CONNECTING PEOPLE USING LATENT SEMANTIC ANALYSIS FOR KNOWLEDGE SHARING

MASTER THESIS REPORT

SUBMITTED BY

David Muchangi Mugo

Matri.No:34348

TUHH SUPERVISORS

Prof. Dr. rer. –nat. habil. Ralf Möller

Prof. Dr. rer. pol. Kathrin Fischer

Dipl.-Ing. Sylvia Melzer

COMPANY SUPERVISOR

Dr. Bastian Maier

This report is submitted in partial fulfillment of the requirements for the Master of Science degree in Media and Information Technologies

Hamburg, January 2010

Abstract

A shift from technology-oriented knowledge management to people-oriented knowledge management is indispensable. To achieve this, organizations must understand the nature of knowledge. In this work, knowledge has been found to be both a process and a collection of artifacts. This makes knowledge and the knower to be two inseparable entities. Consequently, the appropriate way to share both the explicit and the implicit knowledge components is through people-with-people connection. However, from existing barriers like location and time differences among others, people-with-documents connection is proposed as an intermediate step. The investigation of latent semantic analysis (LSA) in achieving people-with-documents connection has revealed decreased precision performance at higher recall performance. A solution to include annotations in the technique has been proposed to refine knowledge representation into the LSA technique. Annotation process based on domain ontologies has been proposed to compliment the LSA knowledge mining process from documents with domain knowledge represented by ontologies.

Declaration

I hereby declare that I have personally authored this report and that no other sources other than the ones listed at the end of the report have been used. The report is therefore, original and has not been published in Germany or in any other country. All the registered trademarks, names, and icons appearing in this report are the property of their respective owners.

David Muchangi Mugo
15th January 2010

Acknowledgement

I am very grateful to Prof. Ralf Möller for helping me with the definition and the organization of the topic within a company and his guidance throughout the work. On the same breadth, I appreciate the guidance I received from Sylvia Melzer in every step of this work. I also thank Prof. Kathrin Fischer for accepting to review my work as a second supervisor. My other special appreciation goes to the management of Grindaix GmbH specifically Dr. Bastian Maier and Dr. Dirk Friedrich for providing me with the challenge to carry out my research work within the company. Lastly but not the least, I thank all my colleagues in Grindaix GmbH and WZL department of RWTH-Aachen for their understanding and support throughout the entire work period.

Table of Contents

List of Figures

List of Tables

1. Introduction

This chapter introduces briefly the concept of knowledge and its importance in creating organization's competitive advantage. The chapter further introduces the motivation and objectives of this work in achieving knowledge sharing and the role of latent semantic analysis in the process.

1.1. Knowledge

Knowledge has become one of the most important resources for the success of any corporation. According to Oxford English Dictionary knowledge is the expertise and skills acquired by a person through experience or education [38]. In the corporate world, knowledge is one of the most important assets for creating and sustaining a competitive advantage. This is demonstrated below:

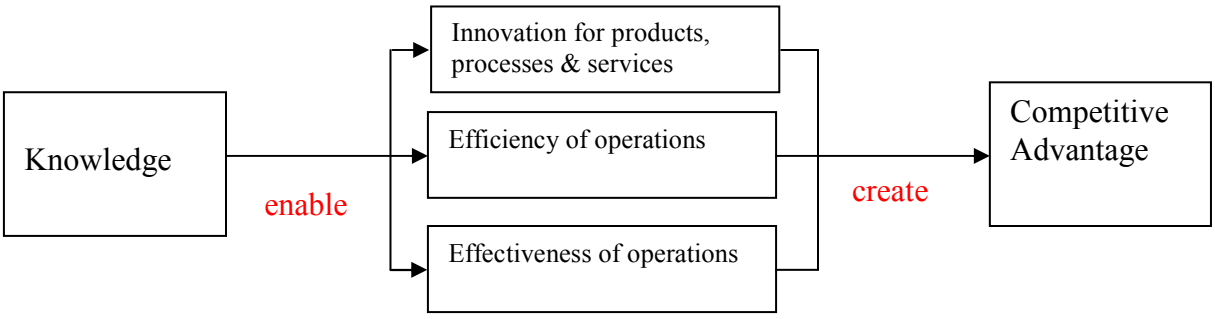


Figure 1.1: Creation of competitive advantage using knowledge

Despite the importance of knowledge, its nature remains unclear due to the complexity of human brain from which it is primarily generated and stored. Many views continue to emerge regarding the nature of knowledge and its representation in human brain among psychologists and cognitive neuroscientists. This is backed by arguments for understanding the nature of knowledge before attempts to share it are done [14]. Despite concerns about the true nature of knowledge, sharing of knowledge remains a major objective for many organizations in order to create competitive advantage. Knowledge

sharing refers to the process where individuals mutually exchange their knowledge and jointly create new knowledge [11]. Knowledge consists of two components, tacit component and explicit component. Explicit component can be communicated directly amongst people but tacit component cannot be communicated although it can be learned through doing or observing [27].

Can we therefore capture the explicit component of people's knowledge and share it without their involvement? This question has brought about a tag of war between 2 schools of thoughts, the objectivists and subjectivists [47]. The objectivists believe that knowledge can be separated into 2 distinct types, tacit and explicit. Their argument suggests that explicit knowledge can be captured into computer based information systems and used without its knower. According to them, tacit knowledge can also be shared if efforts are made to convert it into explicit form. This leads to an approach of sharing knowledge that is technology-oriented as opposed to a people-oriented approach. On the other hand, subjectivists argue that knowledge cannot be separated from the knower. They argue that knowledge has some degree of tacitness and some degree of explicitness. The two components must exist together for knowledge to be complete. For instance, while it is possible to capture explicit component into documents, it must be tacitly understood and applied. According to [25], knowledge is based in sentient beings, or emanates from them, and thus it is always changing with human nature making it difficult to capture it into documents like information which is static. This is further supported by the argument that knowledge has both the process element (for knowing) and the artifacts resulting from the process [25]. While artifacts can be captured into documents, the process takes place in human brain and cannot be captured into documents. Subjectivists are therefore opposed to technology-oriented approach to knowledge sharing since it focuses only on sharing knowledge captured in documents and databases ignoring the process component, i.e. it is incomplete and flawed approach.

In this work, subjectivists' suggestion for a people-centered knowledge sharing model is adopted. This requires people-with-people connections for both tacit and explicit knowledge sharing. However, knowing the right person with the right knowledge can be

problematic in the sense that people are located in different parts of the world and in varying time zones among other limiting factors. I therefore propose the use of an intermediate step of people-with-documents connection. Documents are used by people to articulate their explicit component of knowledge. Understanding the explicit knowledge articulated in the documents without connecting directly to the knower can be a challenge. Documents are written in natural languages. Processing of natural languages suffer from polysemy (where a term has several meanings) and synonymy (where several terms have a common meaning) problems. A technique to address these challenges is vital. LSA which is a fully automatic mathematical/statistical technique for extracting and inferring relations of expected contextual usage of words in passages of discourse represents documents in one semantic space from which knowledge querying process can be performed. This technique has been shown to address challenges encountered in the processing of natural languages [21]. Further investigations into the LSA technique are therefore carried out in this work to realize people-with-documents connections.

1.2. Motivation

With globalization phenomena assuming prominence, business competition has reached zenith. This demands that companies cope by either adopting a low cost strategy or a differentiated strategy. Low cost strategy which involves offering products at a price lower than that offered by competitors for similar products has resulted in unnecessary price wars and it's no longer sustainable. On the other hand, a differentiated strategy which offers products at a quality higher than that offered by competitors requires a continuous process of innovation. The process of innovation heavily relies on the use of knowledge to generate, evaluate, and develop new ideas. In [4], it is argued that the study of innovation ought to focus mainly on knowledge. The process of innovation requires an interaction between knowledge previously captured in documents and databases, innovators' brains and other involved persons' brains. Connecting people to the relevant sources of knowledge therefore generates synergy in the process of innovation which couldn't be achieved if people relied only on their individual knowledge. The created

synergy translates into increased velocity at which new processes, services, and products are innovated. The need for knowledge sharing in an innovation process can be illustrated as shown below:

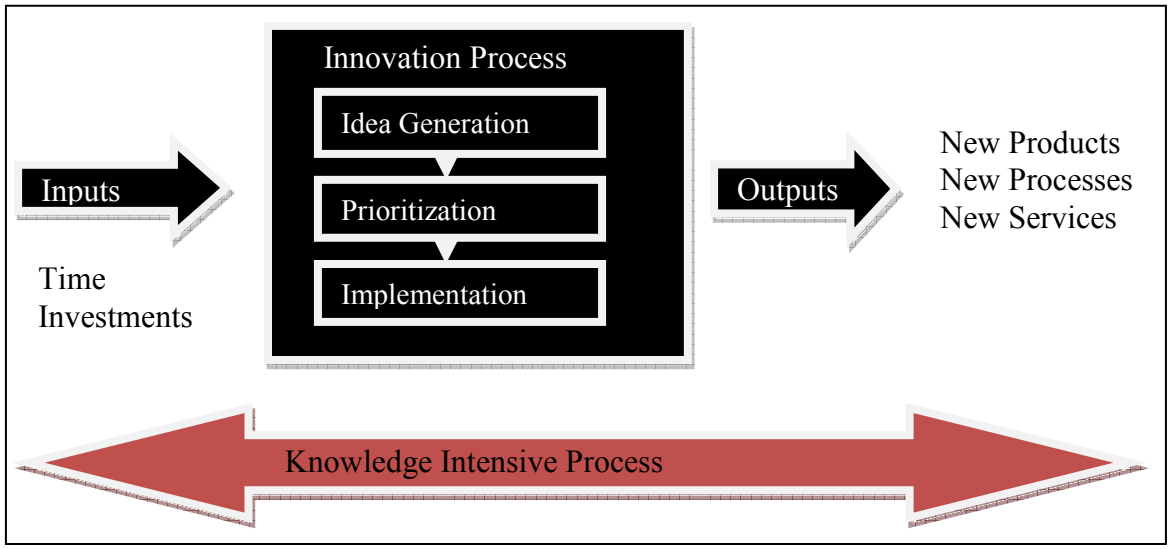


Figure 1.2: Knowledge requirement in the innovation process

As shown in the diagram above, inputs like time and investments are required to come up with a new innovation in a company. The process proceeds with generation of many ideas where the most important ones are prioritized for implementation to produce new products, processes or services. To combine all inputs towards generation of new ideas and to sustain the process until ideas materialize into outputs, knowledge remains the most important factor and as such should be shared among the involved stakeholders.

1.3. Objectives of the Work

The objectives of the work are to investigate the nature of knowledge and propose an appropriate model that support knowledge sharing, to investigate the application of LSA in realization of the developed model, and to investigate the inclusion of document annotations into the LSA process in order to improve knowledge representation and precision performance

1.4. Case Scenario

The case scenario used in this academic work mimics the challenge that was to be solved within Grindaix Company. Assume the following scenario of a large company AA with many engineers where knowledge has to be shared to realize synergy for innovations.

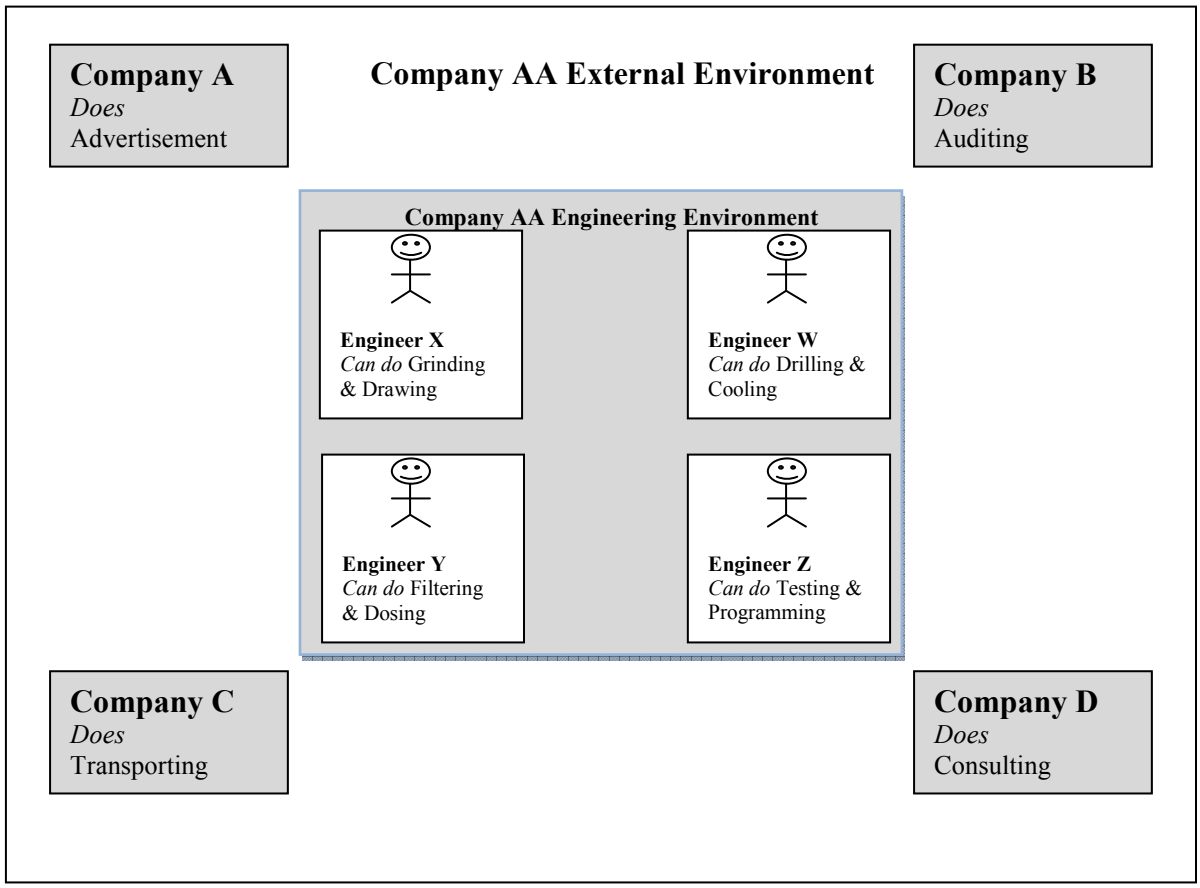


Figure 1.3: Case Scenario of a company's engineering environment

An engineer *X* wants to design a *grinding* process which should be as efficient as possible, e.g. *least energy utilization, grinding a workpiece in least possible time, etc*. To design this process various components or sub-processes have to be selected, designed, redesigned or reused in the process. For instance, use of either *centerless-grinding* or *grinding-between-centers* technology, a choice of a *cooling process* or *materials*, *cutting power* to implement, *specific material rates* to deploy, *tangential forces* to apply, etc. Alone, Engineer *X* is able to design the process by selecting the already existing

components. However, sharing knowledge with other engineers could lead to a more optimal solution. Even if the knowledge possessed by other engineers is totally different, knowledge sharing may unearth new potential or synergy effects e.g. Engineer *W* could provide his previously used visualization knowledge in his area of specialization that could enable optimization.

Based on this scenario, every engineer should be able to articulate what he/she knows to make it possible to be found by any other engineer to facilitate further sharing of tacit knowledge. Therefore, it should be possible to establish connections among people as shown in the diagram below if each person is to articulate what he/she knows to allow him/her to be contacted and that he/she knows what others are knowledgeable at to be able to contact them.

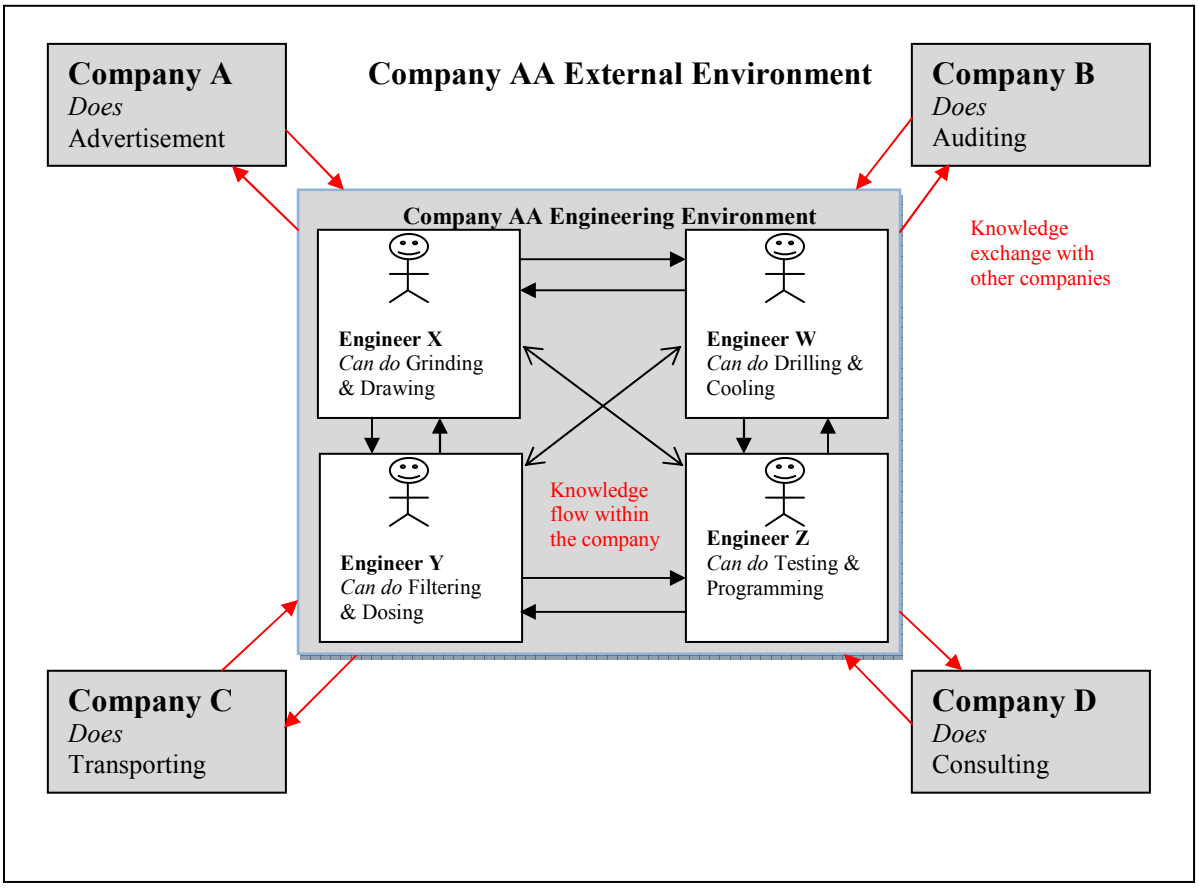


Figure 1.4*:* Expected knowledge flow within Company's Engineering Environment. The black arrows show the flow of knowledge within the company while the red arrows show the exchange of knowledge with other companies.

Based on the case scenario, knowledge sharing is expected to be done based on the following model.

