	Duplicates	Freshness	Duplicates	Freshness
0.25	3.83 (σ=2.97)	2.33 (σ=2.60)	3.82 (σ=2.54)	2.52 (σ=2.85)	4.02 (σ=2.81)	2.69 (σ=3.06)
0.50	5.11 (σ=3.13)	1.74 (σ=2.40)	5.04 (σ=2.71)	1.90 (σ=2.64)	4.48 (σ=2.84)	2.14 (σ=2.75)
0.75	5.62 (σ=3.16)	1.62 (σ=2.42)	5.73 (σ=2.78)	1.47 (σ=2.33)	5.82 (σ=2.82)	1.74 (σ=2.70)
1.00	5.82 (σ=2.97)	1.39 (σ=2.14)	6.26 (σ=2.73)	1.17 (σ=2.08)	6.21 (σ=2.99)	1.28 (σ=2.19)

Table 9.12: The knowledge base means and standard deviations of *length* and *freshness* for various algorithms. This was measured at the end of the final epoch of the model.

An increase in the coverage shows that the number of duplicates in the knowledge base increases, and that the freshness decreases, meaning that the books are more up-to-date. This effect is strong for the Ant Colony optimization algorithms than for Random Walk.

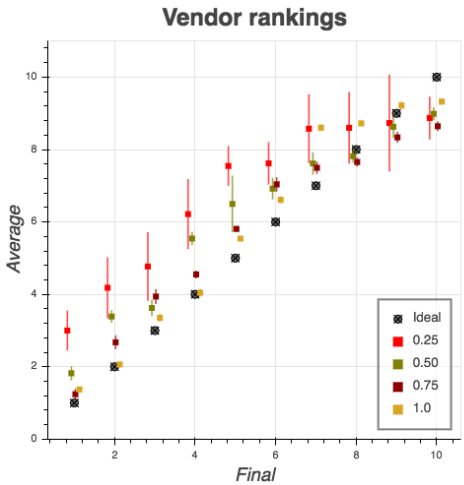


Figure 9.17: The actual and ideal ranks for the top ten vendors using Ant System.

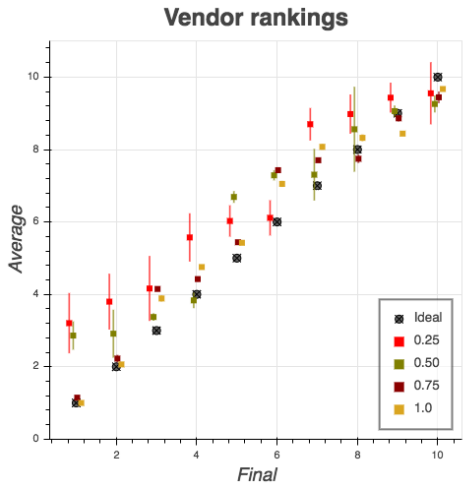


Figure 9.18: The actual and ideal ranks for the top ten vendors using Rank-based Ant System.

Both Figures 9.17 and 9.18 show that for a higher coverage, the rankings begin to approach their ideal ranks.

Coverage (%)	randomWalk	Ant System	Rank-based Ant System
0.25	0.478 (σ=0.052)	0.490 (σ=0.079)	0.487 (σ=0.087)
0.50	0.459 (σ=0.034)	0.481 (σ=0.106)	0.483 (σ=0.073)
0.75	0.431 (σ=0.031)	0.468 (σ=0.146)	0.470 (σ=0.157)
1.00	0.411 (σ=0.043)	0.460 (σ=0.189)	0.451 (σ=0.174)

Table 9.13: The average efficiency in the retrieval of a book by the Crawlers for varying coverages and algorithms. The lower the value is, the more crawlers did not retrieve a book.

Table 9.13 depicts the percentage of Crawlers that did not retrieve a book. On average, about 50% of the Crawlers are actively retrieving a book per epoch. Therefore, if every Crawler that retrieves a book, succesfully retrieves it from the database, the mean should be at 0.50%. The lower the mean is, the more Crawlers attempt to retrieve a book from an empty vendor. Due to the increasing differences in the crawler efficiency between the various algorithms at higher coverages, a second parameter sweep is conducted.

9.6.2 Second parameter sweep

For the second run, the *coverage* variable was tested for 0.01, 0.05 and 0.10, while the other parameters are kept the same as in Section 9.6.1.

Coverage (%)	randomWalk Count			Ant System Count			Rank-based Ant System Count		
	Book	Bookkeeper	Ratio	Book	Bookkeeper	Ratio	Book	Bookkeeper	Ratio
0.01	41408 (σ=104)	18805 (σ=47)	2.20	31530 (σ=742)	10278 (σ=193)	3.07	31410 (σ=700)	14214 (σ=324)	2.21
0.05	140791 (σ=166)	45342 (σ=86)	3.11	124442 (σ=3417)	26336 (σ=336)	4.73	120775 (σ=952)	39959 (σ=733)	3.02
0.10	228499 (σ=324)	64355 (σ=121)	3.55	217199 (σ=3023)	40276 (σ=464)	5.39	205540 (σ=2188)	62000 (σ=1286)	3.32

Table 9.14: An overview for the mean and standard deviations of various algorithms for different coverages. This was measured at the end of the final epoch of the model.

Coverage (%)	randomWalk		Ant System		Rank-based Ant System	
	Duplicates	Freshness	Duplicates	Freshness	Duplicates	Freshness
0.01	1.58 (σ=1.27)	3.60 (σ=2.67)	2.15 (σ=1.60)	3.90 (σ=3.06)	2.14 (σ=1.61)	3.98 (σ=3.07)
0.05	2.29 (σ=2.16)	3.21 (σ=2.73)	2.67 (σ=2.02)	3.48 (σ=3.10)	2.73 (σ=2.10)	3.60 (σ=3.16)
0.10	2.78 (σ=2.53)	2.92 (σ=2.72)	3.04 (σ=2.23)	3.14 (σ=3.02)	3.17 (σ=2.39)	3.26 (σ=3.14)

Table 9.15: The knowledge base means and standard deviations of *Duplicates* and *Freshness* for various algorithms. This was measured at the end of the final epoch of the model.

Table 9.14 depicts that for the lower coverages, the book count of the knowledge base is larger than for the Ant Colony optimization algorithms. Furthermore, Table 9.15 shows that Random Walk has the lowest freshness, but also a lower number of duplicates.

Coverage (%)	randomWalk	Ant System	Rank-based Ant System
0.01	0.499 (σ=0.276)	0.496 (σ=0.222)	0.495 (σ=0.195)
0.05	0.494 (σ=0.125)	0.494 (σ=0.146)	0.493 (σ=0.142)
0.10	0.489 (σ=0.089)	0.493 (σ=0.110)	0.492 (σ=0.103)

Table 9.16: The average efficiency of the crawlers for varying coverages and algorithms. The lower the value is, the more crawlers did not retrieve a book.

Table 9.16 also shows the efficiency of the Crawlers in retrieving books from the database.

Figures 9.19 and 9.20 show that the model reaches a local optimum every run, because the average ranks and standard deviation of the vendors is very high. Also, Rank-based Ant System stabilizes more when there is a higher coverage, while the standard deviations of the top vendors become lower.