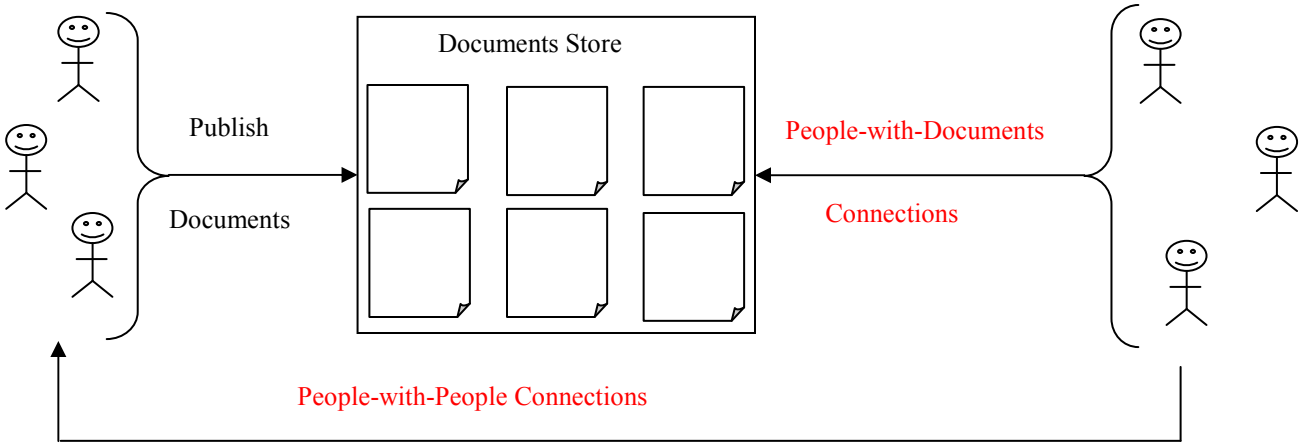


Figure 1.5: A people-connecting model for knowledge sharing

In order to share knowledge both within the company and its external environment, two possibilities could be investigated as shown in the diagram above.

1. People-with-people connection/personalization strategy: this provides direct sharing of tacit and explicit components of knowledge through direct people interactions. For instance, engineers *X*, *W*, *Y* and *Z* could directly interact with each other, hence sharing their individual knowledge. The advantage of people-with-people connection is that it allows both pull and push communication i.e. people can send (push) and request (pull) messages at the same time.

2. People-with-documents connection / codification strategy: this entails conversion or codifying knowledge from people into secondary sources (documents or databases) and providing others with accesses to these secondary sources. People-with-documents connection only supports the sharing of explicit knowledge which is basically push communication.

People-with-documents connection does not address full knowledge sharing. It can only enable the sharing of explicit knowledge component. To achieve full knowledge sharing people-with-people connection is required. The success of people-with-people connection is determined by the ability to determine the right people with the right knowledge. In the

current era of globalization, geographical variations and time zone differences among people make it difficult to have direct face-to-face interactions with the right people. However, published documents could help to determine the right people in addition to sharing explicit knowledge. This work therefore investigates how to achieve people-with-documents connection to share explicit knowledge component and as an intermediate step to people-with-people connections. This demands emphasis on articulation of explicit knowledge component into documents and connecting people to the right documents. For realization of people-with-documents connection, LSA technique is investigated in the next sections. This follows its superior recall performance compared to traditional term matching technique that has been used previously for documents retrieval.

1.5. Related Work

In this section, term-matching technique widely used to achieve people-with-documents connection is introduced. Additionally, the field of knowledge management which has been concerned with knowledge sharing is discussed. Limitations characterizing knowledge management are finally discussed.

1.5.1. Term-Matching Technique

Term-matching technique is based on Boolean logic and classical sets theory where documents and queries are treated as sets of terms [46]. Retrieval of a document is determined by the presence of query terms in a document. If a document contains the query terms, it is considered relevant with respect to the query and therefore included in the result set. Term-matching technique is currently applied in virtually all information retrieval systems [23]. The limitations encountered with term-matching technique are discussed in the next section.

1.5.1.1. Limitations of Term-Matching Technique

The following limitations characterize term-matching technique:

- Synonymy problem

  Term-matching technique does not offer retrieval of documents that do not contain query terms but rather contain other terms similar in meaning to query terms. For instance, a search for documents about *ailments* will miss documents with the term *diseases* which might still be relevant. Techniques that have been used to address synonymy challenge like thesaurus may present further challenges. For instance, added words may have different meanings than the intended meaning causing further deviations.

- Polysemy problem

  If the user query contains a term that has several meanings e.g. a *router (used to mean a wood shaper or in the context of internet connectivity)*, term-matching technique does not support discrimination against the several meanings leading to the retrieval of all documents with the query term in which case many documents may be irrelevant. Suggested solutions to address polysemy by use of controlled vocabularies and human intermediaries acting as translators are extremely expensive and less effective.

- Bag of words model

  Each term in a document is treated independently of any other term thus matching of two terms that always occur together e.g. *operating system* is counted as heavily as matching two that are rarely found within the same document [10]. This fails to take redundancy into account which may result into distortion of the results to unknown degree.

1.5.2. Knowledge Management

Knowledge management (KM) is defined as a systematic, holistic approach to the sustainable improvement of the handling of knowledge on all levels of an organization [11]. The aim of KM is to exploit an organization's intellectual assets for greater productivity, new value, and increased competitiveness [13]. To effectively achieve this, KM encompasses various activities some of which include the identification, acquisition, preservation, dissemination and sharing of knowledge. KM has been fueled by the increasing product complexity, globalization, virtual organizations and customer orientation which demand more thorough and systematic management of knowledge within an enterprise and between several cooperating enterprises [42].

KM remains limited in achieving knowledge sharing due to inadequacies in understanding the nature of knowledge. According to [47], KM has been centered more on technology rather than people. Based on objectivists from whom KM has been practiced, knowledge is viewed as two distinct types, tacit and explicit. KM practice concerns itself in transforming tacit knowledge from people into explicit knowledge and then using computer based information systems (CBIS) to distribute it. This approach to KM appears flawed. Firstly, the dynamic nature of knowledge is not reflected in the use of static knowledge management models or static CBIS. Knowledge consists of a dynamic process and artifacts and what can actually be shared, stored or captured are only the knowledge artifacts. Therefore, a model to share or manage knowledge ought to be dynamic to reflect this dynamic nature of knowledge that creates and updates the knowledge artifacts. Secondly, reliance on current information systems creates a misleading notion. Many organizations confuse knowledge with data and information. The capturing, storage and retrieval of data and information using the current information technology tools like databases has been considered as KM in many organizations. This is a quite misleading notion. In the section below, I discuss the limitations of CBIS that have been named knowledge management.

1.5.2.1. Limitations of CBIS

As mentioned above, KM has been done from an information processing perspective [21] based on objectivists' school of thought. This approach has resulted into a suite of CBIS that have been named knowledge management. Unless a CBIS is able to adapt itself to learn new knowledge, update its existing knowledge and make decisions as human beings would do, this view of KM remains greatly flawed. Arguments presented below show why an information processing view of KM cannot address knowledge needs.

- CBIS cannot deliver the right knowledge to the right person at the right time. We live in a world characterized by constant changes. Given this environment, a CBIS cannot with certainty predict the right knowledge for a given situation or even the right person to react to a given situation or to whom certain knowledge should be directed to. The solution would be to have CBIS that adapt themselves to acquire new knowledge, a dream that is far from reality.

- CBIS cannot store human intelligence and experience. Current technologies like databases can only store static information. Human brains are capable of storing knowledge and initiating the processes that continuously update this knowledge while generating new knowledge from the existing one or learning processes. The complex representation of knowledge that still remains an open question in cognitive neuroscience and neural informatics is not yet possible with modern CBIS at this time. According to [5], it is difficult to automate the cognition process.

- CBIS cannot distribute human intelligence. While storing and searching for information can be achieved via modern technologies, the intelligence of people who created the information cannot be transmitted. Additionally, modern technologies do not guarantee that distributed information will be understood as was meant by the original creator due to natural language challenges.

Apart from these challenges many other negative doubts among researchers continue to emerge regarding the capacity of KM to deliver on its promises [48] [21].

1.6. Thesis Structure

This reports starts with an introductory chapter providing a motivation as to why knowledge sharing is vital. The nature of knowledge is then investigated in Chapter 2. The chapter further digs into knowledge representation in human long term memory and in computer science in an attempt to unearth the nature of knowledge. Chapter 3 introduces the principle of document annotations whose inclusion is proposed into the LSA process. Chapter 4 discusses the principles of LSA and its limitations that have hindered its wide commercial use. Despite its limitations, possible areas where LSA can be used have been given. In Chapter 5, LSA use in knowledge sharing is investigated. Low precision performance at high recall values motivates the proposal to include documents' annotations within LSA process, a solution that is further discussed in the same Chapter 5. Finally, Chapter 6 provides a conclusion and expectations for future work.

1.7. Summary

This chapter has introduced the term knowledge and presented briefly two schools of thoughts regarding its nature, objectivists who argue for distinction of knowledge into tacit and explicit component and subjectivists who refute this by the claim that knowledge is inseparable from the knower. Knowledge management has focused on codification of knowledge and use of computer based information system as supported by the objectivists' school of thought. The practice of knowledge management based on codification of knowledge has led to practitioners perceiving knowledge management as an improved version of technology management or information management [6]. This is a greatly flawed view given that technology and knowledge are totally different entities. I have therefore argued for knowledge sharing that focuses on subjectivists' school of thought. This requires people-with-people connection. However, given the differences in location and time among other factors, people-with-documents connection is suggested as an intermediate step.

2. Nature of Knowledge

Effective sharing of knowledge is determined by how best its nature is understood [14]. To establish whether knowledge is separable from the knower and whether it can exist as two distinct types, tacit and explicit, an investigation into its nature and representation in the context of human memory and how this relate with its representation in computer science is investigated briefly in this chapter. Knowledge representation and storage in the brain continues to be an open question in cognitive neuroscience and neural informatics. I shall therefore attempt to develop arguments based on models that have surfaced so far. I start by providing a model that will help in distinguishing the concept of data, information, knowledge, and wisdom. I finally investigate knowledge representation in the context of human brain and in computer science. In the conclusion, I provide personal opinions on whether knowledge is separable from the knower to develop an effective sharing approach.

2.2. DIKW Hierarchical Model

Various attempts have been made to define knowledge and closely related terms like data, information and wisdom. In this work [1], DIKW (Data, Information, Knowledge and Wisdom) hierarchical model is used to explain these concepts. DIKW hierarchical model has gained wide use in library science, management information systems and computer science. The model shows how the four elements relate in terms of usefulness and time orientation. The elements are explained in the next paragraphs.

Data is basically raw meaningless points in space and time. It simply exists and has no significance beyond its existence. When, for example, a person fills in a form giving their name, address, age, social security number, the inscriptions in the form becomes data [14].

Information refers to data that has been given meaning by way of relational connections. The meaning derived from relational connections can be useful but it's not a strong necessity. Information can be seen as an attempt to use data to answer questions like who, what, where, when or how many e.g. what is the average amount of rainfall that fell last year [14]?

Knowledge is the awareness of what one knows through study, reasoning, experience or association, or through various other types of learning [25]. While information need not be useful, knowledge should lead to actions. Additionally, unlike information, knowledge is dynamic. This is because knowledge is contained in people's brain where it changes with sensory stimulation, experiences, reasoning and learning. Other sources have distinguished knowledge from information by the addition of truths, beliefs, perspectives and concepts, judgments and expectations, methodologies, and know-how [5]. To avoid confusion, we differentiate between knowing process and knowledge artifacts. Knowledge can be seen as a dynamic process that continuously creates, deletes and modifies knowledge artifacts. The artifacts reside in human brain but some of these artifacts can be captured in documents or databases. It is difficult to provide a clear separation of artifacts from the process. A model to manage or share knowledge should therefore include the dynamic process for updating the artifacts.

Wisdom is an extrapolative and nondeterministic, non-probabilistic process that helps to discover new understanding or predictions about the future. It can be seen as oriented towards predicting knowledge that will cope with future environments. According to Philosopher Bertrand Russell, the pursuit of knowledge should be guided by wisdom in order to avoid it having negative consequences [34].

DIKW model is illustrated below:

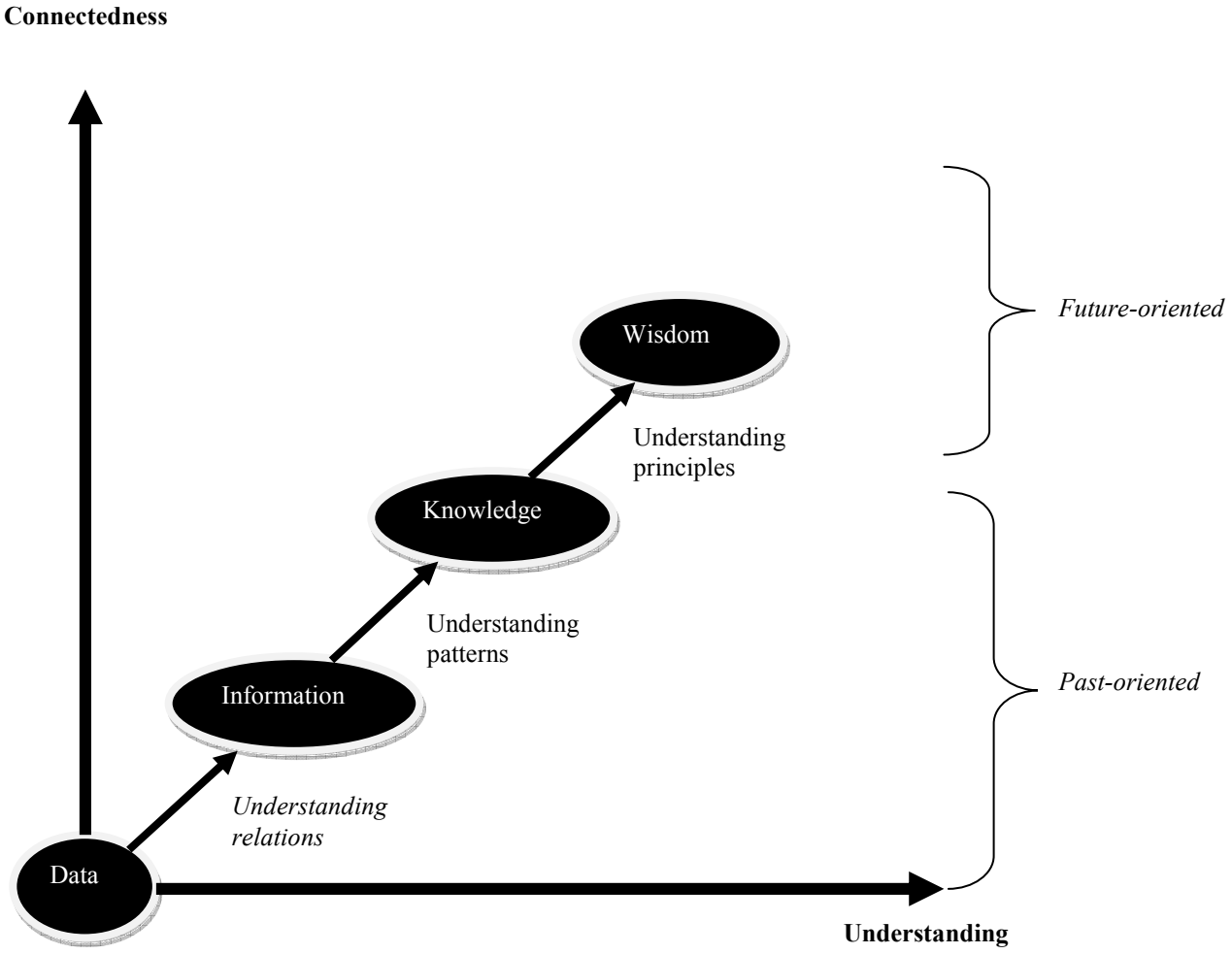


Figure 2.1: DIKW model

The elements of DIKW can be compared on how they relate to time. While data, information and knowledge relate to the past, wisdom incorporates vision and design to relate to the future.

The DIKW model discussed above has faced criticism. According to [14], the dated and unsatisfactory philosophical positions of operationalism and inductivism have been identified as the philosophical backdrop to the model. For instance, concepts that are not defined in terms of operations would be seen as meaningless.

2.3. Types of knowledge

Understanding types of knowledge helps in determining the best approach for knowledge management [36]. The big questions are however, is knowledge separable into distinct types and can knowledge be separated from the knower? For a long time, objectivists have classified knowledge into distinct tacit and explicit knowledge. *Tacit knowledge* can be considered as knowledge that is embedded in human brain but cannot be captured for storage in documents or databases. Tacit knowledge can however be transferred in form of learning by doing and learning by watching [27]. On the other hand, *explicit knowledge* refers to the knowledge which can be captured in documents and databases making it possible to communicate it with other people.

The classification of knowledge into entirely distinct types has faced criticism from subjectivists. The argument provided is that knowledge exists in both components with the degree of each component varying from one domain to another. In some cases, knowledge may be totally inexpressible (high degree of tacitness) whereas in other cases, it may be explicit among many people especially if they share a common background like training. The diagram shown below illustrates the levels in which knowledge can occur.

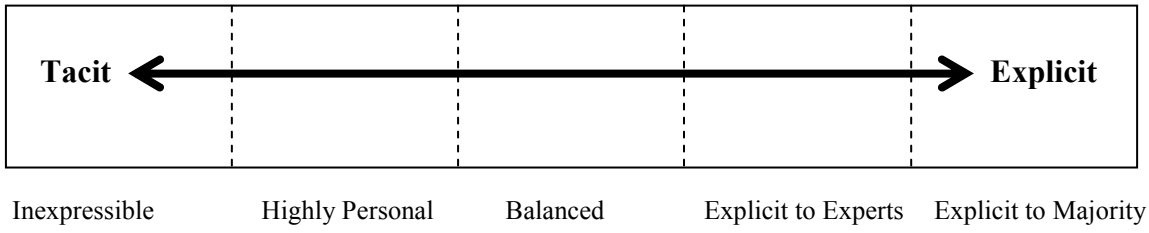


Figure 2.2: Varying dimensions of knowledge

2.4. Knowledge Representation

Following the differences in schools of thoughts (subjectivists and objectivists), it is unclear the best approach to use in order to achieve efficient knowledge sharing. Questions may then be asked as to how knowledge has been represented in human memory. Is it possible to duplicate knowledge as represented in human memory into knowledge management systems? In artificial intelligence, knowledge is mentioned in virtually all literature sources. How does artificial intelligence treat knowledge? To achieve a better understanding into the nature of knowledge, investigation into the nature of knowledge and its representation in the context of human long term memory and in computer science is done in the next section.

2.4.1. Human Memory

Human brain remains the most intriguing organ in a human being. In this section, I focus only on memory which is part of the brain. Memory is the foundation of all forms of natural intelligence and therefore investigation into it should provide a framework for analyzing knowledge representation. It is the memory that performs the tasks of encoding, storage and retrieval of knowledge.

For many years, human memory was classified into sensory-buffer memory, short-term memory and long-term memory [45]. Later, the classification was expanded to include action-buffer memory. In this work I only explore long-term memory (LTM) based on its role. Firstly, it provides permanent storage for knowledge. It has also been found [45] that all the 39 cognitive processes of the brain at the layers of sensation, memory, perception, action, meta-cognitive functions, and higher cognitive functions interact with LTM. Additionally, it provides capabilities to relate past knowledge in current situations to provide new reasoning which updates already held knowledge.

2.4.1.1. Knowledge Representation in Long-Term Memory

Knowledge representation in long-term memory (LTM) remains an open question in cognitive neuroscience and neural informatics [35]. It is not exactly known how human memory stores sounds, sights, smells, emotions, procedures and abstract ideas. However, over the years researchers have come up with general models to explain it. Three models presented here are taxonomic, thematic models and object attribute model.

2.4.1.1.1. Taxonomic Models

According to this model of knowledge representation, items are categorized based on the level of similarities, e.g. a category of vehicles with members such as car, train, bus, etc. Categories are then grouped hierarchically based on shared characteristics. Taxonomic models have been used to unearth the nature of the brain. This model is used to explain whether domain specific knowledge is associated with domain-specific localization in the brain. It does not provide explanation to other functions necessary for reasoning.

2.4.1.1.2. Thematic Models

Using these models, items not sharing similar characteristics are categorized based on their abilities to relate externally in some form of external relations. Examples include: a car & garage, a hospital & a doctor etc. Although this modeling of knowledge has been suggested, a clear illustration of how this is achieved in the long term memory has not surfaced.

2.4.1.1.3. Object Attribute Relation (OAR) Model

Here the OAR model [45] that was developed to formally represent the structures of internal information and knowledge acquired in the brain is introduced. In this model, a relational metaphor is used to describe memory. Based on the metaphor, knowledge is represented by the connections between neurons in the brain, rather than neurons themselves as information containers. The model represents all knowledge items in the long term memory in terms of objects, attributes and relations. The model is summarized as shown below:

$$OAR \triangleq (O, A, R) \quad (2.1)$$

Where: O is an object, A is an attribute and R is a relation.

Objects (O) refers to a set of abstractions for external entities and/or internal concepts. Examples of objects include trees, animals, buildings, etc. This set is represented as:

$$O = \{o_1, o_2, \ldots, o_i, \ldots, o_n\} \quad (2.2)$$

Attributes (A) refers to a set of sub-objects that are used to denote detailed properties and characteristics of the given object. For each given objects $o_i \in O$, $1 \leq i \leq n$, A_i is a finite set of attributes for characterizing the object in the following manner:

$$A_i = \{A_{i1}, A_{i2}, \ldots, A_{ij}, \ldots, A_{im}\}, \quad (2.3)$$

Where each $o_i \in O$ or $A_{ij} \in A_i$, $1 \leq i \leq n$, $1 \leq j \leq m$, is physiologically implemented by a neuron in the brain. Attributes used maybe generic and/or specific attributes. Examples of attributes may include: chemical attributes, specifications, economic attributes, space-related attributes, cognitive attributes, measurements, etc.

Relations(R) refers to a set of connections or inter-relationships between an object and another object, an object and an attribute or between two attributes.

The resulting model of knowledge as represented by OAR model is illustrated in the diagram below using two objects represented by O_1 and O_2, sets of relations for the two objects represented by $r(O_i, O_{j,k})$ and a set of attributes represented by A_{ij}.

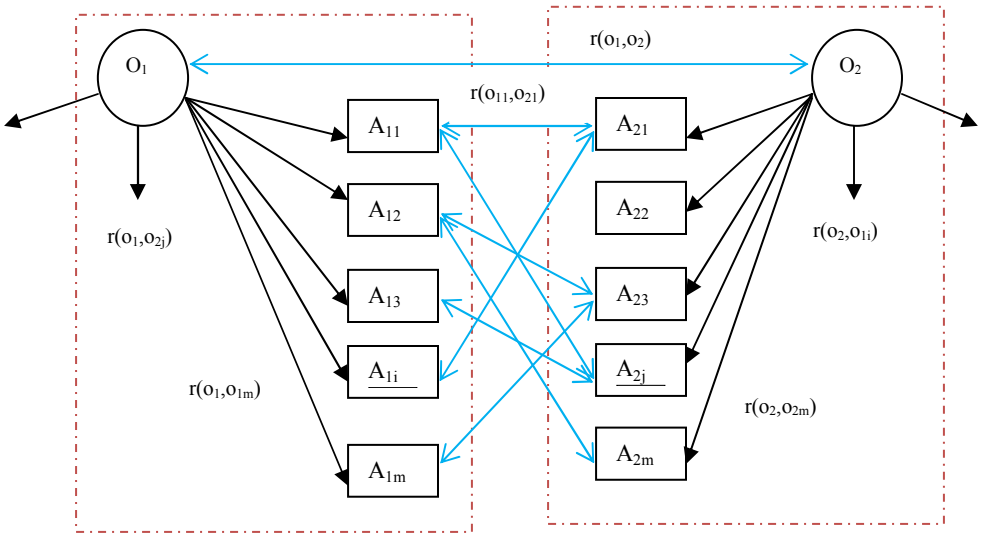


Figure 2.3: The OAR model of Memory Architecture

2.4.2. Knowledge Representation in Computer Science

Models shown above for knowledge representation in human memory do not provide us with a way to clearly define the nature of knowledge. What can be deduced is only the interrelatedness of concepts in human memory but the updating process of these concepts or even the nature of interrelatedness cannot be deduced. With knowledge representation not clearly known in human brain, how can we achieve artificial intelligence? Artificial intelligence aims at making machines intelligent to solve problems like human beings do but to achieve this, knowledge has to be transferred from human memory.

Knowledge representation in the world of computer science seeks to come up with formalism for knowledge about domains intended for processing by modern computers. The formalism should support expressivity as well as reasoning about the domain knowledge. Knowledge representation in computer science should be identical with its representation in human mind. However, this has not been achieved. As a result,

simulation efforts to mimic intelligence within machines proceed alongside efforts to understand human memory. Efforts to simulate intelligence have resulted into three basic models of knowledge representation in computer science. These models are either based on semantic networks, frames or logic calculus. These concepts are briefly discussed below.

2.4.2.1. Semantic Networks

Semantic networks consist of a set of interconnected nodes and arcs that can be used to represent knowledge or reason about knowledge. Nodes are used to represent concepts whereas arcs represent relations. It is closely related to the OAR model suggested to explain knowledge representation in human memory. There are many types of semantic networks which include [40]: definitional networks, assertional networks, implicational networks, executable networks, learning networks and hybrid networks. These semantic networks differ in varying degrees in terms of structure and the purpose they serve. In this case an example of implication network is shown. This kind of semantic network derive its name from the fact that it uses implication as the primary relation.

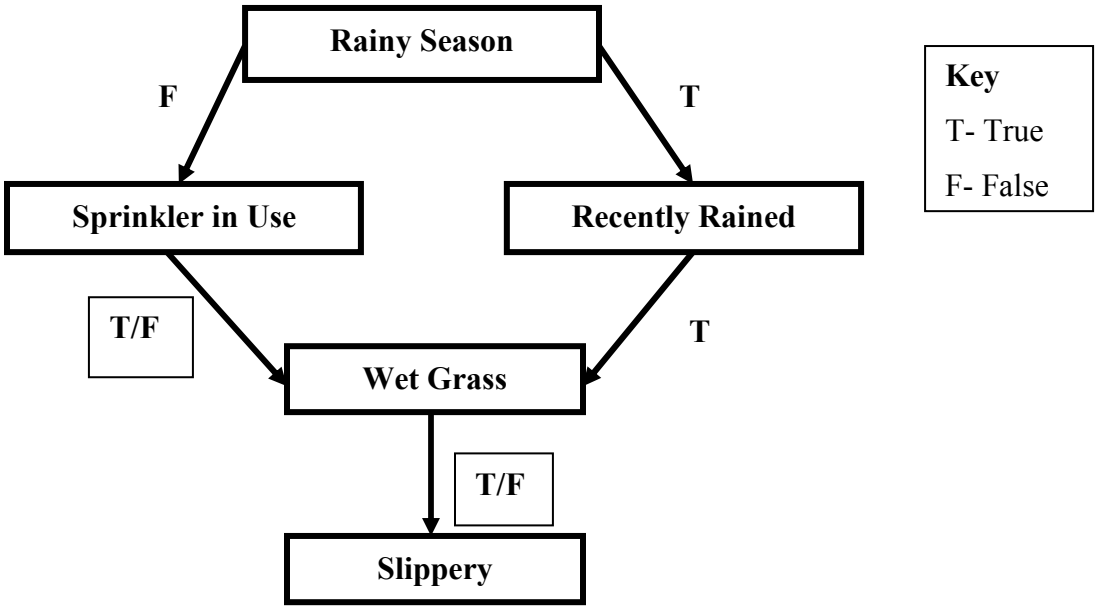


Figure 2.4: An implication network for reasoning about wet grass [40].

The above implication network shows possible causes for slippery grass. In each box a proposition is shown and each arrow shows the implication from one proposition to another. Starting with the proposition 'Rainy Season' the arrow marked T (=True) implies that it had rained recently otherwise the arrow marked F (=False) implies that the sprinkler must be used for the same effect to be achieved (wet grass).In the case of only one outgoing arrow from a box, the truth of the first proposition implies the truth of the second proposition, but falsity of the first makes no prediction about the second.

Reasoning in semantic networks can be done using different approaches. In the above semantic network, logics and probabilities can be used [40]. Logical inferences can be used in forward and backward directions from a particular node making it possible to infer new knowledge, check for consistencies, and search for contradictions or find locations where the expected implications do not hold. Probabilistic reasoning would entail adapting the forward and backward logic reasoning to probabilistic interpretation since truth can be considered a probability of 1.0 and falsity as 0.0.

2.4.3. Frames

A frame is a data structure invented by Marvin Minsky for knowledge representation [26]. Minsky thought that in human memory, knowledge is stored as different chunks, which motivated his idea for a frame. A frame is a data structure that contains a collection of attributes (slots), potentially having types (or value restrictions) and potentially filled initially with values [24]. The slots contain relationships to other frames or methods for recognizing patterns instances.

The structure of the frame hierarchy is based on the production-frame knowledge representation [39]. This exploits classification hierarchy with inheritance relation and active slots with query procedures and daemons, around which productions rules are clustered. The advantage of this structure is that it allows for the combination in one model for static knowledge about a problem in the form of slot values, structural

knowledge of the problem domain in the frame hierarchy and the dynamic knowledge in the form of attached procedures that drive logical inference [24]. Based on the case scenario, knowledge about an Engineer X can be represented using a frame as shown below:

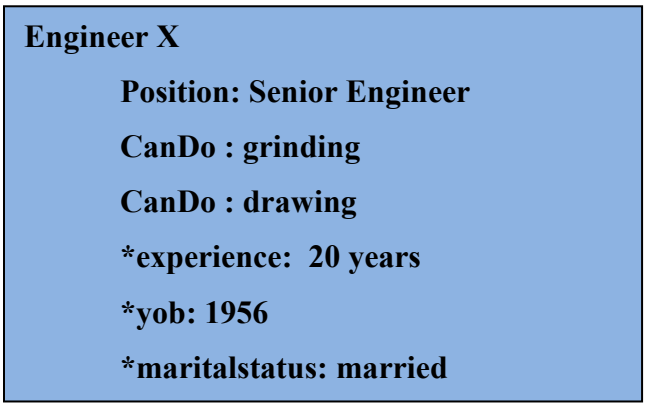


Figure 2.5: A frame representing knowledge about Engineer X. Asterisks are used to denote typical properties such as experience, year of birth and marital status.

In the example above, knowledge about Engineer X is described using slots, *Position* and *CanDo* that link to other frames namely *Senior Engineer*, *grinding* and *drawing*. In terms of reasoning, frames do not offer guarantees that everything that could be deduced from a given set of information may be deduced [24]. In addition to this reasoning limitation, there are still many things that cannot be represented using frames e.g. negation, disjunction and some types of quantification.

2.4.4. First-Order Predicate Calculus

First-order predicate calculus is a formal logic that permits the description of relations and the use of variables. Its inclusion of quantifiers makes it different from propositional calculus. First-order predicate calculus has proved its worth in the field of knowledge representation. To show the application of first-order predicate calculus, I briefly introduce description logics which is based on first-order predicate calculus and used for knowledge representation.

2.4.4.1. Description Logics

The design of knowledge systems involves two aspects [15]. The first aspect is about providing a precise characterization of a knowledge base. This involves precisely characterizing the type of knowledge to be specified to the system as well as clearly defining the reasoning services the system need to provide. The second aspect is about providing a rich development environment where a user can benefit from different services that can make his/her interaction with the system effective. Description logics (DL) serve the first aspect. DL are a family of languages developed as a formalism to achieve knowledge representation and reasoning support for knowledge which makes it possible to develop usable knowledge bases. They are based on first order predicate calculus that helps handle semantic ambiguities unlike the semantic networks and frame systems that are non-logic based systems. Use of first order predicate logic presents efficient reasoning (the ability to infer new implicit knowledge from explicitly represented knowledge) support not comparable with semantic networks and frame systems. Later, web ontology language (OWL) based on DL is introduced.

DL languages employ syntactic rules and semantic rules for knowledge representation. Using syntactic rules, concepts (unary predicates) about a domain, roles (binary predicates) and individuals (constants) can be described. For the semantics of DL, concepts are normally given set-theoretic interpretation. This means that concepts are interpreted as sets of individuals and roles as sets of pairs of individuals.

2.4.4.1.1. Basic Constructs for Concept Expressions

DL languages in general use atomic roles and concepts to represent knowledge. From these, complex descriptions can be built inductively by use of concept constructors. Various variants of DL languages exist with different syntactic constructs. In this case, the syntax of Attributive Language ($\mathcal{AL}$-Language) used for concept descriptions is shown as an example [15].

The table shown below introduces the abstract notation of $\mathcal{AL}$-Language. Letter A represents an atomic concept, letter R the atomic roles and the letters C and D represent concept descriptions.

$C, D \rightarrow A \mid$	(Atomic Concept)
$\top \mid$	(Universal Concept)
$\bot \mid$	(Bottom Concept)
$\neg A \mid$	(Atomic Negation)
$C \sqcap D \mid$	(Intersection)
$\forall R.C \mid$	(Value Restriction)
$\exists R.\top \mid$	(Limited Existential Quantification).

Table 2.1: Syntax of $\mathcal{AL}$-Language

2.4.4.1.2. DL-Based Knowledge System

Description logics can be used to develop knowledge based system as shown in the diagram below. In such a system the tasks supported by description logics would include: knowledge representation in the knowledge base and the reasoning and manipulation support for the represented knowledge.

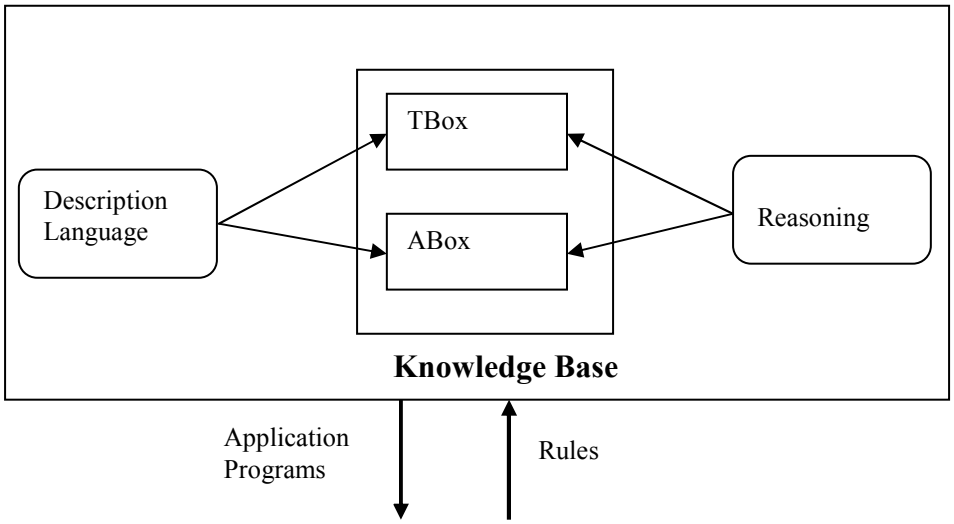


Figure 2.6: Architecture of a DL-based knowledge representation system [15].

The DL knowledge base consists of TBox and ABox:

TBox contains intensional knowledge otherwise referred to as *Taxonomy* or *Terminology* or *Vocabulary* in various contexts. These are basically declarations about the general properties of concepts and roles. To some extent, it is the knowledge that is thought to be constant over time. An example of a TBox is shown on the next page.

Tool ≡ Thing Π ∃ canDo.Action
Process ≡ Action Π ∃ use.Tool
Employee ≡ Person Π ∃ canDo.Process
Engineer ≡ Employee Π ∃ study.EngineeringCourse

Table 2.2: A TBox showing concepts from the Case Scenario

Reasoning tasks for a TBox: the following reasoning tasks can be applied on TBox.

- *Subsumption:* Given two concepts *C* and *D, subsumption* involves checking whether the *subsumer (D)* is more general than the *subsumee(C).* It is therefore checking whether the first concept always denotes a subset of the set denoted by the second one.

- *Satisfiability:* This is a problem that involves checking whether a concept expression does not necessarily denote the empty concept. Satisfiability can be considered as a special case of subsumption, with the subsumer being the empty concept, meaning that a concept is not satisfiable. Basically, all the reasoning tasks in a TBox can be reduced to satisfiability checks.

- *Equivalence:* Given two concepts *C* and *D, equivalence checking* entails finding whether in every model of the TBox *C* and *D* are equal.

- *Disjointness:* Two concepts *C* and *D* would be disjoint if their intersection *(C⊓D) is empty*

ABox contains basically the extensional knowledge otherwise known as assertional knowledge. It is basically the knowledge that is specific to the individuals of the domain of discourse (membership assertions). This knowledge is thought to be contingent or dependent on some circumstances and therefore subject to occasional or even constant change. An ABox can contain role or concept assertions e.g. *Engineer(X)* in an ABox would be a concept assertion whereas *CanDo (X, grinding)* would be a role assertion. *Instance checking* is performed in ABoxes to verify whether a given individual is an instance of a specified concept. The table below shows an ABox based on the TBox shown previously.

Tool (NeedleNozzle)
Engineer(X)
canDo(X, grinding)
canDo(X, *drawing*)

Table 2.3: An example of ABox

Reasoning Tasks in ABox: the following reasoning tasks can be applied to ABox.

- ABox *satisfiability:* This involves checking whether the assertions contained in an ABox are satisfied (no contradictions) with respect to TBox.

- ABox *realisation*: In this reasoning task, the most specific concept for each object in the ABox with respect to TBox assertions is computed.

- ABox *subsumption*: This task involves checking whether a particular object in ABox is a specific instance of a given concept in TBox

2.5. Aligning Knowledge Sharing Components

Models discussed above fail to clearly explain the concept of knowledge. Lack of clear understanding into the nature of knowledge has led to organizations spending huge amounts of money with little or no return. For instance, in 2003 the top 500 companies in the USA were projected to lose at least $31.5 billion annually due to poor knowledge management strategies [12]. This can be avoided by aligning the components involved as more work is done to reveal the concept of knowledge. Delivering the right knowledge to the right person at the right time involves proper alignment of the following components:

- People: How do people share experiences, skills and competencies in the organizations?
- Processes: What processes can be exploited to achieve knowledge sharing from one person to another?
- Structure: What adjustment in organizations' structures would support knowledge sharing?
- Technology: What necessary technologies would support the sharing of knowledge among people?
- Corporate culture: What ideal culture in the group would motivate people to share their knowledge?

One of the poor strategies used by companies is a focus on technology as a solution to their knowledge management needs. Technology is only an enabler but not the solution. In fact, strategies to acquire computer based information systems like ERP have led to overreliance on them, reducing people's creativity and reliance on their knowledge. In [33], it is mentioned that electronic dependence causes negative effects on innovation velocity. The most important message for companies willing to implement knowledge sharing is to focus on people. It has been shown that the vast majority of people possess: more creativity, more resourcefulness, more initiative, more talent, and more intelligence than their jobs can offer [12]. Therefore, companies should focus on empowering their people to increase socialization or social capital through such strategies like:

- Developing a culture of trust that allows free knowledge sharing.

- Developing communities of practice: These are groups of people who share a concern, a set of problems, or a passion about a topic, and who deepen their knowledge and expertise in this area by interacting on an ongoing basis [8]. The communities help people acquire knowledge from each other making it possible for them to continue functioning even after the exit of knowledgeable employees. It has even been argued that the conversation and knowledge exchange that is needed to combine knowledge for innovation takes place mainly at group level [33]
- Maximizing individual payoff's for sharing knowledge. This can be addressed through necessary incentives.