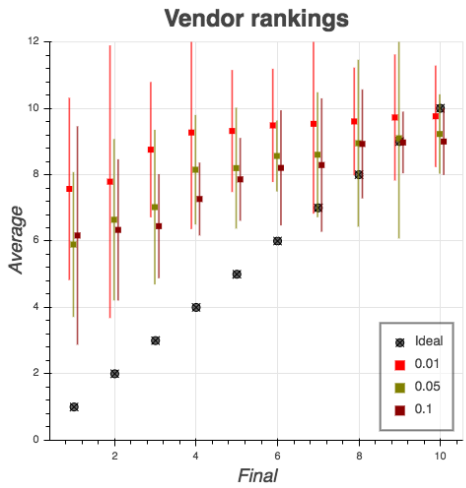


Figure 9.19: The average number of fluctuations in Ant System for the top 10 vendors.

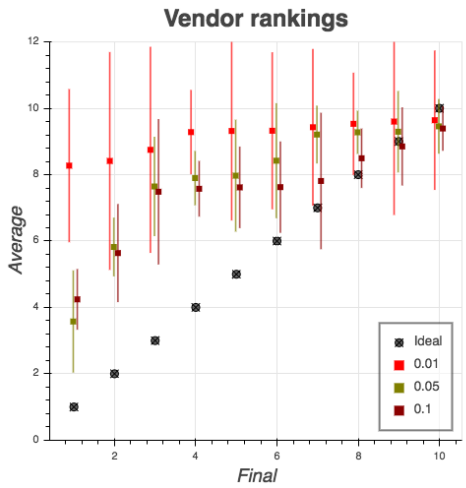


Figure 9.20: The average number of fluctuations in Rank-based Ant System for the top 10 vendors.

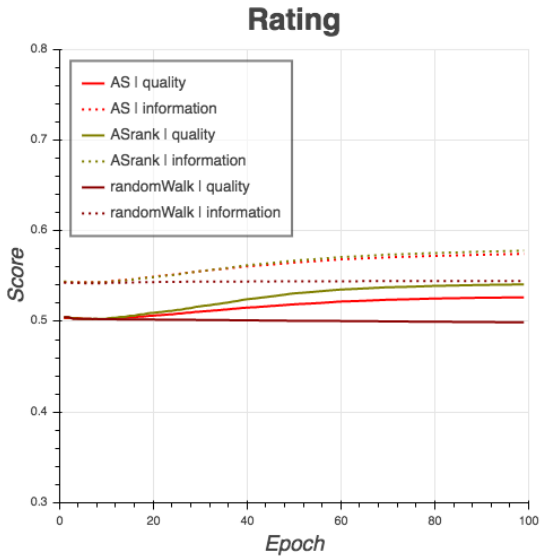


Figure 9.21: The development of the rating scores for the various algorithms with a *coverage* of 1289.

As Figure 9.21 shows, there is an increase in the rating when using Ant System and Rank-based Ant System compared to Random Walk. However, it is relatively small.

Chapter 10

Discussion

Even at the small scale that was used for this thesis, the same issues appeared both during crawling and the running of the model that appeared during crawling for search engines: the run time increased drastically because long lists had to be checked. For the crawling aspect, this resulted in the crawling becoming slower because the program had to check new URLs against increasing lists of URLs that were in the queue or already had been crawled. The model had a time bottleneck when a Crawler had to find the matching Bookkeeper. Because the model had correlation clustering, the Jaro-Winkler distance potentially had to be calculated for every author-title combination, as long as no match was found. The general settings that were used had over 32000 Crawlers, which had to find a matching Bookkeeper out of about 93000 Bookkeepers every two epochs, by using the Jaro-Winkler distance calculation.

10.1 General results

First a comparison is made between Ant System, Rank-based Ant System and Random Walk, with the general settings. These results can be seen in Section 9.1. Figure 9.1 shows the total number of books in the model over time. It is interesting to see that the opposite happens of our Hypotheses 10, 12 and 14. The expectation was that Random Walk would have a larger knowledge base than Ant System and Rank-based Ant System; this seems not to be true. Table 9.1 shows the actual book numbers in the knowledge base at the last epoch, and it shows that Ant System has the highest number of books, followed by Random Walk and finally Rank-based Ant System. This also means that the opposite of Hypothesis 15 has happened, since Rank-based Ant System has a smaller knowledge base than Ant System. Because the database is not normally distributed, with over 2000 vendors that have an average of less than ten books per week, it is likely that these vendors get visited by Random Walk, but not the others. The result is then that crawlers attempt to retrieve books from an empty vendor.

This belief is further strengthened by the curve that Random Walk shows in Figure 9.1. After every time that a new database for the week is added, Random Walk shows a u-shaped curve, where the increase in total books is smaller at the beginning of the week. Figure 9.4 depicts the number of vendors per epoch that are removed and it clearly shows for Random Walk that each time after a new database is added, a large number of vendors get removed. The figure shows a clear difference between the Ant Colony optimization algorithms and Random Walk. Whereas Random Walk clearly shows that most of the empty vendors are removed at the beginning of the epoch, the opposite happens for the two Ant Colony optimization algorithms. Because they have a preference for several vendors, first they visit them and then they start spreading to other vendors

when their preferred ones run out. The algorithm Rank-based Ant System is less focused on one vendor than Ant System and Random Walk, and therefore empties more vendors.

The Bookkeepers also show some differences between the algorithms. Table 9.1 contains the number of Bookkeepers, and the ratio is compared to the total number of books in the knowledge base. The number of Bookkeepers can be seen as the number of book types in the model, and it differs per algorithm. For the algorithm Ant System, the Bookkeepers have an average book collection of 5.90 books, while Rank-based Ant System has 3.56 and Random Walk has 4.12. This means that Ant System has a lot less Bookkeepers than Rank-based Ant System and Random Walk. The ratio between single-book and multiple-book Bookkeepers however did not appear to differ by much.

The lower number of Bookkeepers for Ant System was not expected. If just a few vendors are clearly preferred over the rest, then this would be expected to be seen in the average number of duplicates that the books have appeared in the knowledge base. These values can be seen in Table 9.1. However, there does not appear to be a real difference. On the contrary, the Ant Colony optimization algorithms have lower average number of duplicates than Random Walk. The hypothesis was that books would be found more often, because the vendors where they came from are visited more often. This does not appear to be the case for Ant System, it keeps finding new book tokens for existing book types, because the book count in the knowledge base is the highest, while the Bookkeeper count is the lowest. This could be due to the way the database was constructed; because of the number of vendors with very few books, they are emptied very fast by the Crawlers when using Random Walk. There could be a lot of overlap between the books in those vendors over the weeks, thus increasing the number of duplicates, since the same ones are found. This hypothesis has been tested and the results are discussed in Section 10.6.2.

The average number of duplicates also influences the freshness of the books, which also is shown in Table 9.1. Having preferred vendors did not mean that number of duplicate books is different. This is also the case for the freshness. In fact, Random Walk has the lowest freshness, meaning that those books are the most up to date on average. This result is, once again, likely due to the distribution of the database. Since there is a large number of vendors with few books, if these books keep overlapping every week, then that means that these books will have a lower freshness.

Figure 9.5 shows the fluctuations, which are in line with the hypotheses. These hypotheses predicted that there would be more fluctuation at the end of a week, so before a new database for the week is added. This can clearly be seen, because there is an increase in rank fluctuations in the last few epochs before a new database is added. These fluctuations are caused by the high-ranking vendors being out of books and therefore they quickly drop in pheromones, while the lower-ranked vendors increase in pheromones because they now are being visited by Crawlers.