2.6. Summary

This chapter reveals that more research into the nature of knowledge and its representation in human memory needs to be done. Models suggested so far to explain knowledge representation in human long term memory are not fully expressive. From this limitation, representation of knowledge in computer science is done through the simulation of the human knowledge as opposed to its duplication. Despite the unclear nature of knowledge, some conclusions can be made about it. Models presented suggest that knowledge has a process part unlike information. The process addresses tasks such as reasoning and creation of new knowledge artifacts from the existing ones. Therefore it can be concluded that knowledge has 2 parts: process and artifacts. This view of knowledge makes it impossible to entirely separate knowledge from the knower. The process is in the mind and cannot be captured into documents. To effectively share knowledge, companies should pursue people-with-people connections. Documents can help in determination of the people to connect with. This is the reason why LSA role in achieving people-with-documents connections is investigated in the next chapters. Additionally, an investigation into the inclusion of documents annotations in the LSA process to improve knowledge representation and precision performance is done.

3. Documents Annotation

In general terms, annotation is the practice of adding or attaching metadata to documents. Metadata may be defined as a record that consists of structured information about a resource or simply data about data. Metadata is used in many areas such as library sciences, semantic web, database systems, and file systems among others to give extra information about given data. In this chapter, I investigate annotation of documents in line with the objective of this work to achieve people-with-documents connection.

3.1. Annotation Process

The basic objective of annotation is to provide for additional set of information that was not specified by the initial author of the document [31]. Including annotations to the LSA can be compared to the reasoning process that takes place in human brain. Through the reasoning process, human mind is able to understand the available information it is presented with. Annotations on the other hand can help to provide higher-level interpretation of given information or resources. Annotation process can either be done manually, semi-automatically or automatically. Manual annotation is done by a user based on some ontology or on the basis of a given vocabulary. Semi-automatic process involves use of a tool and user's participation still based on a given ontology or vocabulary while automatic process involves use of only an annotating tool. At the moment, manual annotation is the most common despite the fact that it is expensive and error prone. More research geared towards automating the process is still under progress. On the next page, the annotation process is illustrated:

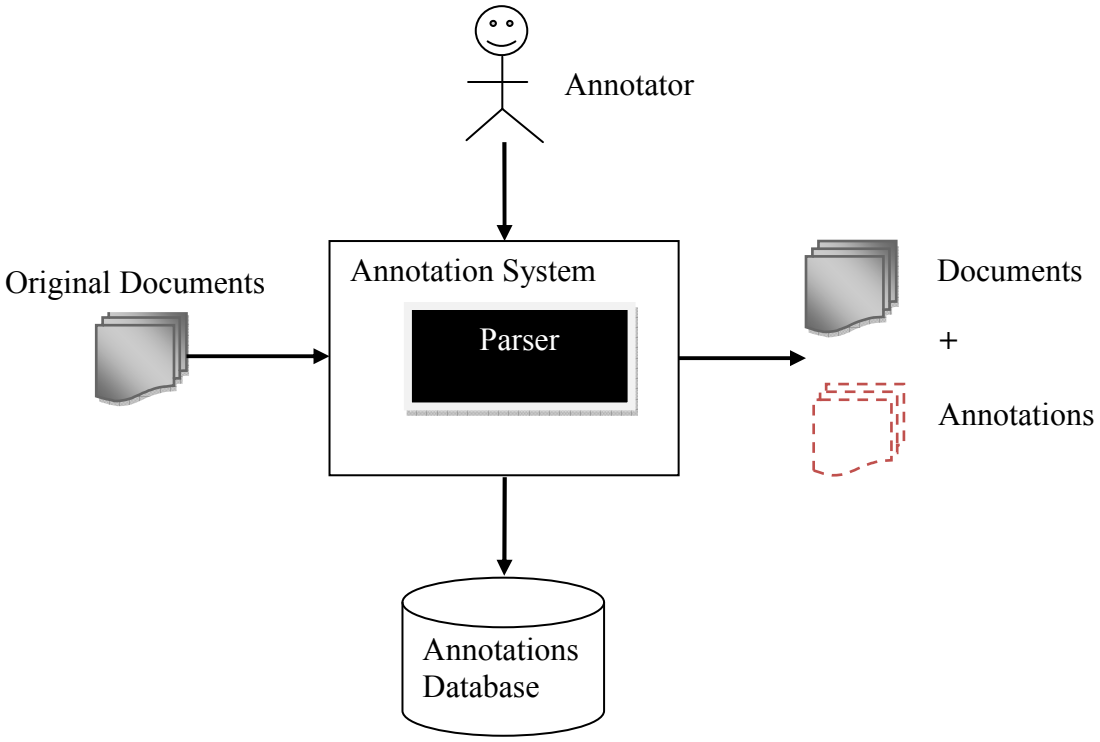


Figure 3.1: The annotation process

Annotations can be expressed using different languages (some natural and others formal languages), can be stored in different ways (e.g. within the original documents, in databases, in separate documents, etc.) and can be used for different purposes. According to the diagram on the previous page, the annotator adds annotations to documents as they are parsed. Produced annotations can then be stored in a database and/or as separate documents.

3.2. Classification of Metadata

The annotation process can result into three different kinds of metadata [2]:

1. Standard metadata: They are formal specifications used to annotate materials of any kind. Examples include Dublin Core widely used to describe digital materials such as video, sound, image, text and composite media like web pages [46].
2. Semantic metadata: This is metadata that provides semantic descriptions based on ontologies about a domain.
3. Attention metadata: This deals with collected detailed information about the relation between users and the content they access.

The current trend is towards usage of semantic metadata due to its advantages that include [3]:

1. It is machine processable: use of ontologies to generate semantic metadata can allow having well-formed metadata that machines can read and understand.
2. Flexibility and extensibility: this can be done by extending metadata with other metadata or by mixing different ontologies.
3. Reasoning: reasoning rules can be formulated as the metadata is expressed formally. New relations can also be derived, thus exploiting the use of semantic search.
4. Interoperability: even though standard metadata promotes interoperability, semantic metadata has the added value of supporting partially agreed ontologies that will enable systems to interoperate much more easily.

Based on the advantages mentioned above, this work proceeds on the basis of semantic metadata. This is done using ontologies. Before discussion of ontology-based documents annotation, the concept of ontology is briefly introduced.

3.3. Ontology

An ontology is a [13] formal, explicit specification of a shared conceptualization. It defines a common vocabulary for people willing to share knowledge in any domain. In semantic web, they perform the role of providing machine-processable semantics of data and information sources that can be communicated between different agents (software and people) [22]. Various formal languages to encode ontologies have been developed. Some of these languages include Common Algebraic Specification Language, Common Logic, DOGMA, OIL, OWL, OBO and KIF among others [46]. OWL (Web Ontology Language) is the main language for the definition of ontologies. OWL is discussed later.

An ontology basically consists of the following:

- *Classes*: These are concepts about the domain being represented. In grinding industry, examples of classes would include: *coolants*, *grinding wheels*, *cooling nozzles*, etc.

- *Properties:* These are attributes of the classes. They include data type properties and object properties. Data type properties are attributes that describe a class itself, e.g. the *speed* of a class *grinding wheel*. Object properties relate a class with another class or instance of another class. These properties are shown later in this report.

- *Individuals*: These are instances of the classes e.g. *oil* is an instance of the class *coolant* in grinding domain.

3.4. Ontology Based Document Annotation

Annotation on documents can be done on the basis of controlled vocabularies or thesaurus where annotations are normally pointers to terms in the vocabulary. Examples of vocabularies include MeSH for medical subject headings (http://www.nlm.nih.gov/mesh/meshhome.html), TGN for Geographical names (http://www.getty.edu/research/tools/vocabulary/tgn/index.html), etc. However, these controlled vocabularies are limited in semantic expressions. A new approach is to use ontologies as vocabularies for semantic metadata annotation [9]. Why use ontologies to annotate documents? Ontologies would be useful in the following ways [28]:

- To facilitate a common understanding of the structure of information among people
- To enable reuse of domain knowledge
- To make domain assumptions explicit
- To analyze domain knowledge
- To separate domain knowledge from operational knowledge.

Ontology-based document annotation involves the use of ontologies about a domain to include semantics into documents in order to have semantically enriched documents. The annotation process proceeds by creating an association or links between sections of the document and the concepts represented in the ontology. This is elaborated in later sections.

Ontology-based document annotation is made up of three types of information that include concept instances, attribute values and relation instances [9]. A concept instance is used to relate some document information to a certain concept in the ontology. An attribute value relates a concept instance to a part of a document where the part of the document is the value of one of the instance attributes. A relation instance relates two concept instances by some domain specific relation.

3.5. Representation and Storage of Annotations

The semantic annotations can be stored in the following formats:

- *Representation within the original documents*: annotations could be attached directly to documents and stored within those documents by use of special styles like different color, use of tags, fonts, signs, and images that do not form part of the original document
- *Use of formal languages*: two formal languages, resource definition framework (RDF) and web ontology language (OWL) developed for Semantic Web could be used. These two languages are briefly introduced below:

3.5.1. Resource Definition Framework (RDF)

RDF is a syntactical convention and a simple data model for representing machine-processable semantics of data especially for Web resources [13]. RDF can be compared to class diagrams or the entity-relationship diagrams in software engineering. RDF is based on describing resources in the form of subject-predicate-object expressions. The combination of an instance of subject, predicate and object forms a single RDF triple or statement. *Subject* is the resource to be described. It can be a Web page, part of a Web page or an actual entity not accessible via the Web e.g. a printed book. *Predicate* refers to a specific aspect, characteristic, attribute or relation used to describe a resource. It is represented using a URI. *Object* refers to the value of a particular predicate that describes a resource. The object can be defined as a URI, blank node or a Unicode string literal.