Despite the fluctuations in the top ten vendors, there is overall not a lot of fluctuation in the final ranks. Figure 9.6 shows the final ranks. In an ideal situation, the average of the rank of a vendor would be placed on its respective "ideal" place, because then the same vendors consistently reach the same ranks. However, this is not the case. A prediction was that the models would reach local optima. That would mean higher average ranks with more standard deviation, but that is not the case here. However, there is a difference between the algorithms: Ant System has a ranking in line with ideal for the lower rankings, while Rank-based Ant System appears to have two plateaus, one for the first three ranks and another for ranks four through six. Hypotheses 11 and 13 stated that this was likely to happen. This means that Rank-based Ant System has a preference for multiple vendors, while Ant System is focused more on a single vendor. There is also a difference between the vendors in the top rankings, since only six vendors appear in both rankings. A fear was that these results might be influenced by the number of books that a vendor had, so that the vendors

with the highest number of books in their inventory would end up in the highest ranks, but this is not the case.

There is learning in preferred vendors through high quality vendors. This can be seen in Figure 9.3. Table 9.2 shows that these ratings are very close together at the end of the model. The hypothesis was that if the coverage increased, the influence of the Ant Colony optimization algorithms would decrease. However, this might already occur at a coverage of 25%. Since the original experiments did not include coverage lower than 25%, another set of experiments was done with lower coverages; these are discussed in Section 10.6.2. Since normalizing the books is also a complex process, it is likely that the current method still has too much trouble in defining the distinctive qualitative features of books and that a different method for comparison and quantification is better suited.

10.2 Evaporation rate

The results in Section 9.2 show that varying the evaporation rate has a large influence on the knowledge base. Both Ant System and Rank-based Ant System show an increased number of vendors that are removed for a lower evaporation rate. This can be seen in Figure 9.7 for Ant System and in Figure 9.9 for Rank-based Ant System. The evaporation rate is lower, meaning that less pheromones get removed at the end of the epoch, increasing the distance between ranks. Therefore, more of the crawlers focus on a few vendors, emptying them faster before moving on. For Rank-based Ant System this effect is larger: Figure 9.9 shows that the lowest points are lower than for Ant System. Furthermore, it is interesting to see that for a very high evaporation rate, the model removes very few vendors at high epochs. This is likely due to a clear preference for a single vendor.

The low evaporation rate does cause more instability for Ant System. This can be seen by the larger standard deviation for the evaporation rate of 0.1. Furthermore, the average rank is higher when the standard deviation is higher, as can be seen in Figure 9.8. This is due to the vendor rankings fluctuating more at higher rates than at lower rates, which was expected in Hypothesis 3. The standard deviation for Rank-based Ant System is also higher when the evaporation rate is lower: this can be seen in Figure 9.10.

Furthermore, it is also interesting to see how the size of the knowledge base is influenced by the evaporation rate. Table 9.3 shows the effects of the evaporation rate on the Ant System algorithm, while Table 9.4 shows the effects on the Rank-based Ant System algorithm. Both methods show an increase in the size of the knowledge base when the evaporation rate increases. This is likely due to less Crawlers attempting to retrieve books from empty vendors. The size of the Rank-based Ant System knowledge base increases more than that of the Ant System algorithm, but the number of Bookkeepers does not. The number of Bookkeepers stays within the range of 70000 to 75000 for Ant System, while increasing the knowledge base by 100000 books, but Rank-based Ant System does not only increase the knowledge base, but also creates more Bookkeepers. This results in the book count to Bookkeepers ratio increasing for Ant System, but decreasing for Rank-based Ant System. The number of duplicates decreases and the freshness increases for a higher evaporation rate. This is expected, since the size of the knowledge base also increases. Both the information and quality ratings did not really change for the various evaporation rates and therefore have been omitted.

10.3 Pheromone amount update

Dorigo et al. already the number for the pheromone amount update to be of negligible influence when using it for the traveling salesman problem [31]. However, since our model has significantly more interactions than the traveling salesman problem, it still was interesting to see if and how the pheromone amount update had an influence. The results that are of interest have been shown in Section 9.3. Because the effects were visible both in Ant System and Rank-based Ant System, only Rank-based Ant System is shown and discussed, since these effects were more visible. The relevant Ant System information can be seen in Appendix A.1.

Table 9.5 shows that the influence of the pheromone amount update on the size of the knowledge base, number of Bookkeepers, number of duplicates and freshness are negligible, as was expected in Hypothesis 2. The main difference can be seen in Figure 9.11, which shows that there is a difference in convergence speed between the algorithms. It does appear to have negligible influence on the model, as was expected.

10.4 Category importance

The results for varying the category importance are discussed in Section 9.4. Varying the category weights changes the height of the scores, but only has a limited impact on the development of the scores: this is shown in Figure 9.12. Hypothesis 9 predicted this behaviour. The rating where only the information score is taken into account does show an increasing line.

If one of the scores is no longer taken into account, then the results between these two models do not vary that significantly, which can be seen in Tables 9.6 and 9.7. However, there is a large difference when comparing them to the original or 'all categories equal' setting, as stated in Hypothesis 7. When both scores are taken into account, the book count of the knowledge base increases, but the number of duplicates lowers. The freshness also slightly increases. Furthermore, it can be seen that for Ant System there is not much difference for the number of Bookkeepers, while for Rank-based Ant System the number of Bookkeepers decreases when both scores are taken into account. There is not much difference between the two category settings of all equal and the other ones when taking both scores into account, as stated in Hypothesis 8.

A difference can also be seen when looking at the top ten rankings for both Ant System and Rank-based Ant System. When only the qualScore is taken into account, the average ranks are higher with a larger standard deviation, so there is more fluctuation between the vendor ranks. Using both scores but with different weights appears to have a small influence on the ranks, since there is only one vendor difference between the weight parameters for Ant System and Rank-based Ant System. It does have a large influence when just one of the scores is used.

10.5 Assignment feedback

The addition of assignments was expected to greatly influence the size of the knowledge base, the number of duplicates and the freshness. Figure 9.15 clearly shows that the number of books stops increasing as soon as assignment feedback becomes available. A comparison between the Tables 9.1 and 9.8 also shows the difference. Hypothesis 6 stated that this was likely to happen. The number of duplicates has increased, because the books are being checked more often, and books that no longer are available in the database, are removed from the knowledge base, which is in line with Hypothesis 5. This is also visible in the freshness of the books, it has been greatly reduced,

as was expected in Hypothesis 4. There is still not much difference between the algorithms. The option of assignments also did not influence the ratings of the books.

Since the books found through assignments did not give any positive feedback to the vendors, the expectation was that it would influence the rankings. Table 9.16 shows the fluctuations within the top ten vendors at the end of an epoch. There are no fluctuations at some epochs, which is expected because all of the Crawlers are then busy with the assignments, so the amount of pheromones drops, however, that does not influence the ranking. The result is that the overall model has less pheromones, increasing the chances for other vendors to be visited. Another effect is that the vendors that already have books in the knowledge base, will probably have less books available. Since the Crawlers perform a directed search for a book, they will take that from a vendor if available, decreasing its number of books.

Table 9.10 depicts the percentage of assignments for the size of the knowledge base. Since assignment feedback is not possible until epoch 40, it is zero before that. There is a difference between the algorithms, where Ant System has the lowest percentage of assignments, and Rank-based Ant System the highest. This difference is likely due to the difference in searching methods. The algorithm Ant System focuses on a very few vendors, making it likely that it will find all the books of that vendor, while Rank-based Ant System is more spread out. It is likely that Random Walk is lower than Rank-based Ant System because Random Walk keeps visiting vendors with only a few books. It is likely that these are the same books every week, because they are so few.

10.6 Coverage

10.6.1 First parameter sweep

The first parameter sweep went between 25% and 100% coverage of the database, in increments of 25%. Since the Crawlers can also visit vendors that no longer have books within a single epoch, it is unlikely that 100% coverage would be reached. The opposite of what was expected in Hypothesis 1 can be seen, since the algorithms do not appear to become very similar to each other. Table 9.11 shows the size of the knowledge base and the number of Bookkeepers per algorithm and coverage. It shows that a higher coverage means that the difference between the algorithms becomes smaller. The Ant Colony optimization algorithms also have 10% more Bookkeepers on average than Random Walk. The Ant System algorithm has also made a large increase in the number of Bookkeepers for the higher coverages. Table 9.12 shows the number of duplicates and freshness of the books and it can be seen that for a higher coverage, the Ant Colony optimization algorithms have a lower freshness and more duplicates. This means that the books are more recent and the same books are found more often. The knowledge base is still larger for the Ant Colony optimization algorithms than for Random Walk.

The rankings also stabilize for both Ant System and Rank-based Ant System; this is presented in Figures 9.17 and 9.18, respectively. The higher coverage shows that there is there are less fluctuations for means and the standard deviations of the average rank of the vendors. The vendors also approach their ideal ranks. However, the issue is that this could very much be influenced by the number of books that a vendor has available in its collection, so that the vendors with the highest number of books always are first. So the fear was that the influence of the algorithms is negligible because the Crawlers can theoretically reach all the vendors and their books. In order to test this assumption, a new sweep was done with a lower coverage parameter.

Since the opposite of Hypothesis 1 occurred, a possible cause was attempted to be found. A possible explanation for the current results can be deduced from Table 9.13. For higher coverages, especially the crawlers using the random walk algorithm become less efficient, meaning that they more often visit vendors that are already empty than the Ant Colony optimization algorithms. The efficiency of the Crawlers can heavily influence the results, so in order to reduce this effect, a second parameter sweep has been conducted with lower coverages. This reduced the influence of the crawler efficiency and allowed for the implemented algorithms to work more optimal.

10.6.2 Second parameter sweep

The second sweep tested for the coverage parameters of 0.01%, 0.05% and 0.1%. The expectation is that a lower coverage rate will further decrease the influence of the number of books that a vendor has present in their collection. These results are presented in Section 9.6.2.

The efficiency of the Crawlers has been increased and they are more equal between algorithms, this can be seen in Table 9.16. The average values for the successful retrieval of books by Crawlers are now closer to each other. Table 9.14 shows for the first time the expected results of a larger knowledge base for Random Walk versus the Ant Colony optimization algorithms. The Ant System algorithm still shows a significantly smaller number of Bookkeepers than for the other algorithms. But in Table 9.15 it can be seen that, while the average number of duplicates is larger for the Ant Colony optimization algorithms, it still has a higher freshness than Random Walk.

Furthermore, the expected results of a local optimum are also visible, whic is shown in Figures 9.19 and 9.20. Especially for the coverage of 0.01%, or 1289, there are high means that are close together. There are also high standard deviations, meaning that there is a lot of fluctuation in the ranks between models and epochs. This effect becomes smaller when using higher numbers of Crawlers.

The ratings also change more apparent for the lower coverage ratings. Figure 9.21 shows the average increase in the ratings over time. It shows that the Ant System and Rank-based Ant System learn and show a preference for books with a higher quality.

10.6.3 Trust and reputation

Overall the results show that there is not a large increase in scores for qualScore and infoScore over time. The results from Section 9.4 suggest that the qualScore and infoScore cancel each other out, because when just the infoScore is present, the model does show a continuous increase, depicted in Figure 9.12. Contrary to this can be seen that qualScore is decreasing, when only that score is used.

Changing the weights for the categories did not really influence Ant System, just the height of the scores. However, Rank-based Ant System showed different behaviour when using different settings. Using only the infoScore, the model has a continuous increase in its score, meaning that overall quality of the books in the knowledge base keeps increasing. Therefore, the model learns who the high-quality vendors are when using Rank-based Ant System. This is opposite to using qualScore, where the overall quality of the knowledge base is decreasing.

It does appear that the method of representation used for the quality of a book requires more fine-tuning in order to differentiate better between vendors. The advantage of representing quality

and trustworthiness of a vendor through this method is that one is no longer dependent on user-generated feedback, which is subjective and requires substantial community effort through users leaving feedback.

Chapter 11

Conclusions and further research

This chapter gives the conclusions that resulted from this research. First, the conclusions are given. Secondly, the research questions are answered and thirdly is a variety of potential further research possibilites is presented.

11.1 Conclusions

This thesis attempted to create a MAS that qualitatively scores and ranks books, and then to use that information in order to rank book vendors based on the quality of their inventory. The ranking of the books was done locally based on matching books within the author-title combinations. The main findings are:

- Information discovery and integration is a complex process with many different and complex aspects.
- Random walk currently out-performs Ant System when judging by freshness, however, optimal settings have not yet been found for the model that uses Ant System.
- Random walk currently out-performs Rank-based Ant System when judging by freshness, however, optimal settings have not yet been found for the model that uses Rank-based Ant System.
- Requesting feedback and therefore making use of a two-way information stream greatly influences the quality of the knowledge base. It reduces the size of the knowledge base. It does severely lower its freshness.
- Rank-based Ant System finds more unique author-title combinations than Ant System, but Ant System has a higher book count to Bookkeepers ratio.
- Coverage heavily influences the results. The expected results were only seen for low coverage numbers. When coverage increases, Ant System and Rank-based Ant System out-performs Random Walk. This also suggests that optimal settings have not yet been found for the model.

11.2 Research Questions

11.2.1 How does a Multi-Agent System provide a suitable solution for the issues of information discovery and integration in unstructured data?

We discuss this topic in Chapters 4 and 5, where the issues for their respective fields are named. For information discovery, the problem is that the information is unstructured and available for a limited time. Therefore, it is not known where it is located and how long the information will be available. A decentralized approach would allow a model to autonomously learn and crawl pages in a frequency that is least invasive.

A problem where MAS can be useful is for the matter of information integration, as well as information fusion. When books have to be grouped under a shared denominator such as author or title, having an agent responsible for them allows the model to create localized interactions specific to the situation. Furthermore, it also allows for easier information retrieval from a database and creates a method where the correctness of information can be handled as a two-way street. This means that information can be checked and updates can be requested on a local level, as discussed in Section 9.5.

11.2.2 Can a proof-of-concept be created that is a complete model for information discovery and integration?

The short answer is that it is possible, but IDIMAS is only a subset of a potential complete model. A complete model would have the crawler agents interact with the Internet and URLs, where the problem becomes more complex, since it is not known beforehand whether the pages found contain relevant information. For IDIMAS, a database was created that already extracted the information and represented in a structured manner.

The model IDIMAS would have to be extended with website crawling and complete in-model normalization in order to be considered a model with complete in-model information discovery and integration.

11.2.3 Can Ant Colony Optimization be used for learning the quality of databases?

The premise of this thesis was to give the books a qualitative score, that could then be used in relation to other books from an agent that contains all the books with a unique author-title combination to give an insight to the relative quality of the vendor that it came from. As is shown in Chapter 9 and discussed in Chapter 10, the model does learn based on the qualitative scores given which vendors have higher quality books. However, this is based on variables that are chosen by a human, and therefore more research is required as to what defines a good vendor and a high-quality book.