Figure 3.2: The building blocks of an RDF statement

The collection of RDF files linked together will intrinsically represent a labeled, directed multi-graph or a semantic network without inheritance. RDF statements can be serialized using different formats. The mostly used formats are XML and Notation 3. In the example below, XML serialization is used. The example below demonstrates the use of RDF in describing a resource, *Engineer X* from our case scenario:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:contact=" http://www.grindaix.de/contact">
   <contact:Person    rdf:about="http://www.grindaix.de/People/Engineer X">
            <contact:fullName>Engineer X </contact:fullName>
            <contact:mailbox rdf:resource="mailto:ex@grindaix.de"/>
            <contact:personalTitle>Eng. </contact:personalTitle>
   </contact:Person>
</rdf:RDF>
```

Table 3.1: An example of an RDF file

RDF Schema (RDFS) is used to define the structure of RDF. RDFS defines additional modeling primitives on top of RDF like the definition of classes (concepts), inheritance hierarchies for classes and properties, and domain and range restrictions for properties. As a matter of summarization, the following 3 statements about RDF documents and RDF Schema documents can be stated:

- At the syntactic level, they can be serialized as XML files
- At the structure level they consist of a set of triples as explained above
- At the semantic level they constitute one or more graphs with partially predefined semantics.

3.5.1.1. Expressive Power of RDF & RDF Schema

In description of resources a language should offer high expressive power and efficient reasoning support. However, reasoning support and expressive power pull in opposite directions demanding a balance to be struck. This is because the richer the language is, the more inefficient the reasoning support becomes. RDF and RDF Schema have some very powerful modeling primitives e.g. rdfs:Class (the class of all classes) and rdf:Property (the class of all properties) which are very expressive in representation of some ontological knowledge. However, they still have some limitations in terms of their expressive nature [19]:

- Local scope of properties: In RDF Schema, range restrictions that apply to some classes only cannot be declared e.g. *rdfs:range* defines the range of a property which must apply for all classes. For example, it is not possible using RDF Schema to state that a goat *eats* only grass while other animals might also *eat* meat.
- Disjointness of classes: RDF Schema allows only the stating of sub-class relationships but does not offer a means to state disjointness of classes e.g. you can state that a *male* is a sub-class of *person* but you cannot state that *female* and *male* are disjoint.
- Cardinality restrictions: Restrictions on how many distinct values a property may or must take are impossible to express in RDF Schema e.g. you cannot state that a person has only 2 parents or a course is taught by at least one lecturer.
- Special characteristics of properties: There are other special characteristics about properties that cannot be expressed using RDF Schema e.g. to say a property is transitive (like “greater than”), unique (like “is mother of”), or the inverse of another property (like “eats” and “is eaten by”).

3.5.2. Web Ontology Language (OWL)

The motivation behind the development of OWL follows the limitations in RDF and RDF Schema whose expressive powers are limited as noted above. The definition of OWL is in line with the requirements for an ontology language which include: a well-defined syntax, a well-defined semantics, efficient reasoning support, sufficient expressive power and convenience of expression. However, these requirements cannot be fulfilled in one language, for instance it is difficult to have a language that is both highly expressive as well as with efficient reasoning support. This has led to three species of OWL namely OWL Full, OWL DL and OWL Lite. More details about these species can be found in [19]

3.6. Summary

In line with the objective of this work, to investigate the inclusion of annotations into the LSA technique, this chapter has provided basics of the annotation process. Semantic annotation based on ontologies has been argued for compared to other annotation schemes. Inclusion of annotations within the original documents and the use of formal languages have been mentioned as the two ways that could be exploited for representing annotations. In Chapter 5, the investigation of these annotations into the LSA technique is further investigated. But before this, LSA is discussed in the next chapter.

4. Latent Semantic Analysis

This chapter discusses the principles of LSA, possible areas of applications and limitations characterizing the technique.

4.1. Overview

Latent semantic analysis (LSA) is a theory and a method for analyzing the global relationships between compositions and the units they contain. This happens by use of mathematical or statistical techniques that eliminate unnecessary noise to reveal the deep semantic structures from compositions and units contained in them. For instance, in the case of information retrieval, LSA maps documents and the terms they contain in one semantic space from which querying can be done. In the field of information retrieval, LSA is commonly referred to as Latent Semantic Indexing (LSI).

Unearthing of semantic structures by LSA provides new opportunities for application of processes such as querying, clustering and classification among others to compositions represented. In the context of information retrieval, LSA is used to improve the performance of retrieval by helping to overcome the fundamental natural language challenges (polysemy and synonymy). It enables retrieval on the basis of conceptual content instead of merely matching words between queries and documents [6]. By use of dimensionality reduction which makes complex natural language problems tractable and the intrinsically global outlook of the approach which tends to complement the local optimization performed by more conventional techniques, LSA appears to be a much better approach for information retrieval. It also seems particularly attractive due to the mapping of discrete entities onto a continuous parameter space where efficient machine learning algorithms can be applied. The illustration on the next page shows how LSA can reveal documents' semantics in the context of information retrieval thus enabling query answering process.

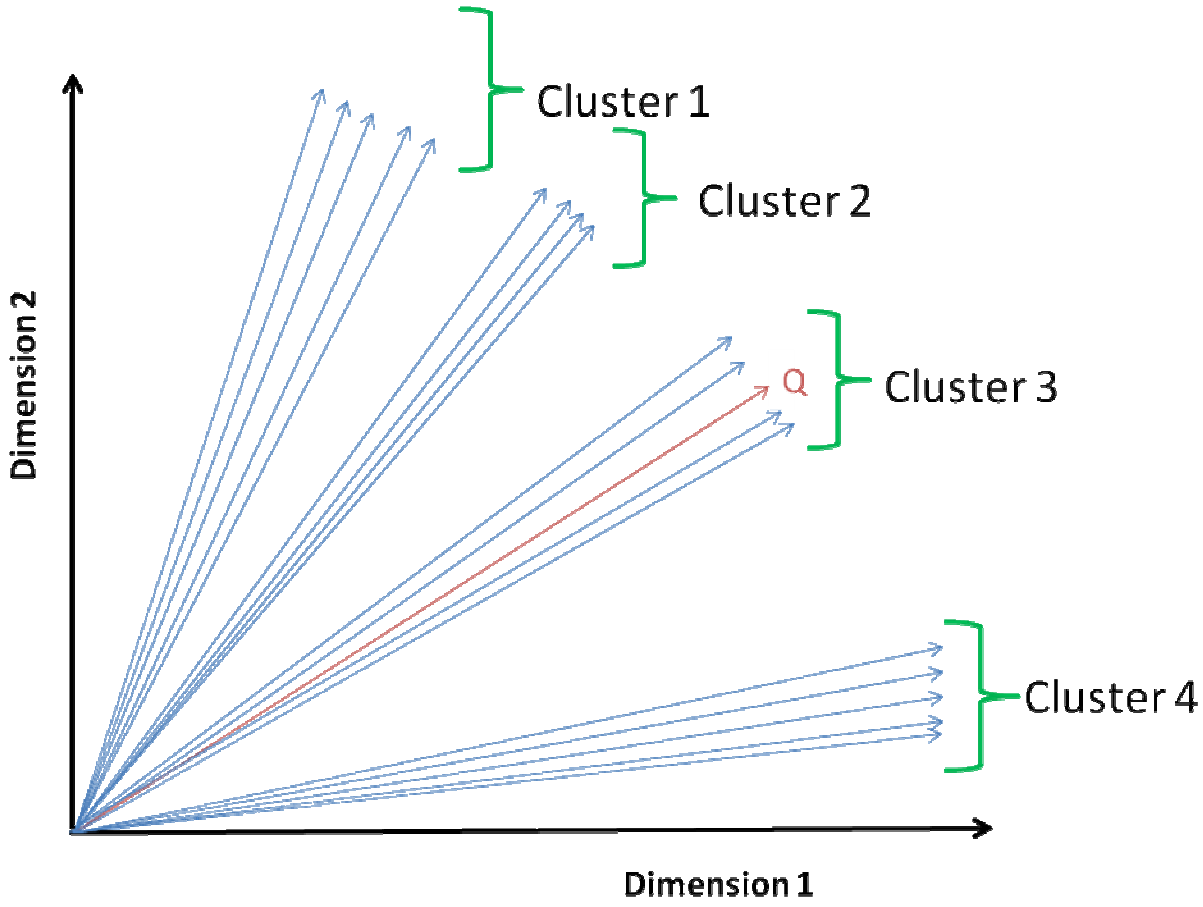


Figure 4.1: Illustration of LSA semantic space

From the illustration shown above, LSA is applied on a collection of 18 documents using two dimensions. This process results into a common semantic space with 4 clusters (where a cluster refers to documents with similar semantics). A query Q is also mapped on the same semantic space. Using query Q, 4 documents in cluster 3 would be retrieved.

4.2. LSA Technical Details

LSA process can be summarized into 4 steps. These steps are briefly discussed below.

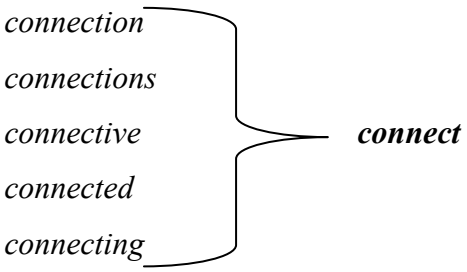
4.2.1. Pre-processing step

This step prepares the collection by eliminating unnecessary redundancies to have a dictionary (list of important units/keywords from the collection) that is used in vector space representation of the collection. Preprocessing step involves elimination of stop words, stemming and lemmatization. Elimination of stop words aims at the exclusion of terms that do not contribute to the documents' semantics. For instance, in a collection of documents written in English, the following terms would be eliminated.

a, an, and, are, as, at, be, by, for, from, has, he, in, is, it, its, of, on, that, the, to, was, were, there, how, much, when, will & with.

Table 4.1: Examples of stop words.

On the other hand, lemmatization and stemming eliminate the redundancies caused by structures of terms contained in documents [23]. For instance, in English language, we could have within the same document terms like: *connection, connections, connective, connected* and *connecting* which would create redundancies if all of them were to be stored. An example of preprocessing is shown below.



One disadvantage with preprocessing step is that it can induce performance degradation in LSA process. In some cases, a document may involve entire use of stop words, making it impossible to retrieve such documents if all or most of their terms are eliminated in the process. Stemming and lemmatization strategies used may also affect the performance of the retrieval system e.g. where a Porter stemmer is used, the following words might all be stemmed to *oper* which affects the querying for *operating system, operative dentistry* and *operational research*:

operate ,operated, operating, operates, operation, operative, operatives, operational

4.2.2. Vector Space Model (VSM)

After the preprocessing step, the collection is mapped into a VSM of dimensions m x n. A VSM is an algebraic model for representing units and compositions contained in a collection as vectors. This model can be visualized as a matrix A of dimensions m (number of terms/units in a collection) by n (number of documents/compositions in the collection). VSM maps every document into a vector c_j where $1 \leq j \leq n$ and c_j represents the columns of term-document matrix A. This is shown below:

$$A := (c_1, c_2, \ldots\ldots, c_n) = (r_1, r_2, \ldots., r_m)^T \in \mathbb{R}^{m \times n} \quad (4.1)$$

In the equation above, $r_1, r_2, \ldots..,r_m$ are the rows of the matrix A. T denotes transposition. Each entry $A_{i,j}$ of the VSM represents the weight of i-th term/unit in j-th document/composition. Two approaches that can be used for weighting terms or units are discussed below.

a) *Term frequency ($tf_{i,j}$)*

This approach assigns weights to terms based on how many times an *i*-th term occur in a document *j*. The critical problem faced by this approach is that all terms used are considered to be of equal relevance to the query answering process.

Consider the document shown below which describes functions of a *grinding wheel* [46]:

> **Grinding wheels** are self **sharpening** to a small **degree.** For optimal use they may be dressed and trued by the use of **grinding dressers**. **Dressing** the **wheel** refers to removing the current layer of **abrasive**, so that a fresh and sharp **surface** is exposed to the work **surface**. **Truing** the **wheel** makes the **grinding** surface parallel to the **grinding table** or other reference **plane**, so the entire **grinding wheel** is even and produces an accurate **surface**. The **wheel** type fit freely on their supporting **arbors**, the necessary **clamping force** to transfer the rotary **motion** being applied to the **wheels** side by identically sized **flanges**.

Assuming after the preprocessing stage the terms in bold are retained in the dictionary, the terms *table, motion, dressing,* and *flanges* marked by bold would weigh equally even though it appears logical to weigh the term *table* less. The other problem with term frequency weighting arises from differences in documents' lengths with terms likely to weigh highly within long documents. This is because a higher term frequency for a given term is more likely in a longer document than in a shorter document. To prevent a bias towards large documents where terms may have unnecessarily high frequency, it is desirable that term frequency ($tf_{i,j}$) is normalized with a per document factor λ_j.

$$A_{i,j} = \frac{tf_{i,j}}{\lambda_j} \tag{4.2}$$

Depending on the application, different ways may be used to calculate λ_j *factor.*

Examples of λ_j *calculation*

I. $$\lambda_j = \left|\left|tf_{i,j}\right|\right|1 := \sum_i \left|tf_{i,j}\right| \quad \text{where } \left|\left|.\right|\right|1 \text{ is one-norm} \tag{4.3}$$

II. $$\lambda_j = \left|\left|tf_{i,j}\right|\right| := \sqrt{\sum_i f^2_{i,j}} \quad \text{where } \left|\left|.\right|\right| \text{ is two-norm} \tag{4.4}$$

b) Tf-idf weighting

This approach of assigning weights is better than the term frequency discussed above as it combines both local and global statistics from a collection. This weighting scheme combines term frequency ($tf_{i,j}$) and inverse document frequency (idf_i). Inverse document frequency comes up with a better discriminating mechanism for query answering by considering document-level statistics. The aim is to have terms that appear in many documents weigh less on a query than those terms which appear in fewer documents. The inverse document frequency of *i*-th term in a collection of *N* documents is given by:

$$idf_i = log \frac{N}{df_i} \tag{4.5}$$

Where df_i is the document frequency (number of documents in the whole collection that contains the term *i*). Using the inverse document frequency of *i*-th term in j-th document, we combine term frequency explained earlier to weight terms as shown below.

$$Tf\text{-}idf_{i,j} = tf_{i,j} * idf_i \tag{4.6}$$

The above equation means that $tf\text{-}idf_{i,j}$ assigns to a particular term *i* a weight in a document *j* that is [32]:

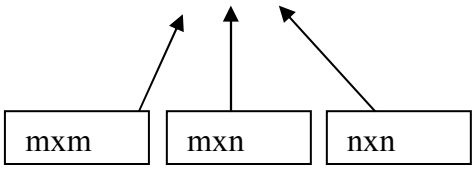
- Highest when term *i* has high occurrence within a small number of documents
- Lower when term *i* occurs fewer times in a document or occurs in many documents
- Lowest when the term occurs in virtually all documents.

4.2.3. Dimensionality reduction

Dimensionality reduction is a strategy aimed at ensuring economical representation of data and unearthing of semantic structures within represented data by noise elimination. In LSA, the vector space model mentioned earlier is very sparse and large to the effect that it would demand more storage space in computer memory. Singular Value

Decomposition (SVD) is used to address this problem of sparsity as well as to unearth the latent semantic structures. SVD is a technique closely related to eigenvector decomposition and factor analysis. It is an important factorization of a rectangular real or complex matrix used in linear algebra. The *m*-by-*n* matrix *A* of rank r is factorized as shown below:

$$A = USV^T \tag{4.7}$$



Where:

- U is a matrix of dimensions *m* by *m* whose columns are orthogonal eigenvectors of AA^T matrix
- S is a matrix of dimensions *m* by *n* whose diagonal values are singular values of matrix A and with nonnegative numbers on the diagonal
- V^T is a transpose of V where V is a matrix of dimensions *n* by *n* whose columns are eigenvectors of A^TA matrix.

Dimensionality reduction involves truncating the three matrices obtained by full SVD. Basically, the first *k* columns of *U*, the first k rows of V^T and the first *k* rows and *k* columns of *S* are retained. The underlying principle is to have a matrix A_k of rank at most *k,* so as to minimize the Frobenius norm of the matrix difference $X=A - A_k$ which measures the discrepancy between matrix A_k and matrix *A* and defined as:

$$\left|\left|X\right|\right|_F = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N} X_{ij}^2} \tag{4.8}$$

The goal in this case is to find a matrix A_k that minimizes this discrepancy, while constraining A_k to have a rank of at most *k.* This process of finding *k* such that the matrix

A_k has a rank lower than that of the original matrix A is called low-rank approximation. There has been no agreed general strategy so far for deciding the optimal *k* to use for the retained dimensions; it is rather an empirical issue and depends on methods used for the evaluation of the retrieved results. If *k* is too large we may have more noise in the vector space while too low *k* may lead to factors loosing important information. The reduction process is demonstrated below:

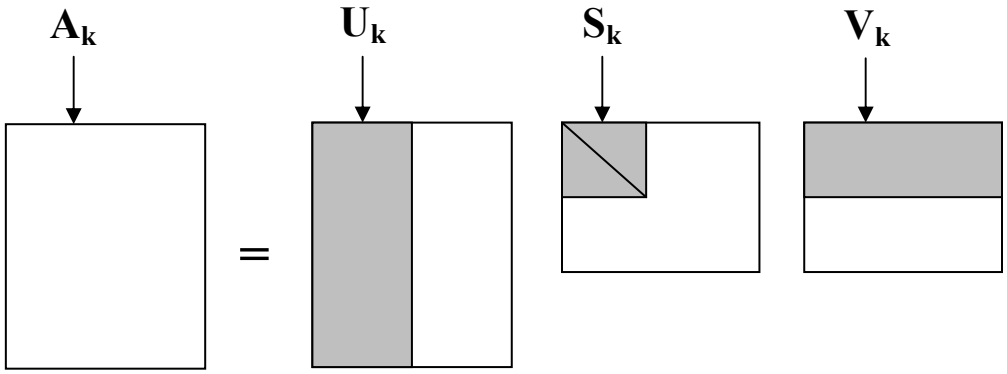


Figure 4.2: Illustrating dimensionality reduction

Dimensionality reduction yields a new representation for both terms and documents in the collection. While VSM discussed above is able to treat queries and documents uniformly, dimensionality reduction reveals semantic structures that would not be revealed by VSM. In the reduced *k*-space, terms which occur in similar documents are mapped close to each other even though they may not co-occur in the same document. Query terms are also mapped in the same *k-space* where similarity metrics can be used to measure distances between query terms and documents. It is on this comparison of documents and queries on the same *k-space* that LSA is used for information retrieval.

4.2.4. Query mapping on k-vector space

To answer queries using LSA process, queries have to be mapped on the semantic space. A query vector $\vec{q}$ is represented in the *k-vector space* by the transformation below:

$$\vec{q}_k = S^{-1}{}_k U^T{}_k q \qquad (4.9)$$

Once a query has been mapped into the reduced vector space (*k-vector space*) it can be compared against the documents in the vector space using a similarity metric. In the vector space model, *angle cosine* is used to calculate the distance from a document *d* to any query *q*. This method can also be used to calculate the distance between any two documents or terms. Measuring the distance from a document to a query involves the following:

Assume we denote by $\vec{V}(d)$ the vector representing document *d,* with every dictionary (all key words used in the generation of term-document matrix) term represented. Suppose we also denote a query by vector $\vec{V}(q)$. Then the angle cosine which is used to show the score of query *q* on document *d* is calculated as:

$$\text{score (q, d)} = \frac{\vec{V}(q).\ \vec{V}(d)}{\left|\vec{V}(q)\right|\left|\vec{V}(d)\right|} \qquad (4.10)$$

From the above equation the numerator represents the dot product of the two vectors $\vec{V}$ (q) and $\vec{V}$(d) which would sufficiently be enough to measure the score if document *d* and query *q* were of the same length. However, since this is not the case, the denominator is introduced as a way of length normalization. This is called Euclidean normalization, since it involves the calculation of Euclidean lengths for each vector being considered. To rank the other documents based on the query the score is calculated as above for each document and based on the result set, the document that score highest is ranked first.

4.3. Comparing LSA Model and Human Memory Models

In Chapter 2, the taxonomic model of the brain was briefly introduced. According to the model, brain stores similar items together, e.g. knowledge relating to cars, buses or trains would be stored together. Likewise, LSA represents semantically similar items together. LSA model can also be compared to the learning process of the human memory. Human memory acquires chunks of data daily from which it discovers and generates knowledge implicitly. On the other hand, LSA obtains documents and discovers knowledge implicit in those documents.

There has also been previous work that compared how LSA mimics human judgements. In [21], LSA performance on TOEFL (Test of English as a Foreign Language) was simulated and compared with average test-takers. It was shown that LSA scored 65% correct which was identical to the average score of a large sample of students applying for college entrance in the United States of America from non-English speaking countries.

4.4. Comparing LSA and Semantic Web

LSA and Semantic Web can be compared in terms of how knowledge is represented in each case. In Semantic Web, knowledge is modelled explicitly by use of ontologies. Ontologies represent sets of interrelated concepts. On the other hand, LSA models knowledge implicitly by representing documents into a continuous vector space where dimensionality reduction is applied.

4.5. Measuring LSA Performance

To evaluate the performance of LSA in connecting people-with-documents, two performance metrics explained below can be used:

- *Recall:* gives the fraction of the relevant documents in the repository that are returned. A recall of 100% is achieved if all documents in the repository are retrieved.

$$\text{Recall} := \frac{\left|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}\right|}{\left|\{\text{relevant documents}\}\right|} \qquad (4.11)$$

- *Precision:* gives the fraction of the returned results that are relevant to the query.

$$\text{Precision} := \frac{\left|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}\right|}{\left|\{\text{retrieved documents}\}\right|} \qquad (4.12)$$

4.6. Applications of Latent Semantic Analysis

The application of LSA appears promising in a number of fields due to [6]:

a) The mapping of discrete entities (units and compositions) onto a continuous parameter space, where efficient machine learning algorithms can be applied
b) The dimensionality reduction inherent in the process, which makes complex natural language problems tractable
c) The intrinsically global outlook of the approach which tends to compliment the local optimization performed by more conventional techniques.