The main issue with Ant Colony optimization is that it locks into a vendor at one point. By treating the problem as an on-going process that does not have an optimal solution, that means that the model will keep trying to find the best vendor. If a vendor starts dropping in quality, then a different vendor will rise to take its place, since the model is self-reinforcing in that way. A high-ranking vendor will receive more Crawlers on average, but if the books are low quality,

then the vendor will only see a short-term return. While the books have been taken, the vendor will not receive any pheromones and this makes make it less likely that the Crawlers will make a return visit.

Does Ant Colony Optimization provide a more suitable alternative than a non-learning Multi-Agent System?

The objective of the model is to have a database with the lowest freshness values, since that means the database is most current and therefore, most likely to be up to date. Since the database that the model uses is skew, it was shown that the number of crawlers relative to the size of the database has a large influence. The general settings show that the freshness of the information was lowest for Random Walk. Since this is the version of the model that does not have any learning, it suggests that the Ant Colony optimization does not out-perform a non-learning MAS.

The use of a learning algorithm does show that the books in the database are found more often than for Random Walk. Since the Ant Colony optimization algorithms focus on a few vendors, it is likely that the information is skew due to books being in the knowledge base that have been found at the beginning. Since the model learns and therefore focuses on a few vendors, this could mean that these books have a high freshness because the vendor that they come from is no longer being visited.

11.2.4 What are requirements for a real-world implementation of a multi-agent system for information discovery and integration?

One of the major bottlenecks found during the research was processing time. Since this was a proof-of-concept, the model was not created to be executed in parallel. This meant that agents could not perform an action until after the agent before them was finished. Creating a parallel model has its own challenges, such as agents that have to wait before they can interact, in the case that the agent they are trying to reach is busy.

The other processing time bottleneck was finding the right agent. Since a Crawler did not yet know to which of the Bookkeepers it was trying to deliver a book, this would mean that it could potentially be *O(N)* per Crawler to find the Bookkeeper and deliver a book, where N is the number of Bookkeepers present.

Overall the model requires a good problem-solving algorithm for information retrieval. Afterwards the relevant information has to be extracted correctly, which then needs to be normalized. The resulting data objects then have to be compared and clustered based on overlapping linked features. These are all steps that have their own complex challenges and approaches in order to solve the subsets of the overarching problem.

11.3 Further research

The dataset itself could use improvement, since it was created before the model. This was done in order to create a dataset that could be used in order as a benchmark tester. The issue is that the data in the domain of booksx can be seen as an infinitely wide data stream, where each stream has its window. Using a very general crawler for two websites could mean that the resulting dataset therefore was too diffuse.

The dataset could also be formatted better, by selection of a better subset of the crawled data. When crawling a database with a number of crawlers that was based on sum of books in the database, then the vendors with more books are at an advantage when the coverage rate is high enough. This would mean that a more balanced dataset had to be created with vendors that have not only around the same number of books, but also through crawling on specific sets of books of which multiple editions and sets exist, in order to clearly see the effects of the Ant System and Rank-based Ant System algorithms.

Secondly, the information extraction methods could be improved. Part of the relevant book information is placed in the *details* section of a detail page, for which only a basic string matcher was used that looked for the 'signed by author' phrase. Other relevant information is still present, but was not used for this thesis. A more comprehensive natural language approach would have to be taken in order to extract that information. Afterwards, the information was normalized. The method that was used in this thesis could also be expanded on, because that would mean that the books can be better rated qualitatively. An issue here was again that natural language is used to describe aspects of a book. The quality of a book could be described as *good*, but then also include a detailed explanation of any tears or stains that it might have.

This also influences the fusion of books under a agent with a unique author-title combination. The method for book fusion is contained to finding an earlier found book based on its identifier, and clustering books based on their author and title. It is possible that different sources contain the vendors, however this is something that was not looked at beyond clustering vendors based on their names. So it is possible that a vendor has books in their database twice, because they came from different sources.

The method for deciding the quality and information of a book in the scheme of the model is arbitrary. It has been done based on human-chosen variables. Due to the scale of the process, is this also something that could be tested and expanded on. Not just on deciding which features are distinctive for the category that one is looking for, but also how many features should influence the final result.

The model could also be expanded by incorporating the normalization into the model. The same is possible for the crawling of the information. A database was created and crawlers interacted with that database, however, would it be an improvement if the model interacted with the Internet. This is a complex step, however, an improvement would be to have a database of HTML pages, containing both relevant and non relevant pages, that could be used as a basis for the model to not only learn which vendors are of high quality, but also which web pages or even style of web pages can be crawled.

The model could also benefit from splitting the Crawler, by explicitly differentiating between the tasks of information retrieval and extraction. More focus could then be placed on the quality of the books retrieved, since the model suffers from being in a local optimum. Vendors that have few but high-quality books are passed on in this model, because the pheromone trail does not have time to develop.

Finally, the model be can also be improved using a different Ant Colony optimization, by implementing another relevant variation such as Max-Min Ant System. It is also an interesting approach to use a different colony optimization algorithm, like the Bees algorithm [82]. This colony algorithm has several classes of agents, where one of the classes is the scout. The scout visits sources and retrieves samples. It then judges the sample based on quality. If the quality of the sample is high enough, the scout then goes to the 'dance room', where it performs a dance in order to recruit harvester agents. Based on the quality and size of the source, the length of the

dance is increased. The longer the dance, the higher the quality of the source and amount of food available. This in turn recruits more workers to follow the scout back to the source. It has been used for a range of problems, one of which is data clustering [81]. Another use for Bees algorithm was for task allocation [63]. This approach provides an alternative for preventing the model from reaching a local optimum as well as adjusting to changing quality of vendors.

Appendix A

Plots and Tables

A.1 Pheromone amount update

	Count			Knowledge base	
Algorithm	Book	Bookkeeper	Ratio	Duplicates	Freshness
1.00	425207 (σ=3681)	72045 (σ=476)	5.90	3.82 (σ=2.54)	2.52 (σ=2.85)
1.75	422173 (σ=3067)	70991 (σ=507)	5.95	3.85 (σ=2.54)	2.50 (σ=2.83)
2.50	422251 (σ=5299)	71201 (σ=299)	5.93	3.85 (σ=2.54)	2.50 (σ=2.84)
10.0	418086 (σ=4948)	70510 (σ=753)	5.93	3.88 (σ=2.57)	2.47 (σ=2.82)

Table A.1: The means and standard deviations for the various pheromone amount updates at the final epoch, using the Ant System algorithm.

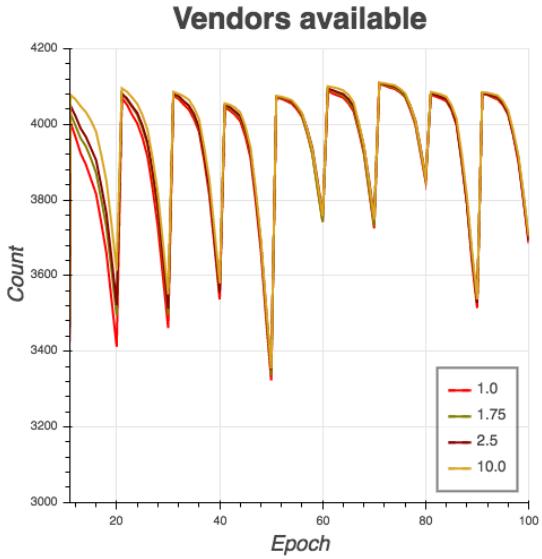


Figure A.1: The number of vendors available at the end of each epoch for various pheromone amount update for Ant System.

A.2 Raw book numbers

Search term and source	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Total
abebooks_clas_fic	86800	86315	85144	87833	85969	85487	82784	94875	95523	93929	884659
abebooks_clas_lit	138678	145195	145645	141333	145312	143287	160022	162374	159613	161001	1502460
biblio_am_fic	23513*	77418	72118	77978	65912	77106	78089	75723	71444	81664	700965
biblio_cla_lit	190641	185786	173158	160324	148993	183957	179133	238399	151279	162399	1774069
biblio_en_fic	43551	34112	33209	35682	38794	33412	29554	24198	37376	28353	338241
biblio_mo_fic	96143	131997	132443	139666	95771	143980	133378	127520	120258	123373	1244529
Total	579326	660823	641717	642816	580751	667229	662960	723089	635493	650719	6444923

Table A.2: Final number of books per crawler source and category search term. The week stands for the calender week in 2017 from which it was obtained. ∗Even though the result from this date is significantly lower than the average of the rest of the weeks for that crawler, it was kept in as its influence overall is minimal.

Bibliography

[1] R. Albert, H. Jeong, and A. Barabási. "Internet: diameter of the world-wide web". In: *nature* 401.6749 (1999), pp. 130–131.

[2] *Amazon website*. http://www.amazon.com. Accessed: 2017-01-27.

[3] N. Bansal, A. Blum, and S. Chawla. "Correlation clustering". In: *Machine Learning* 56.1 (July 2004), pp. 89–113. DOI: 10.1023/B:MACH.0000033116.57574.95.

[4] C. Batini and M. Scannapieco. *Data and Information Quality: Dimensions, Principles and Techniques*. Springer, 2016.

[5] T. Berners-Lee and M. Fischetti. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. HarperInformation, 2000. ISBN: 006251587X.

[6] J. Bleiholder and F. Naumann. "Data fusion". In: *ACM Comput. Surv.* 41.1 (Jan. 2009), 1:1–1:41. DOI: 10.1145/1456650.1456651.

[7] C. Blum. "Ant colony optimization: introduction and recent trends". In: *Physics of Life Reviews* 2.4 (2005), pp. 353 –373. DOI: 10.1016/j.plrev.2005.10.001.

[8] C. Blum. "Beam-ACOhybridizing ant colony optimization with beam search: an application to open shop scheduling". In: *Computers & Operations Research* 32.6 (2005), pp. 1565 –1591. DOI: 10.1016/j.cor.2003.11.018.

[9] C. Blum and M. Dorigo. "The hyper-cube framework for ant colony optimization". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 34.2 (2004), pp. 1161–1172. DOI: 10.1109/TSMCB.2003.821450.

[10] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: from natural to artificial systems*. 1. Oxford university press, 1999.

[11] R. Bouffanais. *Design and control of swarm dynamics*. Springer, 2016.

[12] S. Brin and L. Page. "Reprint of: The anatomy of a large-scale hypertextual web search engine". In: *Computer Networks* 56.18 (2012), pp. 3825 –3833. DOI: 10.1016/j.comnet.2012.10.007.

[13] B. Bullnheimer, R. F. Hartl, and C. Strauss. "A new rank based version of the Ant System. A computational study." In: (1997).

[14] B. Bullnheimer, R.F. Hartl, and C. Strauss. "An improved Ant System algorithm for the vehicle routing problem". In: *Annals of Operations Research* 89.0 (Jan. 1999), pp. 319–328. DOI: 10.1023/A:1018940026670.

[15] J. Carter, E. Bitting, and A. A. Ghorbani. "Reputation formalization for an information-sharing MultiAgent System". In: *Computational Intelligence* 18.4 (2002), pp. 515–534. DOI: 10.1111/1467-8640.t01-1-00201.

[16] C. Castelfranchi and R. Falcone. *Trust Theory: A Socio-Cognitive and Computational Model*. 1st. Wiley Publishing, 2010. ISBN: 0470028750, 9780470028759.

[17] C. Castelfranchi and F. Paglieri. "The role of beliefs in goal dynamics: prolegomena to a constructive theory of intentions". In: *Synthese* 155.2 (Mar. 2007), pp. 237–263. DOI: 10.1007/s11229-006-9156-3.

[18] H. Chen, R. H. L. Chiang, and V. C. Storey. "Business intelligence and analytics: from big data to big impact". In: *MIS quarterly* 36.4 (2012), pp. 1165–1188.

[19] J. Cho and H. Garcia-Molina. "Parallel Crawlers". In: *Proceedings of the 11th International Conference on World Wide Web*. WWW '02. Honolulu, Hawaii, USA: ACM, 2002, pp. 124–135. ISBN: 1-58113-449-5. DOI: 10.1145/511446.511464.

[20] G. G. Chowdhury. "Natural language processing". In: *Annual Review of Information Science and Technology* 37.1 (2003), pp. 51–89. DOI: 10.1002/aris.1440370103.

[21] R. Collobert and J. Weston. "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: ACM, 2008, pp. 160–167. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390177.

[22] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. "Natural language processing (almost) from scratch". In: *Journal of Machine Learning Research* 12.Aug (2011), pp. 2493–2537.

[23] R. Conte and M. Paolucci. *Reputation in artificial societies: social beliefs for social order*. Kluwer Academic Publishers, 2002.

[24] R. Cooley, B. Mobasher, and J. Srivastava. "Web mining: information and pattern discovery on the World Wide Web". In: *Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence*. Nov. 1997, pp. 558–567. DOI: 10.1109/TAI.1997.632303.

[25] O. Cordon, I. F. de Viana, F. Herrera, and L. Moreno. "A new ACO model integrating evolutionary computation concepts: The best-worst Ant System". In: *Abstract proceedings of ANTS 2000 From Ant Colonies to Artificial Ants: Second International Workshop on Ant Algorithms*. Bruxelles, Belgium, 2000, pp. 22–29.

[26] B. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in Practice*. 1st. USA: Addison-Wesley Publishing Company, 2009. ISBN: 0136072240, 9780136072249.

[27] D. L. Davies and D. W. Bouldin. "A cluster separation measure". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1.2 (1979), pp. 224–227. DOI: 10.1109/TPAMI.1979.4766909.

[28] J. L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels. "The self-organizing exploratory pattern of the Argentine ant". In: *Journal of Insect Behavior* 3.2 (Mar. 1990), pp. 159–168. DOI: 10.1007/BF01417909.

[29] L. Derczynski, D. Maynard, G. Rizzo, M. van Erp, G. Gorrell, R. Troncy, J. Petrak, and K. Bontcheva. "Analysis of named entity recognition and linking for tweets". In: *Information Processing & Management* 51.2 (2015), pp. 32 –49. DOI: 10.1016/j.ipm.2014.10.006.

[30] A. van Deursen, A. Mesbah, and A. Nederlof. "Crawl-based analysis of web applications: prospects and challenges". In: *Science of Computer Programming* 97 (2015), pp. 173 –180. DOI: 10.1016/j.scico.2014.09.005.

[31] M. Dorigo. "Optimization, Learning and Natural Algorithms". PhD thesis. 1992. URL: http://ci.nii.ac.jp/naid/10027631617/en/.

[32] M. Dorigo, E. Bonabeau, and G. Theraulaz. "Ant algorithms and stigmergy". In: *Future Generation Computer Systems* 16.8 (2000), pp. 851 –871. DOI: 10.1016/S0167-739X(00)00042-X.

[33] M. Dorigo and L. M. Gambardella. "Ant colony system: a cooperative learning approach to the traveling salesman problem". In: *IEEE Transactions on Evolutionary Computation* 1.1 (Apr. 1997), pp. 53–66. DOI: 10.1109/4235.585892.