From the points given above, LSA can be generalized to Latent Semantic Mapping (LSM). This will help in expanding the application of LSA in other areas as opposed to the original model only developed to capture hidden words pattern in a text document corpus. LSM requires discrete entities (units and compositions) as noted above. The table

below shows examples of potential areas of applications with the corresponding discrete entities considered.

Application Area	Units	Compositions
Information Retrieval [23]	terms	Documents
Junk email filtering [6]	word, symbols	Emails
Speech recognition [6]	letter n-tuples	Words
Text summarization [41]	terms	Sentences
Speech synthesis [6]	pitch periods	time slices
Document clustering [6]	Terms	Documents

Table 4.2: Potential Applications of LSA/LSM

4.6.1. Information Retrieval

LSA use in information retrieval is motivated by its capability to address natural language challenges like polysemy and synonymy [23]. The relationships between documents in a collection and the corresponding terms are harnessed using mathematical/statistical techniques. This results into mapping of documents into one semantic space. Documents related in terms of their semantics are located close to each other. A query is also mapped to the same semantic space, making it possible to retrieve documents that are closely related to it. This retrieval by LSA is on the basis of conceptual content as opposed to mere matching of terms.

4.6.2. Documents Clustering

Document clustering is an [16] automatic grouping of text documents into clusters so that documents within a cluster have high similarity in comparison to one another, but are dissimilar to documents in other clusters. By mapping documents into a common semantic space, LSA can be used to reveal documents clusters based on their similarities.

4.6.3. Junk Email Filtering

Junk email filtering aims at [37] blocking unsolicited messages with commercial or pornographic content, otherwise known as spam, while allowing other messages to pass. This requires enquiry into what constitutes a legitimate or illegitimate email message. Some of the approaches used to deal with spam includes [6] header analysis, rule-based predicates, and/or machine-learning approaches.

LSA presents a different approach that examines whether or not the latent subject matter of an email message is consistent with the user's interests. The process of junk-email filtering by LSA can be seen as clustering documents into two clusters: legitimate and junk. This happens by use of two phases: training and actual classification phase. Training phase seeks to identify characteristics of junk and legitimate emails by use of an email collection (training set). Classification phase then performs the actual task of differentiating each incoming email into either of the clusters. This is demonstrated by the diagram shown below.

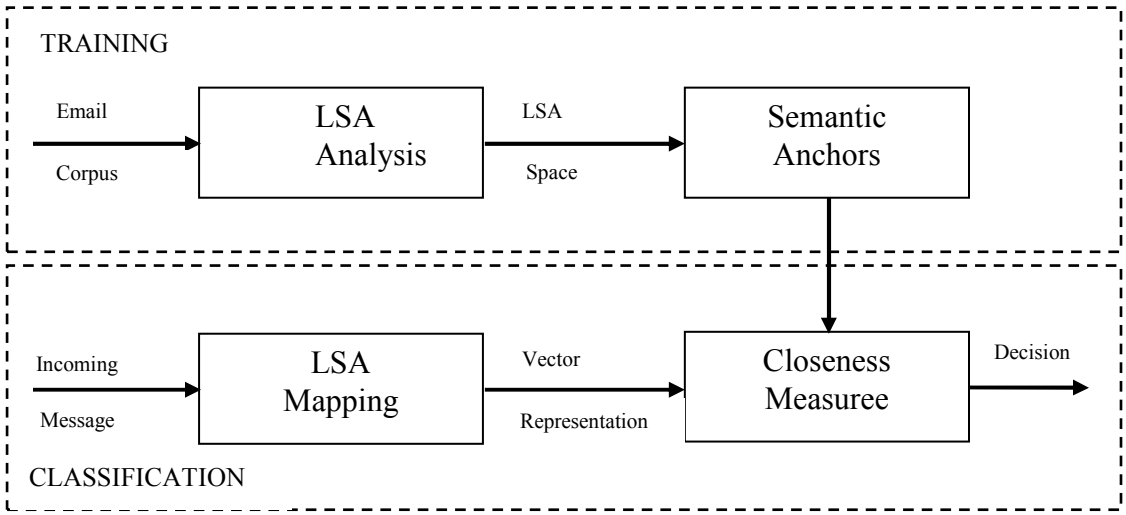


Figure 4.3: Demonstration of LSA junk-email filtering [6].

In the diagram shown above, training step uses an Email corpus assembled from a number of previously received legitimate and junk messages. LSA is then applied to generate a semantic space characterizing what constitutes legitimate versus junk email.

This knowledge is automatically assembled in a vector space featuring two semantic anchors, one representing the centroid of all E-mails that are allowed to the user and the other representing the centroid of E-mails that are not. The centroid of each cluster is the semantic representation of the associated category in the LSM space. Then each incoming message is evaluated against these two anchors using the closeness measure, which indicates how related it is to what the user likes and dislikes.

4.6.4. Language modeling

Language modeling serves a pivotal role in language processing. The fundamental function of language modeling is to [6] encapsulate as much as possible of the syntactic, semantic, and pragmatic characteristics for the task considered. Latent semantic analysis is particularly important in dealing with semantic characteristics in language modeling.

4.6.5. Speaker Verification

Speaker verification serves the purpose of accepting or rejecting the identity of a speaker on the basis of individual information present in the speech waveform. It can be used to replace traditional password-matching schemes, where speakers identify themselves to systems through their spoken voice. Speaker verification can be done over the phone or just used on desktop voice login. Speaker verification by LSA entails [6] the comparison of the acoustic sequence uttered during recognition (verification utterance) with the aggregated acoustic evidence collected during training (key phrase specific reference speaker model).

4.6.6. Text Summarization

The increasing amounts of information demands attempts to summarize it into what is relevant to the user's needs. In this approach SVD is applied on a matrix A of dimension *m* by *n* where *m* is the *terms* contained in the sentences belonging to the document to be summarized and *n* the *sentences* of the document. Applying SVD on this matrix captures the interrelationships between terms and sentences thereby unearthing deeper sentence semantics. From the process of SVD explained earlier, summarization method uses the matrix V^T [17]. This matrix describes an importance degree of each topic in each sentence. The summarization process selects the most informative sentence for each topic.

4.7. Limitations of Latent Semantic Analysis

This section introduces the limitations characterizing LSA technique.

4.7.1. Speed & Updating

The speed of LSA has limited the production of software applications that can be acquired off-shelf by companies despite having been patented in the year 1988. Its major problem is that it is formulated in terms of large matrix operations on the corpus as a whole, i.e. a batch algorithm, demanding huge memory and CPU time in the application of SVD [18]. Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, computing a full SVD is fundamentally an O(mn.min(m,n))- time problem, making decompositions of extremely large matrices infeasible [7]. The other problem of LSA is updating of the data matrix when new data arrives. Updating of the data matrix demands a full recomputation of SVD which is computationally expensive. Alternatively new data could be folded into the low rank space. Folding new data into the process however ignores any component of new data that lies outside the reduced space [7].

To benefit from LSA a lot of research has gone into the speed of the technique or other algorithms that could allow incremental updating for new data without recomputation. An incremental algorithm, Generalized Hebbian Algorithm (GHA) which performs similarly to LSA has been introduced in [18]. GHA allows incremental derivation of the Eigen decomposition of an unseen matrix based on serially presented observations. The advantage of GHA is that it doesn't require that the entire matrix be held in memory simultaneously making large corpora to be used. However, it only provides stochastic approximations as opposed to exact solutions. Brand [7] has introduced recently an algorithm for computing SVD and dimensionality reduction which is accurate, fast, incremental and low-memory demanding making it possible to handle large matrices much faster.

4.7.2. LSA on Count Data

LSA involves handling of count data such as term frequency and inverse document frequency. The choice of LSA on count data has being questioned in [20]. Firstly, the application of SVD on count data is somewhat ad hoc. Secondly, the application of Frobenius norm approximation principle is reminiscent of a Gaussian noise assumption which is hard to justify in the context of count variables. For these reasons, Probabilistic Latent Semantic Analysis (PLSA) is argued to be more principled than LSA. Unlike LSA that is based on SVD and Frobenius norm, PLSA is based on the statistical latent-class model that uses probability functions to predict latent variables from given observations. Despite approximation of latent variables in PLSA as opposed to SVD which can be computed in exact way, PLSA is methodologically more principled and experiments done show that it performs better in information retrieval than LSA. More details regarding PLSA can be found in [20].

4.7.3. Incompleteness of LSA

Estimating the number of dimensions to use for representation of data in LSA remains an empirical function. In the experiments done in this thesis, selection of optimal *k (*number of dimensions to retain in low rank approximation*)* appeared to correlate with the number of topics within the repository. However, a more principled approach is required.

4.8. Summary

This chapter introduced the principles of LSA and some areas where it can be applied. Though primarily developed for information retrieval, LSA can be generalized for use in other areas as shown above. Based on the objective of this work i.e. to achieve people-with-documents connections, major limitations of LSA technique have been investigated. Possible solutions to mitigate the limitations have also been briefly introduced. The major limitations include: incompleteness of the technique, slow computation speed and updating problem. In the next chapter, experiments to achieve people-with-documents have been highlighted where precision performance appears to be another limitation towards the achievement of people-with-documents connections.

5. Practical Work

This chapter introduces the practical work done during the entire period. Preliminary work started by developing an application based on term-matching technique to realize knowledge sharing. The work was followed by investigation into LSA performance and finally the development of an ontology to be used for documents annotations.

5.1. Preliminary Knowledge Sharing Model Based on Term-Matching Technique

Our initial model was based on establishing the effectiveness of term matching in achieving people connections. To capture knowledge, the practical questions given below were formulated:

1. What specific tasks, processes or activities can you perform?
2. With what can you perform (1) above?
3. Where can you perform (1) above?
4. What are the results/effects/products of (1) above?
5. With what quality can you perform (1) above?
6. How can you be contacted?

The model was realized as a web-based application within the company with contacts from each person captured in a database. Term-matching technique was then applied to search for an engineer, his/her competences and contact him/her in case further sharing of knowledge is required. This method was ineffective due to the limitations discussed in chapter 1. Following these limitations, LSA was investigated onwards.

5.2. LSA Performance

To achieve people-with-documents connection for knowledge sharing, the following three requirements are crucial:

- High recall performance
- High precision performance
- High retrieval speed

In the previous chapter, LSA limitation in speed was highlighted as a result of SVD computations. In this chapter, recall and precision performance of LSA are investigated. The experiment done is described in the next paragraph.

LSA was investigated in a work which compared its performance in documents retrieval to term-matching technique introduced in Chapter 1. Investigation was done on a repository consisting of 100 documents from the following subjects**:** *Psychology, Sociology, Mathematics, Religion, Politics, Economics, Business, Computer Science, Medicine* and *Sports***.** The documents were manually prepared from Wikipedia [46], the online Encyclopedia, from which 10 documents were prepared about each subject. The performance of LSA was compared with term matching technique to evaluate recall and precision performance. *JDesktopSearch* application was used for term matching technique and a MATLAB-based application was used for LSA operations. From the dictionary, two terms were selected to search documents about each subject. In total, 20 terms shown below were used in the experiment.

psychology, sociology, mathematics, business, medicine, politics, economics, sport, medicine, computer, cognition, culture, calculus, government, supernatural, inflation, athletics, disease, program, customer.

Table 5.1: Terms used in the experiment involving term-matching and LSA comparison

The results were recorded in the table shown below. Two graphs were then drawn to visualize the performance of LSA in retrieving relevant documents as well as irrelevant documents (false positives).

Term	*Total number of relevant documents in the collection*	*Number of documents retrieved using term-matching*			*Number of documents retrieved by LSA approach*		
		Total	*Relevant*	*Irrelevant*	*Total*	*Relevant*	*Irrelevant*
Psychology	10	9	9	0	10	9	1
Sociology	10	8	8	0	10	8	2
Mathematics	10	7	7	0	10	8	2
Politics	10	2	2	0	10	4	6
Religion	10	3	3	0	10	7	3
Economics	10	3	1	2	10	6	4
Sport	10	2	2	0	10	8	2
Medicine	10	3	3	0	10	8	2
Computer	10	3	3	0	10	8	2
Business	10	10	8	2	10	9	1
Cognition	4	3	2	1	10	4	6
Culture	2	2	2	0	10	2	8
Calculus	2	2	2	0	10	2	8
Government	10	10	6	4	10	6	4
Supernatural	1	1	1	0	10	1	9
Inflation	3	2	1	1	10	3	7
Athletics	3	1	1	0	10	3	7
Disease	5	7	3	4	10	5	5
Program	5	3	2	1	10	5	5
Customer	2	2	1	1	10	1	9

Table 5.2: Comparing Term-Matching Technique with LSA

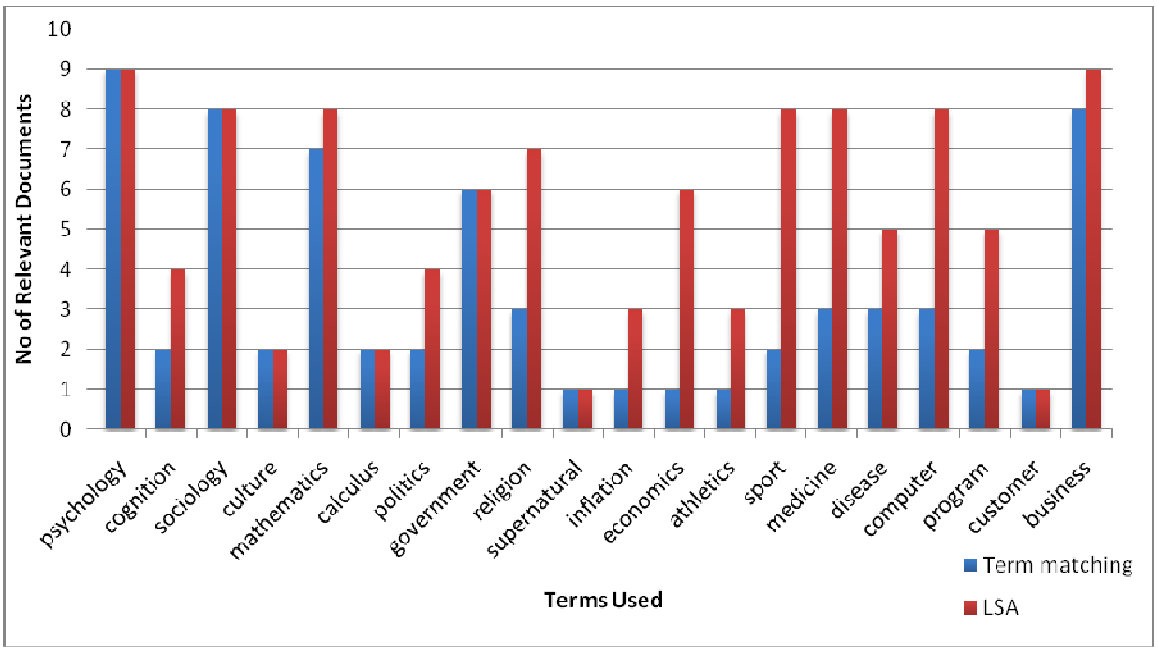


Figure 5.1: A graph showing comparison between LSA and Term-Matching Technique in retrieval of relevant documents in a context of document retrieval.

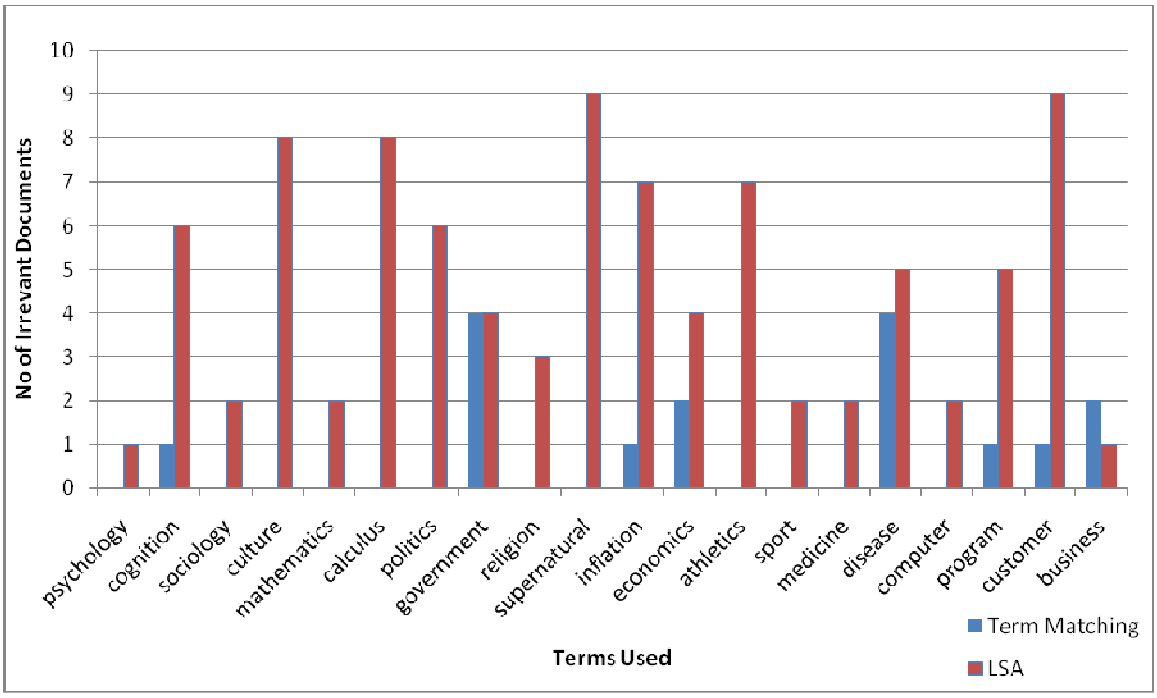


Figure 5.2: A graph showing comparison between LSA and Term-Matching Technique in retrieval of irrelevant documents in a context of document retrieval.

Compared to term-matching technique, LSA was shown to perform better in terms of recall. However, this high recall comes with a limitation of retrieving false positives i.e. decreased precision performance. Other experiments have also shown a decrease in precision with increased recall performance. In [20], the performance of LSA was tested against PLSA and vector based similarity (cos) and based on the results, precision of the three algorithms decreases with an increased recall.

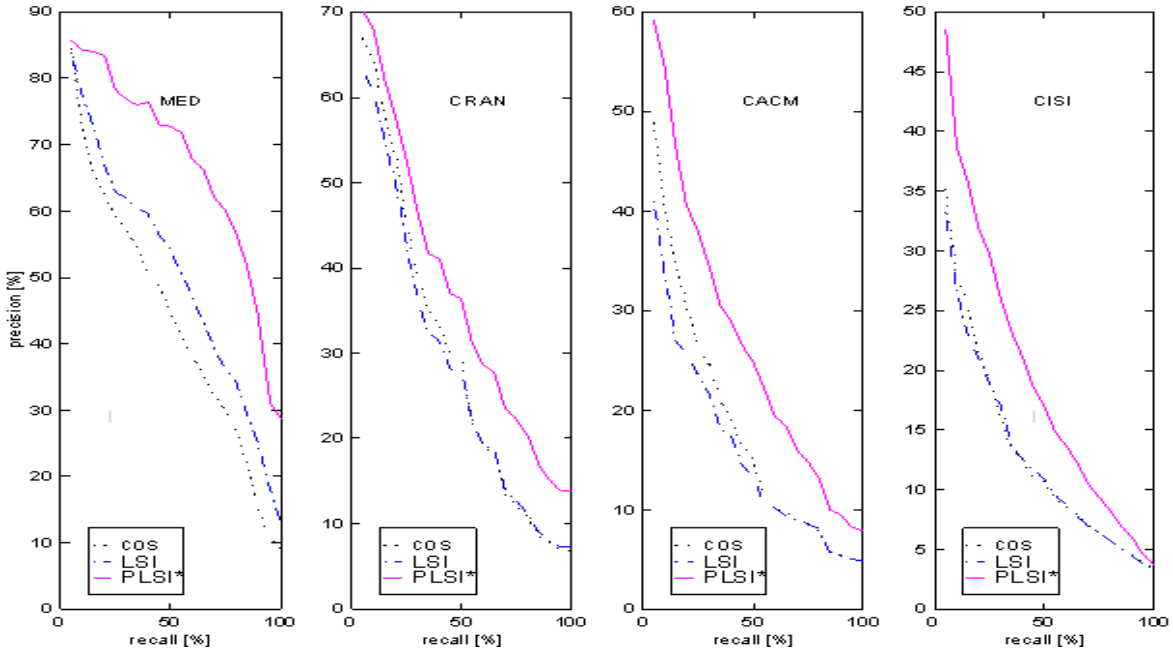


Figure 5.3: Precision-Recall Comparison of PLSA, LSI and Cos [20]

On the next page, the inclusion of documents' annotations to the LSA process as a way of improving knowledge discovery and representation from documents to improve precision performance is proposed.

5.3. Investigating the inclusion of annotations into LSA technique

Based on the experiment above, LSA performs better in terms of recall. However, this comes with a drawback of retrieving false positives. In this section, the inclusion of annotations in LSA process is investigated to improve knowledge discovery on the semantic space. This is expected to improve precision performance. As was discussed in chapter 3, ontology-based document annotation is suggested.

The use of annotations could be exploited in two ways in our knowledge sharing model:

- Use of annotation process to build a recommendation system.
  The annotation process could be used to extract knowledge contained in a document. Processing of these annotations could help a new user evaluate a document without necessarily reading it. This could also help to connect to other documents with similar knowledge by the same author or a different author.
- Inclusion of annotations into the LSA algorithm
  In this case, annotations are represented on the LSA semantic space to help improve knowledge representation within the semantic space.

In this work, I propose the use of the second approach, the inclusion of annotations in the LSA process. The next section highlights some challenges to be addressed.

5.3.1. Challenges

The following challenges need to be addressed:

- How do we develop an ontology about the domain?
- How do we annotate document text using a domain ontology?
- How do we store and incorporate annotations into the LSA process?
- How do we process images contained in documents?

  Computer vision techniques basically process images by extraction of low-level features. On the other hand, human beings associate images with high-level concepts in everyday life. This variation commonly referred to as [30] semantic gap complicates the processing of images. Using the ontology, how do we annotate the images? The problem of image annotation can be broached from machine learning [30]. Supervised learning process involving manually classifying training images into a set of classes or concepts would be done first. A binary classifier would then be trained to detect a class (concept) to allow annotation for a new image. To annotate a new image, the visual similarity to each class from the ontology is then computed. Image annotation is not further investigated in this work.

In the next section, I have proposed a solution that will implement people-with-documents connections while addressing the above challenges.

5.3.2. Inclusion of semantic annotations into LSA process

To achieve people-with-documents connections using LSA, I propose a model with the following steps:

- Development of the ontology about the domain
- Ontology-based documents annotation to generate semantically rich documents
- LSA process on the semantically rich documents
- Retrieval of documents to achieve people-with-documents connections

These steps are summarized by the model shown below:

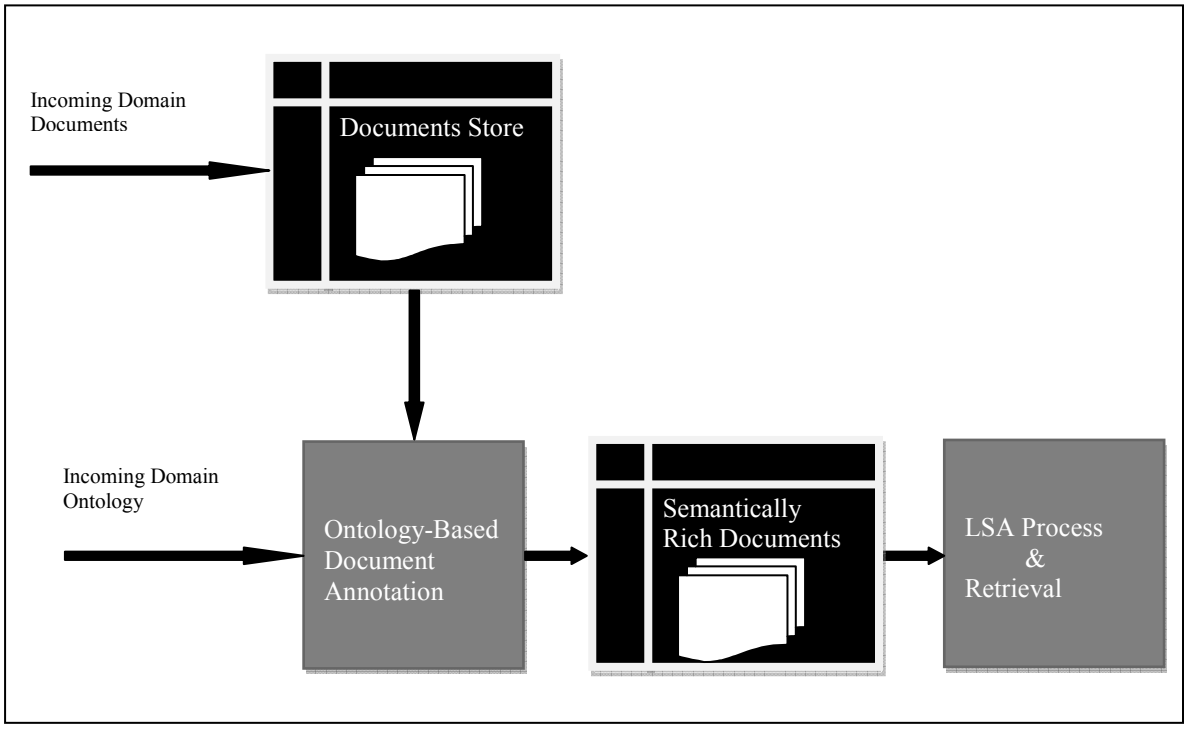


Figure 5.4: Knowledge sharing model based on LSA

5.3.2.1. Implementation

The implementation of the above steps is discussed below:

Ontology Development: to develop an ontology about a domain, a language for its representation has to be selected. Web Ontology Language (OWL) is chosen due to its expressiveness as compared to other languages like Resource Definition Framework (RDF). The following steps were used to come up with a small ontology about *grinding industry:*

- Identification of classes within grinding industry such as *grinding wheels*, *filtering systems, coolants, workpiece*, etc
- Identification of class hierarchy: The most general classes were identified followed by specialization of those classes e.g. I started with a *wheel* then specialized it into *grinding wheel*

- Identification of properties of classes. Two types of properties were identified, data and object type properties. Data properties identify attributes for a particular class, e.g. for *grinding wheel* we would have properties like *grinding time*, *bore diameter*, *axial feed rate*, etc. Object type properties point to other classes that relate to the class being considered e.g. the property *supplying* for *coolant nozzle* points to class *coolant. (i.e. coolant nozzle* supplies *coolant*)
- Definition of constraints for classes and their properties to specify allowed value types, allowed values, the number of values and other features unique to each class or property.

There are a few tools that can be used to develop ontologies. For our experiments, Ontomat-Annotizer tool was used to make a small ontology about grinding industry within the company using the steps listed above. OntoMat-Annotizer is a tool being developed by the Institute AIFB at the University of Karlsruhe for creating OWL annotations [29]. The screen shot below captures the ontology developed within Grindaix GmbH Company.

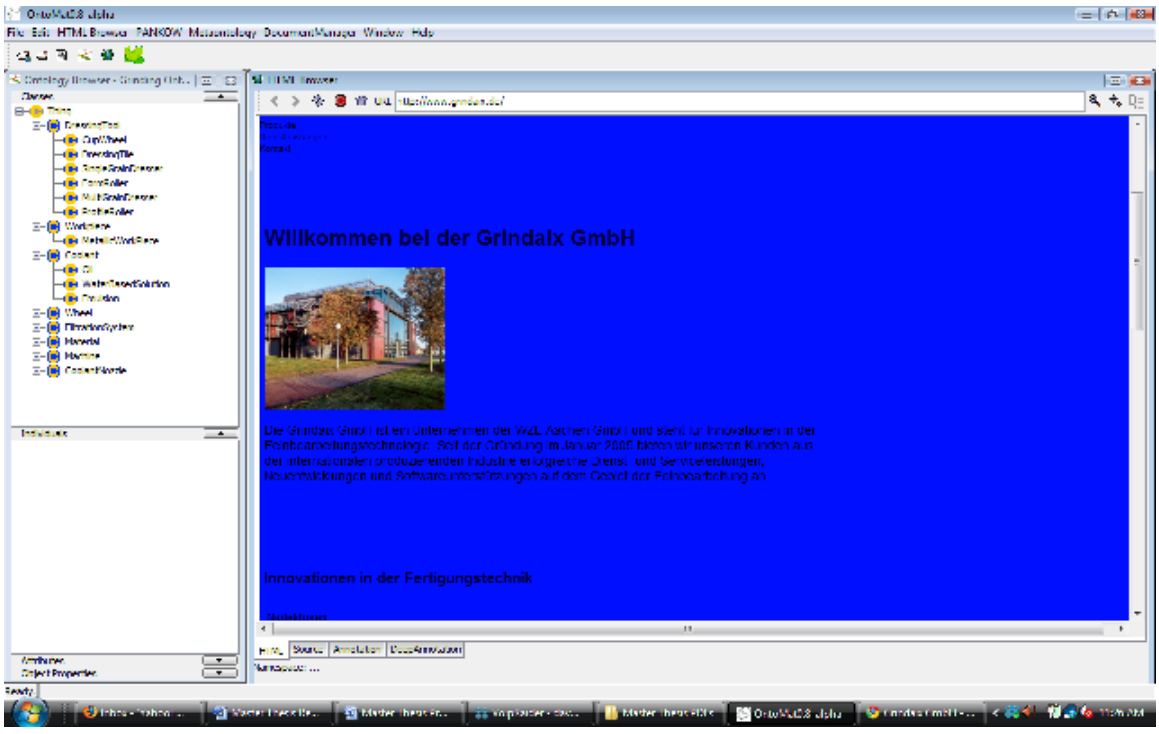


Figure 5.5: OntoMat-Annotizer used to develop ontology

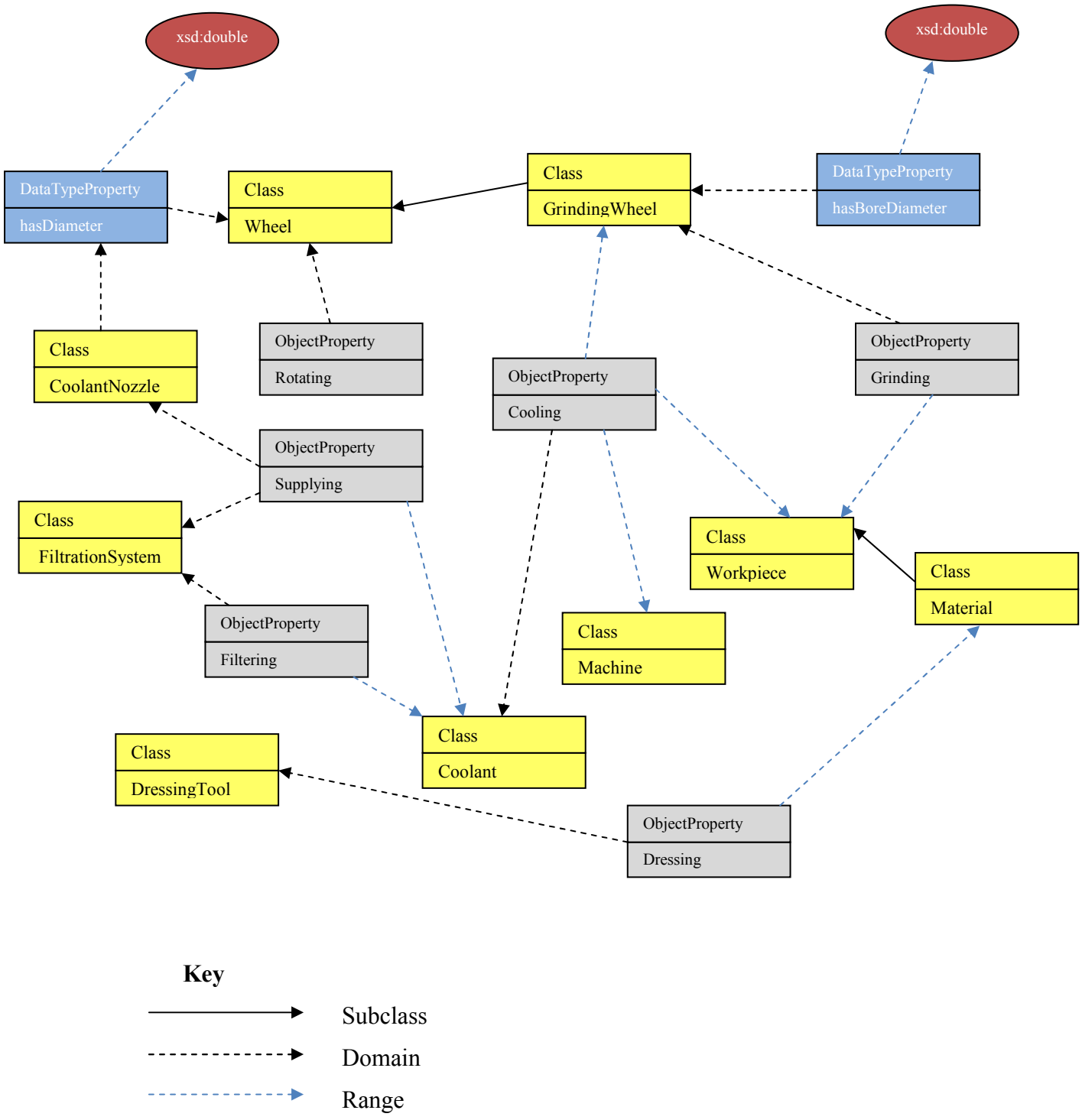


Figure 5.6: A section of ontology about grinding domain

The above ontology is shown on the index part as an OWL file.

Ontology-based document annotation: Documents can be annotated automatically, semi-automatically or manually. In this work, manual annotation of a few documents from *grinding industry* was performed using the developed ontology. Considering the current speed for computation of SVD that makes LSA slow, I propose to represent the semantic annotations within the original documents as opposed to producing separate annotation documents based on the formal languages discussed in Chapter 3.Consider the document below:

> *Grinding process requires cooling oil. To ensure high exit speed of the oil, directed coolant supply into the cutting zone, cost saving for pumps and filtration system and the reduction of thermal workpiece damage, Grindaix GmbH developed a needle nozzle. The nozzle can be used during any grinding process such as gear grinding, profile grinding, internal grinding, surface grinding, centerless grinding and external grinding among others.*

In this work, I focused on the identification of concept/class instances within the documents, and then marking the instances with the class names. This results into semantically-enriched documents as shown below:

> *Grinding process requires cooling* ***<Coolant>*** *oil* ***</Coolant>****. To ensure high exit speed of the* ***<Coolant>*** *oil* ***</Coolant>****, directed coolant supply into the* ***<GrindingWheel>*** *cutting zone* ***</GrindingWheel>****, cost saving for pumps and filtration system and the reduction of thermal workpiece damage, Grindaix GmbH developed a* ***<CoolantNozzle>****needle nozzle* ***</CoolantNozzle****. The* ***<CoolantNozzle>****nozzle****</CoolantNozzle>*** *can be used during any grinding process such as gear grinding, profile grinding, internal grinding, surface grinding, centerless grinding and external grinding among others.*

In the above case, *oil*, *cutting zone*, *needle nozzle* are instances of classes *Coolant*, *GrindingWheel* and *CoolantNozzle* respectively. MnM, which is an annotation tool that provides both automated and semi-automated support for marking web pages with semantic contents, can be explored for use in this step [43].

LSA process and retrieval: LSA is finally applied onto the annotated documents (semantically-rich documents) to represent knowledge from all documents in one common space from where queries can be made. The challenge in this case is how to represent and weight annotations into the LSA process. I propose an introduction of a common vector space where annotations live with the original text for the application of SVD. This is illustrated below.

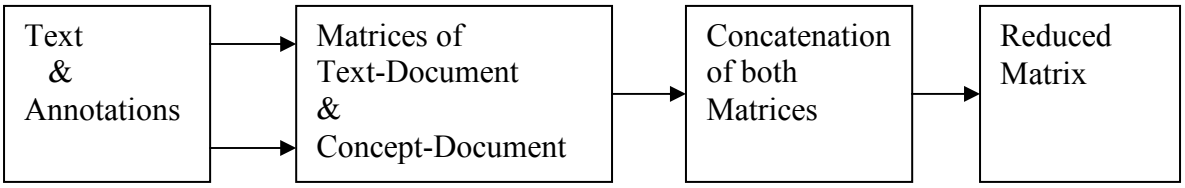


Figure 5.7: Representing annotations and documents in one semantic space

Assume that the original documents generate a *term-document* matrix $A \in \mathbb{R}^{m \times n}$ with dimensions *m* (dictionary length) by *n* (number of documents) where each entry $A_{i,j}$ represents Tf-idf weight for each term. The annotations can be treated in the same way. To do this, we take all the concepts' names represented in the ontology to make a new concept dictionary of length *c*. We then find instances of each concept in each document. To calculate Tf-idf for each concept name within each document, we proceed in the same way as for term-document matrix i.e. for concept *i* we count the number of instances in a document *j* and then count the number of documents from the total *n* documents that have instances of concept *i*. Through this weighting scheme, we get another matrix which I have called *concept-document* matrix $B \in \mathbb{R}^{c \times n}$ where *c* is the number of concepts and *n* is the number of documents. To get one common vector space, we concatenate the rows of matrix B to the rows of matrix A. This gives us a new vector space $X \in \mathbb{R}^{(c+m) \times n}$ where SVD can be applied. The process above is illustrated below:

Assume we have 5 documents shown below that we want to process using LSA technique:

1. Never grind a soft object such as Aluminium
2. Powertec wheel can use grinding oil
3. Too much oil produces a thick black layer on the wheel face

4. Needle nozzle can greatly reduce temperature on a wheel
5. The use of water filtering systems can ensure effectives of grinding process

If the above documents were to be processed using LSA, the term-document matrix $A \in \mathbb{R}^{15 \times 5}$ shown below would be generated after the preprocessing step. Entries represent Tf-idf weights of the respective terms.

	Doc 1	**Doc 2**	**Doc 3**	**Doc 4**	**Doc 5**
Grind	0.2218	0.22218	0	0	0.2218
Object	0.6990	0	0	0	0
Aluminium	0.6990	0	0	0	0
Powertec	0	0.6990	0	0	0
Wheel	0	0.22218	0.22218	0.22218	0
Oil	0	0.22218	0.22218	0	0.22218
Black	0	0	0.6990	0	0
Layer	0	0	0.6990	0	0
Surface	0	0	0.6990	0	0
Needle	0	0	0	0.6990	0
Nozzle	0	0	0	0.6990	0
Temperature	0	0	0	0.6990	0
Filter	0	0	0	0	1.3979
System	0	0	0	0	0.6990
Process	0	0	0	0	0.6990

Table 5.3: A sample term-document matrix

Assuming we use 5 concepts (*WorkPiece*, *GrindingWheel*, *Coolant*, *CoolantNozzle*, *FiltrationSystem*) from the ontology shown on the index part, we perform annotation as shown below:

1. Never grind a soft **<WorkPiece>** object **</WorkPiece>** such as **<WorkPiece>** Aluminium **</WorkPiece>**
2. **<GrindingWheel>** Powertec wheel **</GrindingWheel>** can use grinding **<Coolant>**oil **</Coolant>**
3. Too much **<Coolant>**oil**</Coolant>** produces a thick black layer on the **<GrindingWheel>** wheel **</GrindingWheel>** surface
4. **<CoolantNozzle>** Needle nozzle **</CoolantNozzle>** can greatly reduce temperature on a **<GrindingWheel>** wheel **</GrindingWheel>**
5. The use of **<FiltrationSystem>** filtering systems **</FiltrationSystem>** to filter **<Coolant>** oil **</Coolant>** ensures effectiveness of a grinding process

From the annotations shown above, we can generate concept-document matrix B $\in \mathbb{R}^{5 \times 5}$ shown below:

	Doc 1	**Doc 2**	**Doc 3**	**Doc 4**	**Doc 5**
WorkPiece	1.3979	0	0	0	0
GrindingWheel	0	0.22218	0.22218	0.22218	0
Coolant	0	0.22218	0.22218	0	0.22218
CoolantNozzle	0	0	0	0.6990	0
FiltrationSystem	0	0	0	0	0.22218

Table 5.4: A sample concept-document matrix

We now concatenate the rows of the two matrices to get a matrix $X \in \mathbb{R}^{20 \times 5}$ shown below from which normal dimensionality reduction can take place.

	Doc 1	**Doc 2**	**Doc 3**	**Doc 4**	**Doc 5**
Grind	0.2218	0.22218	0	0	0.2218
Object	0.6990	0	0	0	0
Aluminium	0.6990	0	0	0	0
Powertec	0	0.6990	0	0	0
Wheel	0	0.22218	0.22218	0.22218	0
Oil	0	0.22218	0.22218	0	0.22218
Black	0	0	0.6990	0	0
Layer	0	0	0.6990	0	0
Surface	0	0	0.6990	0	0
Needle	0	0	0	0.6990	0
Nozzle	0	0	0	0.6990	0
Temperature	0	0	0	0.6990	0
Filter	0	0	0	0	1.3979
System	0	0	0	0	0.6990
Process	0	0	0	0	0.6990
WorkPiece	1.3979	0	0	0	0
GrindingWheel	0	0.22218	0.22218	0.22218	0
Coolant	0	0.22218	0.22218	0	0.22218
CoolantNozzle	0	0	0	0.6990	0
FiltrationSystem	0	0	0	0	0.22218

Table 5.5: Concatenated Term-Document Matrix and Concept-Document Matrix

5.3.3. Advantage of Proposed Solution

- *Improved domain knowledge representation:* the addition of domain knowledge represented by ontologies into the semantic space is expected to refine knowledge discovery and knowledge representation helping to maximize precision performance.

5.3.4. Disadvantages of Proposed Solution

- *Extra storage requirement and slow speed:* Representation of ontological annotations introduces an extra storage. Due to the increased size of vector space, the speed of the technique is also slowed.

- *LSA handles a single co-occurrence relationship:* Co-occurrence data occurs in several applications. In these applications many objects occur with possibilities for more than one co-occurrence relationships between objects. However, LSA can only handle a single co-occurrence relationship between two objects e.g. terms and documents in the case of information retrieval. Harnessing of all co-occurrence relationships is likely to result into a better representation of objects in a more unified way in the semantic space.

  In our case above, terms, documents and annotations based on the ontology have been modeled as a single co-occurrence relationship with terms and concepts being merged to form one object. Using LSA, the single co-occurrence relationship between this object and documents was harnessed. This ignored co-occurrence relationship that may exist between terms and concepts. In [44], a more general algorithm, M-LSA, for dealing with multiple-type interrelated data objects has been suggested. Its investigation in future wok is required to represent annotations in the semantic space, i.e. to harness 3 co-occurrence relationships namely: terms-documents, terms-concepts and concepts-documents.

5.4. Summary

Experiments done in this work reveals a need to do more research on how to improve the precision performance of LSA. As one possible strategy, I have proposed the inclusion of annotations into the LSA technique with a newer technique for annotating documents based on ontologies. This generates semantic metadata which is included into the LSA technique. To achieve this, two matrices, *term-document matrix* drawn from the original documents and *concept-document matrix* based on semantic annotations are concatenated.

6. Conclusion and Future Work

6.1. Conclusion

This work has focused on investigating the application of latent semantic analysis in connecting people for knowledge sharing. This follows limitations by knowledge management practice to cope with knowledge needs within organizations. Investigations done during this work reveal lack of understanding into the nature of knowledge despite recognition of its role towards creating organizations' competitive advantages. The unclear nature of knowledge has been propagated into knowledge management practice causing it to suffer from a disconnection between *ideals* and *realization*. *Ideals* are clear in having knowledge management creating and sharing knowledge but the *realization* of knowledge management does not achieve these ideals. The information processing view towards knowledge management is one of the problems characterizing its realization. There has been various computer based information systems that have been implemented with almost close to zero impact other than overwhelming users with information. With organizations spending around 3.5% of their total revenue on knowledge management without desired output, frustrations continue to emerge with implementation processes and users of these systems taking the blame.

Investigation into the nature of knowledge in this work has shown that knowledge has both a *process* and *artifacts* components making it inseparable from the knower. To share knowledge therefore, organizations should focus on connecting people-with-people. Where this is impossible due to location and time differences among other factors, an intermediate step of people-with-documents connection is proposed. Investigation of LSA in achieving people-with-documents connection shows a higher recall performance. However, decreased precision performance at higher recall values is noted. This demands efforts to increase precision while taking advantage of high recall performance.

A sound algorithm for people-with-documents connection should be speedy and accurate in terms of high recall and precision performance. The inclusion of annotations into the LSA process has been proposed to improve precision performance of LSA. The use of ontology to produce semantic metadata has been proposed. To include metadata, a proposal to concatenate *concept-document matrix* with *term-document matrix* was suggested. However, this solution is likely to burden storage facility as well as degrading the speed of LSA further. More work is still needed to find a more principled way of representing document annotations within LSA process without introducing unnecessary drawbacks. Apart from this, other limitations of LSA have to be addressed in future work. LSA remains methodologically incomplete and slow. Firstly, SVD requires huge computation power hence making it slow. Secondly, the selection of the number of dimensions to use in order to eliminate noise and unearth document semantics has to be done heuristically. Use of Frobenius norm to determine the number of dimensions cannot be justified in count data that LSA processes.

6.2. Future Work

The following are some areas requiring attention in future work on knowledge sharing and latent semantic analysis.

6.2.1. Nature of knowledge

For the success of knowledge management and specifically knowledge sharing, further investigation into the concept of knowledge and its representation is required. A starting point should be to make a breakthrough in understanding knowledge from the context of human brain from where it is stored and generated. A breakthrough in understanding the concept of knowledge will also be a recipe for the progress of artificial intelligence.

6.2.2. Completeness of LSA Process

More work should be done to ensure completeness of LSA process. Particularly, a sound algorithm or process for determining the number of dimensions (*k* for dimensionality reduction) to retain should be developed. Similar efforts should be directed towards addressing polysemy which has the potential to negatively impact on people-with-documents connections.

6.2.3. Representation of Annotations into LSA process

While I have proposed the inclusion of annotations into the LSA technique, I heuristically added annotations along dimension *m* in the original vector space model of dimension *m* by *n.* The process was achieved by merging *term-document matrix* with *concept-document matrix* developed from the annotation process. More work is needed to come up with a more principled method to represent annotations on the semantic space. As mentioned earlier, LSA exploits a single co-occurrence relationship between two objects. An investigation of M-LSA to handle multiple co-occurrence relationships is also important. This will address co-occurrence relationships such as between *annotations & documents*, *terms & documents* and *terms & annotations*.

6.2.4. Speed of LSA

Commercial use of LSA remains limited due to its slow speed in executing user queries. A faster and accurate algorithm for performing dimensionality reduction especially to deal with the increased load due to annotations is required. As mentioned in Chapter 4, Brand has introduced a new algorithm for calculating SVD that is faster. This should be investigated further.

References

[1] Ahsan Syed, Shah Abad. *Data, Information, Knowledge, Wisdom: A DoublyLinked Chain?*. Available from <http://ww1.ucmss.com/books/LFS/CSREA2006/IKE4628.pdf> Accessed [01.07.2009]

[2] Al-Khalifa Hend S. *Automatic Document-Level Semantic Metadata Annotation using Folksonomies and Domain Ontologies,* PhD Thesis, School of Electronics and Computer Science, University of Southampton, UK, 2007

[3] Al-Khalifa, H.S and H.C. Davis. The Evolution of Metadata from Standards to Semantics in E-Learning Applications. In Proceedings of Hypertext'06, Odense, Denmark ACM Press

[4] Asimakou Theodora. *The Knowledge Dimension of Innovation Management*, Journal of Knowledge Management Research & Practice, 7, 82-90, 2009

[5] Baskerville Richard and Dulipovici Alina. *The Theoretical Foundations of Knowledge Management*. Journal of Knowledge Management & Practice, 4, 83–105, 2006

[6] Bellegarda R. Jerome, *Latent Semantic Mapping: Principles & Applications*, Morgan & Claypool, 2007

[7] Brand Mathew. *Fast Low-Rank Modifications of the Think Singular Value Decomposition*, Mitsubishi Electric Research Laboratories, 2006.

[8] Canadian International Development Agency. *Knowledge: Methods, Meetings and Tools*. Available from < http://km.fao.org/uploads/media/CIDA_guide_Knowledge.pdf> Accessed [21.10.2009]

[9] Corcho Oscar. *Ontology based document annotation: trends and open research problems.* Int. J. Metadata, Semantics and Ontologies, Vol. 1, No. 1, 2006

[10] Deerwester S., Dumais S.T., Furnas G., Landauer T. & Harshman R. *Indexing by Latent Semantic Analysis*. Journal of American Society for Information Science, 391-407, 1990

[11] Dignum Virginia. *Personalized Support for Knowledge Sharing.* Available from <http://people.cs.uu.nl/virginia/publications/sigchi-vdignum.pdf> Accessed [01.07.2009]

[12] Feeny Luke. Knowledge Management: *The role and limitations of IT.* Institute of Leadership and Healthcare Management. Available from < http://www.ics.ie/psn/presentations/Luke_Feeney.pdf> Accessed [24.09.2009]

[13] Fensel Dieter, Hendler A. James, Lieberman H. & Wahlster W (eds). *Spinning the Semantic Web*. MIT Press, MA, Cambridge, 2003

[14] Flické Martin. *The Knowledge Pyramid: a critique of the DIKW hierarchy*. Journal of Information Science OnlineFirst, 2008

[15] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F.Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[16] Fung M. Benjamin, Wang Ke, Ester M. & Fraser Simon. *Hierarchical Document Clustering*. Available from <http://www.cs.sfu.ca/~ester/papers/Encyclopedia.pdf> Accessed [13.07.2009]

[17] Gong Y. & Liu X. *Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis*. Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, New Orleans, Louisiana, United States 2001, pp. 19-25

[18] Gorrell Genevieve and Webb Brandyn. *Generalized Hebbian Algorithm for Incremental Latent Semantic Analysis*, Department of Information and Computer Science, Linköping University, Sweden

[19] Grigoris Antoniou and Frank van Harmelen. *Web Ontology Language*. Available from <http://www.cs.vu.nl/~frankh/postscript/OntoHandbook03OWL.pdf> Accessed [27.07.2009]

[20] Hofmann Thomas. *Unsupervised Learning by Probabilistic Latent Semantic Analysis*, Machine Learning, 42, 177-196, 2001

[21] Landauer, T.K., Foltz, P.W., & Laham, D. *Introduction to latent Semantic Analysis*. Discourse Processes, 25, 259-284, 1998.

[22] Lausen H., Stollberg M, Hernández R.L, Ding Y, Han S-K and Fensel D**.** *Semantic Web Portals – State of the Art Survey.* Next Web Generation Group, Austria

[23] Manning D. Christopher, Raghavan Prabhakar and Schütze Hinrich, *Introduction to Information Retrieval*, Cambridge University Press, New York, 2008

[24] Marinov Milko. *Using XML to Represent Knowledge by Frames*. In proceedings of International Conference on Computer Systems and Technologies-CompSysTech, 2004

[25] Mclnerney Claire. *Knowledge Management and the Dynamic Nature of Knowledge*. Journal of the American Society for Information Science and Technology, 53(12):1009-1018, 2002

[26] Minsky Marvin. *A Framework for Representing Knowledge*. MIT-AI Laboratory Memo 306, June, 1974

[27] Ngah Rohana and Jusoff Kamaruzaman. *Tacit Knowledge Sharing and SME'S Organizational Performance*. International Journal of Economics and Finance, Vol1, No1, February 2009.

[28] Noy N.F and McGuinness D.L. *Ontology Development 101: A guide to creating Your First Ontology*. Stanford University, Stanford, USA.

[29] Ontomat Homepage. Available from < http://annotation.semanticweb.org/ontomat/index.html> Accessed [10.09.2009]

[30] Pham T-T, Maillot N, Lim J-H, Chevallet. *Latent Semantic Fusion Model for Image Retrieval and Annotation*. Institute for Inforcomm Research, Singapore

[31] Robert Charles Abiodun. *AMIEDoT: An annotation model for document tracking and recommendation service*. International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering-CIS2E 2006, Bridgeport, USA, 2006.

[32] Rosario Barbara, *Latent Semantic Indexing: An overview*, INFOSYS 240 Spring 2000 Final Paper, 2000

[33] Rosendaal Bastiaan. *Sharing knowledge, being different and working as a team Guarino*. Journal of Knowledge Management & Practice, 7, 4–14, 2009

[34] Russell Bertrand. *Knowledge and Wisdom*. Available from < http://www.awayward.com/library/Philosophy/BertrandRussell/Russell,%20Bertrand%20-%20Knowledge%20And%20Wisdom.pdf > Accessed [01.12.2009]

[35] Sachs,O., Weis, Krings, Huber & Kircher. *Categorical and Thematic knowledge representation in the brain: Neural correlates of taxonomic and thematic conceptual relations*. Neuropsychologia 46, 409-418, 2008

[36] Sanchez, Ron. *Approaches to Knowledge Management Practice*. Available from < http://www.knowledgeboard.com/download/3512/Tacit-vs-Explicit.pdf> Accessed [01.07.2009]

[37] Schneider K.-M. *A comparison of event models for naïve Bayes anti-spam e-mail filtering,* in the proceedings of 11th conf. Eur. Chap. ACL, Budapest, Hungary, 2003

[38] Simpson John & Weiner Edmund (eds). *The Oxford English Dictionary*, Oxford University Press, USA, 2nd edition, March 30, 1989

[39] Soshnikov D. *An Architecture of Distributed Frame Hierarchy for Knowledge Sharing and Reuse in Computer Networks*. In Proceedings of the International Conference on Artificial Intelligence Systems, IEEE Computer Society Press, 115-119, 2002.

[40] Sowa F. John. *Semantic Networks*. Available from < http://www.biozentrum.unibas.ch/~schwede/Teaching/BixII-SS05/Semantic_Networks.pdf> Accessed [03.08.2009]

[41] Steinberger Josef and Ježek Karel. *Using Latent Semantic Analysis in Text Summarization and Summary evaluation*. Available from < http://textmining.zcu.cz/publications/isim.pdf> Accessed [23.07.2009]

[42] Studer Rudi and Sure York. *On-To-Knowledge: Content-Driven Knowledge Management Tools through Evolving Ontologies*. EU-IST Project IST-1999-10132 On-To-Knowledge, 2002

[43] Vargas-Vera M., Motta E., Domingue J., Lanzoni M., Stutt A. & Ciravegna F. *MnM: Ontology-Driven Tool for Semantic Markup*. The Open University, Uk. Available from < http://projects.kmi.open.ac.uk/akt/publication-pdf/vargas-saakm02.pdf > Accessed [11.09.2009]

[44] Wang X, Sun J-T, Chen Z & Zhai C. *Latent Semantic Analysis for Multiple-Type Interrelated Data Objects*. Pages 236-243, 2006

[45] Wang Yingxu. *The OAR model of Neural Informatics for Internal Knowledge Representation in the Brain*. Journal of Cognitive Informatics and Natural Intelligence, 1(3), 66-77, 2008

[46] Wikipedia, the free encyclopedia. 2009.

[47] Wilson F.A. *Computer-Based Information Systems and Knowledge Management: Contrasting the Objectivist and Subjectivist Perspectives*, Information Systems Institute, University of Salford UK, Available at <http://www.pacis-net.org/file/2005/154.pdf> Accessed [09.10.2009]

[48] Zhu Zhichang. *Knowledge, Knowing, Knower: what is to be managed and does it matter?* Journal of Knowledge Management Research & Practice, 6, 112-123, 2008

Index

Section of MATLAB Code Used