[34] M. Dorigo, V. Maniezzo, and A. Colorni. "Ant system: optimization by a colony of cooperating agents". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.1 (Feb. 1996), pp. 29–41. DOI: 10.1109/3477.484436.

[35] M. Dorigo and T. Stützle. "Ant Colony Optimization: Overview and Recent Advances". In: *Handbook of Metaheuristics*. Ed. by M. Gendreau and J.Y. Potvin. Boston, MA: Springer US, 2010, pp. 227–263. ISBN: 978-1-4419-1665-5. DOI: 10.1007/978-1-4419-1665-5_8.

[36] M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, and A. Winfield. *Ant Colony Optimization and Swarm Intelligence: 6th International Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008, Proceedings (Lecture ... Computer Science and General Issues)*. 1st ed. Springer Publishing Company, Incorporated, 2008. ISBN: 9783540875260.

[37] B. M. Dunin-Keplicz and R. Verbrugge. *Teamwork in Multi-Agent Systems: A Formal Approach*. 1st. Wiley Publishing, 2010. ISBN: 0470699884, 9780470699881.

[38] *Ebay website*. http://pages.ebay.com/help/feedback/howitworks.html. Accessed: 2017-01-27.

[39] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. "Data exchange: semantics and query answering". In: *Theoretical Computer Science* 336.1 (2005), pp. 89 –124. DOI: 10.1016/j.tcs.2004.10.033.

[40] R. Falcone and C. Castelfranchi. *Social Trust: A Cognitive Approach*. Ed. by C. Castelfranchi and Y-H. Tan. Dordrecht: Springer Netherlands, 2001, pp. 55–90. ISBN: 978-94-017-3614-5. DOI: 10.1007/978-94-017-3614-5_3.

[41] W. Fan and F. Geerts. *Foundations of Data Quality Management*. Morgan & Claypool Publishers, 2012.

[42] A. M. Fard and A. Mesbah. "Feedback-Directed Exploration of Web Applications to Derive Test Models." In: *ISSRE*. Vol. 13. 2013, pp. 278–287.

[43] B. J. Frey and D. Dueck. "Clustering by passing messages between data points". In: *Science* 315.5814 (2007), pp. 972–976. DOI: 10.1126/science.1136800.

[44] S. Subramanian K. Kawakami G. Lample M. Ballesteros and C. Dyer. "Neural architectures for named entity recognition". In: *CoRR* abs/1603.01360 (2016). URL: http://arxiv.org/abs/1603.01360.

[45] J. Gama. *Knowledge Discovery from Data Streams*. Chapman and Hall/CRC, 2010. ISBN: 978-1-4398-2611-9. DOI: 10.1201/EBK1439826119-c3.

[46] L. M. Gambardella and M. Dorigo. "Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem". In: Morgan Kaufmann, 1995, pp. 252–260.

[47] F. Geerts, G. Mecca, P. Papotti, and D. Santoro. "Mapping and cleaning". In: *2014 IEEE 30th International Conference on Data Engineering*. Mar. 2014, pp. 232–243. DOI: 10.1109/ICDE.2014.6816654.

[48] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. "Self-organized shortcuts in the Argentine ant". In: *Naturwissenschaften* 76.12 (Dec. 1989), pp. 579–581. DOI: 10.1007/BF00462870.

[49] A. Gruenheid, X. L. Dong, and D. Srivastava. "Incremental record linkage". In: *Proc. VLDB Endow.* 7.9 (May 2014), pp. 697–708. DOI: 10.14778/2732939.2732943.

[50] M. Guntsch and M. Middendorf. "A Population Based Approach for ACO". In: *Applications of Evolutionary Computing: EvoWorkshops 2002: EvoCOP, EvoIASP, EvoSTIM/EvoPLAN Kinsale, Ireland, April 3–4, 2002 Proceedings*. Ed. by S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, and G. R. Raidl. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 72–81. ISBN: 978-3-540-46004-6. DOI: 10.1007/3-540-46004-7_8.

[51] M. Guntsch and M. Middendorf. "Applying Population Based ACO to Dynamic Optimization Problems". In: *Ant Algorithms: Third International Workshop, ANTS 2002 Brussels, Belgium, September 12–14, 2002 Proceedings*. Ed. by M. Dorigo, G. Di Caro, and M. Sampels. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 111–122. ISBN: 978-3-540-45724-4. DOI: 10.1007/3-540-45724-0_10.

[52] J. Han, J. Pei, and M. Kamber. *Data Mining: Concepts and Techniques*. Elsevier, 2011.

[53] A. Herrouz, C. Khentout, and M. Djoudi. "Overview of web content mining tools". In: *CoRR* abs/1307.1024 (2013). URL: http://arxiv.org/abs/1307.1024.

[54] A. Herzig, E. Lorini, J. F. Hübner, J. Ben-Naim, C. Castelfranchi, R. Demolombe, D. Longin, L. Vercouter, and O. Boissier. "Prolegomena for a logic of trust and reputation". In: *3rd International Workshop on Normative Multiagent Systems (NorMAS 2008)*. Ed. by G. Boella, G. Pigozzi, M. P. Singh, and H. Verhagen. Luxembourg, France, July 2008. URL: https://hal.archives-ouvertes.fr/hal-00405595.

[55] J. Hoffer, R. Venkataraman, and H. Topi. *Modern Database Management*. 12th. Upper Saddle River, NJ, USA: Prentice Hall Press, 2015. ISBN: 0133544613, 9780133544619.

[56] W. H. Inmon. *Building the data warehouse*. John Wiley & Sons, 2005.

[57] *ISBN number explanation*. https://www.isbn-international.org/content/what-isbn. Accessed: 2017-05-10.

[58] P. S. Jacobs. *Text-based intelligent systems: Current research and practice in information extraction and retrieval*. Psychology Press, 1992.

[59] A. K. Jain. "Data clustering: 50 years beyond K-means". In: *Pattern Recognition Letters* 31.8 (2010). Award winning papers from the 19th International Conference on Pattern Recognition (ICPR), pp. 651 –666. DOI: 10.1016/j.patrec.2009.09.011.

[60] M. A. Jaro. "Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida". In: *Journal of the American Statistical Association* 84.406 (1989), pp. 414–420. DOI: 10.1080/01621459.1989.10478785.

[61] M. A. Jaro. "Probabilistic linkage of large public health data files". In: *Statistics in Medicine* 14.5-7 (1995), pp. 491–498. DOI: 10.1002/sim.4780140510.

[62] N. R. Jennings, K. Sycara, and M. Wooldridge. "A roadmap of agent research and development". In: *Autonomous Agents and Multi-Agent Systems* 1.1 (Jan. 1998), pp. 7–38. DOI: 10.1023/A:1010090405266.

[63] A. Jevtic, . Gutierrez, D. Andina, and M. Jamshidi. "Distributed bees Algorithm for task allocation in swarm of robots". In: *IEEE Systems Journal* 6.2 (2012), pp. 296–304. DOI: 10.1109/JSYST.2011.2167820.

[64] A. Jøsang, C. Keser, and T. Dimitrakos. "Can We Manage Trust?" In: *Trust Management: Third International Conference, iTrust 2005, Paris, France, May 23-26, 2005. Proceedings*. Ed. by P. Herrmann, V. Issarny, and S. Shiu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 93–107. ISBN: 978-3-540-32040-1. DOI: 10.1007/11429760_7.

[65] R. Kimball and M. Ross. *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. John Wiley & Sons, 2013.

[66] K. Kravari, E. Kontopoulos, and N. Bassiliades. "EMERALD: A Multi-Agent System for Knowledge-Based Reasoning Interoperability in the Semantic Web". In: *Artificial Intelligence: Theories, Models and Applications: 6th Hellenic Conference on AI, SETN 2010, Athens, Greece, May 4-7, 2010. Proceedings*. Ed. by S. Konstantopoulos, S. Perantonis, V. Karkaletsis, C. D. Spyropoulos, and G. Vouros. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 173–182. ISBN: 978-3-642-12842-4. DOI: 10.1007/978-3-642-12842-4_21.

[67] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2014.

[68] V. I. Levenshtein. "Binary codes capable of correcting deletions, insertions, and reversals". In: 10.8 (1966), pp. 707–710.

[69] T. Liao, T. Sttzle, M. A. Montes de Oca, and M. Dorigo. "A unified ant colony optimization algorithm for continuous optimization". In: *European Journal of Operational Research* 234.3 (2014), pp. 597 –609. DOI: 10.1016/j.ejor.2013.10.024.

[70] D. Maniezzo, M. Dorigo, V. Maniezzo, and A. Colorni. "Ant system: an autocatalytic optimizing process, technical report 91-016 revised". In: (1991).

[71] V. Maniezzo. "Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem". In: *INFORMS journal on computing* 11.4 (1999), pp. 358–369.

[72] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008. ISBN: 0521865719, 9780521865715.

[73] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. "The Stanford CoreNLP Natural Language Processing Toolkit". In: *Association for Computational Linguistics (ACL) System Demonstrations*. 2014, pp. 55–60. URL: http://www.aclweb.org/anthology/P/P14/P14-5010.

[74] P. McBurney, D. Hitchcock, and S. Parsons. "The eightfold way of deliberation dialogue". In: *International Journal of Intelligent Systems* 22.1 (2007), pp. 95–132. DOI: 10.1002/int.20191.

[75] P. McBurney and S. Parsons. "Chance Discovery Using Dialectical Argumentation". In: *New Frontiers in Artificial Intelligence: Joint JSAI 2001 Workshop Post-Proceedings*. Ed. by T. Terano, Y. Ohsawa, T. Nishida, A. Namatame, S. Tsumoto, and T. Washio. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 414–424. ISBN: 978-3-540-45548-6. DOI: 10.1007/3-540-45548-5_57.

[76] F. Menczer, G. Pant, and P. Srinivasan. "Topical web crawlers: evaluating adaptive algorithms". In: *ACM Trans. Internet Technol.* 4.4 (Nov. 2004), pp. 378–419. DOI: 10.1145/1031114.1031117.

[77] J. S. i Mir, Jordi, and C. S. i Garca. "Trust and reputation for agent societies". Consultable des del TDX. PhD thesis. Bellaterra: Universitat Autnoma de Barcelona, 2004. URL: https://ddd.uab.cat/record/36586.

[78] J. S. i Mir and C. Sierra. "Review on computational trust and reputation models". In: *Artificial Intelligence Review* 24.1 (Sept. 2005), pp. 33–60. DOI: 10.1007/s10462-004-0041-5.

[79] S. M. Mirtaheri, M. E. Dinçtürk, S. Hooshmand, G. V. Bochmann, G-V. Jourdan, and I. V. Onut. "A Brief History of Web Crawlers". In: *Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research*. CASCON '13. Ontario, Canada: IBM Corp., 2013, pp. 40–54. URL: http://dl.acm.org/citation.cfm?id=2555523.2555529.

[80] J. Nothman, N. Ringland, W. Radford, T. Murphy, and J. R. Curran. "Learning multilingual named entity recognition from Wikipedia". In: *Artificial Intelligence* 194 (2013). Artificial Intelligence, Wikipedia and Semi-Structured Resources, pp. 151 –175. DOI: 10.1016/j.artint.2012.03.006.

[81] D. T. Pham, S. Otri, A. Afify, M. Mahmuddin, and H. Al-Jabbouli. "Data clustering using the bees algorithm". In: *Proceedings of 40th CIRP international manufacturing systems seminar*. 2007.

[82] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi. "The bees algorithm-A novel tool for complex optimisation". In: *Intelligent Production Machines and Systems-2nd I* PROMS Virtual International Conference (3-14 July 2006)*. sn. 2011.

[83] I. Pinyol and J. S. i Mir. "Computational trust and reputation models for open multi-agent systems: a review". In: *Artificial Intelligence Review* 40.1 (June 2013), pp. 1–25. DOI: 10.1007/s10462-011-9277-z.

[84] L. Popa, Y. Velegrakis, M. A. Hernández, R. J. Miller, and R. Fagin. "Translating Web Data". In: *Proceedings of the 28th International Conference on Very Large Data Bases*. VLDB '02. Hong Kong, China: VLDB Endowment, 2002, pp. 598–609. URL: http://dl.acm.org/citation.cfm?id=1287369.1287421.

[85] B. Roberts. *Spiders*. https://github.com/buckyroberts/Spider. 2016.

[86] M. Sanderson and W. B. Croft. "The history of information retrieval research". In: *Proceedings of the IEEE* 100.Special Centennial Issue (May 2012), pp. 1444–1451. DOI: 10.1109/JPROC.2012.2189916.

[87] S. Sarawagi and W. W. Cohen. "Semi-Markov Conditional Random Fields for Information Extraction". In: *Advances in neural information processing systems*. 2005, pp. 1185–1192.

[88] J. Srivastava, R. Cooley, M. Deshpande, and P.N. Tan. "Web usage mining: discovery and applications of usage patterns from web data". In: *SIGKDD Explor. Newsl.* 1.2 (Jan. 2000), pp. 12–23. DOI: 10.1145/846183.846188.

[89] International Organization for Standardization/International Electrotechnical Commission et al. "Software engineering-Software product Quality Requirements and Evaluation (SQuaRe) data quality model". In: *ISO/IEC* 25012 (2008), pp. 1–13.

[90] G. Stumme, A. Hotho, and B. Berendt. "Semantic web mining". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 4.2 (2006), pp. 124 –143. DOI: 10.1016/j.websem.2006.02.001.

[91] T. Stützle and H. H. Hoos. "MAXMIN Ant System". In: *Future Generation Computer Systems* 16.8 (2000), pp. 889 –914. DOI: 10.1016/S0167-739X(00)00043-1.

[92] P. Suthar and B. Oza. "A survey of web usage mining techniques". In: *International Journal of Computer Science and Information Technologies.(IJCSIT)* 6.6 (2015).

[93] C. Sutton and A. McCallum. "An introduction to conditional random fields". In: *Foundations and Trends® in Machine Learning* 4.4 (2012), pp. 267–373. DOI: 10.1561/2200000013.

[94] C. W. Tsai, C. F. Lai, M. C. Chiang, and L. T. Yang. "Data mining for internet of things: A Survey". In: *IEEE Communications Surveys Tutorials* 16.1 (Jan. 2014), pp. 77–97. DOI: 10.1109/SURV.2013.103013.00206.

[95] D. N. Walton and E. C. W. Krabbe. *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. SUNY press, 1995.

[96] W. E. Winkler. *The state of record linkage and current research problems*. Tech. rep. Statistical Research Division, U.S. Census Bureau, 1999.

[97] M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley & Sons, 2009.

[98] X.S. Yang, Z. Cui, R. Xiao, A. H. Gandomi, and M. Karamanoglu. *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*. 1st. Amsterdam, The Netherlands, The Netherlands: Elsevier Science Publishers B. V., 2013. ISBN: 0124051634, 9780124051638.

[99] G. Zacharia. "Collaborative reputation mechanisms for online communities". PhD thesis. Massachusetts Institute of Technology, 1999.