```
%% Input documents

n = 0;
D = cell(0);
while 1
  filename = ['documents/'  int2str(n + 1) '.txt'];
  fid = fopen(filename, 'r');%...open file for reading
  if (fid == -1), break, end
  n = n + 1;
  D{n} = (fread(fid, '*char'))';
  fclose(fid);
  totalDocs=n;
end

%% Simple document preprocessing
for j = 1:n
  D{j} = lower(D{j});  %...converts string read from files into lower
case
end

%% Removal of stop words
C = textread('stop words/stopwords.txt', '%s'); %...Reading the stop
lists
j=length(C);
i=0; %...Keeps track of documents
P=0;
while (i<n)
  docname=['documents/'  int2str(i + 1) '.txt'];
  temp=0;
  W= lower(textread(docname, '%s'));
  P=cat(1,P,setdiff(W,C));
 i=i+1;
end
P(1) = [];
Dict=unique(P);
i=length(Dict);
save 'dictionary/dictionary.mat' Dict;

%% Input terms
m =length(Dict);
%%Dictionary Statistics
fprintf(1, '%s \n','DICTIONARY TERMS');
for j=1:m
fprintf(1,'%d%s%s\n ', j, ' : ', Dict{j});
end
```

```
%% Occurrence matrix using raw term frequency

for i = 1:m
 for j = 1:n
   A(i, j) = size(findstr(D{j}, Dict{i}), 2); %...Construction of Term-
Document Matrix based on raw term frequency
 end
end

%% Using tf-idf weighting

%{
tf-idf
idf= log N/df
df=0;
mLen= length(Dict);
temp1=0;
j=0;

for i= 1:mLen
   for j=1:totalDocs
    doc=['documents/'  int2str(j) '.txt'];
    docTerms = lower(textread(doc, '%s'));
    docTermsLen=length(docTerms);
    for kk=1:docTermsLen
      if (strcmp(Dict(i),docTerms(kk))>0)
       temp1=temp1+1;
      end
    end
    if (temp1 >0)
        df=df+1;
        temp1=0;
    end
    A(i, j) = size(findstr(D{j}, Dict{i}), 2) * log(totalDocs*df);
   end

   df=0;
end
%}

%% Normalization process

for j = 1:n
  A(:, j) = A(:, j) / norm(A(:, j));
end


%% Dimension reduction

[U,S,V] = svd(A);
Vt=V';
k =10 ;
Sk = S .* [ones(m, k) zeros(m, n-k)]; % retain k singular values
Ak = U * Sk * V';
```

```
%saving results

save 'matrices/A.mat' A;
save 'matrices/U.mat' U;
save 'matrices/S.mat' S;
save 'matrices/Vtranspose.mat' Vt;
save 'matrices/Ak.mat' Ak;

%% Document retrieval

%To search assign the term index to search_request var
search_request =456;
q = zeros(m, 1);
q(search_request) = 1;
r = (q'*Ak)';
[sr, id] = sort(r);
 fprintf(1, '\n                                    search request:
"%s"\n\n', Dict{search_request});

 fprintf(1, '%s %s \n', 'Score','
Documents');
 fprintf(1, '%s
\n','_____________________________________________________________________
_____________________________________________________________________
_____________________________________________________________________
_____________________');

 for j = n:-1:1
      if (sr(j)>0)
        fprintf(1, '%5.6f  %s\n', sr(j), D{id(j)})
      end
  end
```

Section of Ontology Developed Using OntoMat-Annotizer

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY grinding-ontology "http://www.grindaix.de/grinding-
ontology.owl#" >
    <!ENTITY grinding-ontology2 "http://www.grindaix.de/grinding-
ontology.owl#100" >
]>

<rdf:RDF xmlns="http://www.grindaix.de/grinding-ontology.owl#"
     xml:base="http://www.grindaix.de/grinding-ontology.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:grinding-ontology2="&grinding-ontology;100"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:grinding-ontology="http://www.grindaix.de/grinding-
ontology.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about="">
        <rdfs:comment
            >This is an example of Grinding Ontology developed within
Grindaix GmbH </rdfs:comment>
    </owl:Ontology>



    <!--

    // Object Properties

 -->




    <!-- http://www.grindaix.de/grinding-ontology.owl#Cooling -->
    <owl:ObjectProperty rdf:about="#Cooling">
        <rdfs:domain rdf:resource="#Coolant"/>
        <rdfs:range>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Machine"/>
                    <rdf:Description rdf:about="#Material"/>
                    <rdf:Description rdf:about="#WorkPiece"/>
                </owl:intersectionOf>
            </owl:Class>
        </rdfs:range>
    </owl:ObjectProperty>
```

```
<!-- http://www.grindaix.de/grinding-ontology.owl#Dosing -->
<owl:ObjectProperty rdf:about="#Dosing">
    <rdfs:range rdf:resource="#Coolant"/>
    <rdfs:domain rdf:resource="#CoolantNozzle"/>
    <rdfs:domain rdf:resource="#DressingTool"/>
</owl:ObjectProperty>


<!-- http://www.grindaix.de/grinding-ontology.owl#Dressing -->

<owl:ObjectProperty rdf:about="#Dressing">
    <rdfs:domain rdf:resource="#DressingTool"/>
    <rdfs:range rdf:resource="#Material"/>
</owl:ObjectProperty>



<!-- http://www.grindaix.de/grinding-ontology.owl#Filtering -->

<owl:ObjectProperty rdf:about="#Filtering">
    <rdfs:domain rdf:resource="#FiltrationSystem"/>
    <rdfs:range>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#Coolant"/>
                <rdf:Description rdf:about="#Material"/>
            </owl:intersectionOf>
        </owl:Class>
    </rdfs:range>
</owl:ObjectProperty>

<!-- http://www.grindaix.de/grinding-ontology.owl#Grinding -->

<owl:ObjectProperty rdf:about="#Grinding">
    <rdfs:domain rdf:resource="#GrindingWheel"/>
    <rdfs:range rdf:resource="#WorkPiece"/>
</owl:ObjectProperty>

<!-- http://www.grindaix.de/grinding-ontology.owl#Rotating -->

<owl:ObjectProperty rdf:about="#Rotating">
    <rdfs:range rdf:resource="#Material"/>
    <rdfs:domain rdf:resource="#Wheel"/>
</owl:ObjectProperty>


<!-- http://www.grindaix.de/grinding-ontology.owl#Supplying -->

<owl:ObjectProperty rdf:about="#Supplying">
    <rdfs:range>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#Coolant"/>
                <rdf:Description rdf:about="#Material"/>
            </owl:intersectionOf>
        </owl:Class>
```

```
        </rdfs:range>
        <rdfs:domain>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#CoolantNozzle"/>
                    <rdf:Description rdf:about="#DressingTool"/>
                    <rdf:Description rdf:about="#FiltrationSystem"/>
                </owl:intersectionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:ObjectProperty>



    <!--
    // Data properties

   -->

    <!-- http://www.grindaix.de/grinding-ontology.owl#hasAngle -->

    <owl:DatatypeProperty rdf:about="#hasAngle">
        <rdfs:domain rdf:resource="#Machine"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </owl:DatatypeProperty>


    <!-- http://www.grindaix.de/grinding-ontology.owl#hasAxialFeedRate
-->

    <owl:DatatypeProperty rdf:about="#hasAxialFeedRate">
        <rdfs:domain rdf:resource="#GrindingWheel"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </owl:DatatypeProperty>

    <!-- http://www.grindaix.de/grinding-ontology.owl#hasBoreDiameter -
->

    <owl:DatatypeProperty rdf:about="#hasBoreDiameter">
        <rdfs:domain rdf:resource="#GrindingWheel"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </owl:DatatypeProperty>


    <!-- http://www.grindaix.de/grinding-
ontology.owl#hasCircumferentialSpeed -->

    <owl:DatatypeProperty rdf:about="#hasCircumferentialSpeed">
        <rdfs:domain rdf:resource="#Wheel"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </owl:DatatypeProperty>

    <!-- http://www.grindaix.de/grinding-
ontology.owl#hasConstructionNumber -->

    <owl:DatatypeProperty rdf:about="#hasConstructionNumber">
        <rdfs:domain rdf:resource="#Machine"/>
```

```
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>


    <!-- http://www.grindaix.de/grinding-ontology.owl#hasDiameter -->

    <owl:DatatypeProperty rdf:about="#hasDiameter">
        <rdfs:range rdf:resource="&xsd;double"/>
        <rdfs:domain>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#CoolantNozzle"/>
                    <rdf:Description rdf:about="#DressingTool"/>
                    <rdf:Description rdf:about="#Wheel"/>
                </owl:intersectionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:DatatypeProperty>

    <!-- http://www.grindaix.de/grinding-ontology.owl#hasFlowRate -->

    <owl:DatatypeProperty rdf:about="#hasFlowRate">
        <rdfs:range rdf:resource="&xsd;double"/>
        <rdfs:domain>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Coolant"/>
                    <rdf:Description rdf:about="#CoolantNozzle"/>
                </owl:intersectionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-ontology.owl#hasGrindingTime -
->

    <owl:DatatypeProperty rdf:about="#hasGrindingTime">
        <rdfs:domain rdf:resource="#GrindingWheel"/>
        <rdfs:range rdf:resource="&xsd;duration"/>
    </owl:DatatypeProperty>

    <!-- http://www.grindaix.de/grinding-ontology.owl#hasLength -->

    <owl:DatatypeProperty rdf:about="#hasLength">
        <rdfs:domain rdf:resource="#WorkPiece"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-ontology.owl#hasMaxDiameter --
>

    <owl:DatatypeProperty rdf:about="#hasMaxDiameter">
        <rdfs:range rdf:resource="&xsd;double"/>
```

```
        <rdfs:domain>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#CoolantNozzle"/>
                    <rdf:Description rdf:about="#Wheel"/>
                </owl:intersectionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-
ontology.owl#hasMaxRotationalSpeed -->

    <owl:DatatypeProperty rdf:about="#hasMaxRotationalSpeed">
        <rdfs:domain rdf:resource="#Wheel"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-
ontology.owl#hasMaxVolumeFlowRate -->

    <owl:DatatypeProperty rdf:about="#hasMaxVolumeFlowRate">
        <rdfs:domain rdf:resource="#CoolantNozzle"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </owl:DatatypeProperty>


    <!-- http://www.grindaix.de/grinding-
ontology.owl#hasMaxWorkPieceWidth -->

    <owl:DatatypeProperty rdf:about="#hasMaxWorkPieceWidth">
        <rdfs:domain rdf:resource="#WorkPiece"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-
ontology.owl#hasMaxWorkpieceDiameter -->

    <owl:DatatypeProperty rdf:about="#hasMaxWorkpieceDiameter">
        <rdfs:domain rdf:resource="#WorkPiece"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-
ontology.owl#hasMaxWorkpieceHeight -->

    <owl:DatatypeProperty rdf:about="#hasMaxWorkpieceHeight">
        <rdfs:domain rdf:resource="#WorkPiece"/>
        <rdfs:range rdf:resource="&xsd;double"/>
```

```
    </owl:DatatypeProperty>
  <!-- http://www.grindaix.de/grinding-
ontology.owl#hasMaxWorkpieceLength -->

    <owl:DatatypeProperty rdf:about="#hasMaxWorkpieceLength">
        <rdfs:domain rdf:resource="#WorkPiece"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-
ontology.owl#hasMinVolumeFlowRate -->

    <owl:DatatypeProperty rdf:about="#hasMinVolumeFlowRate">
        <rdfs:range rdf:resource="&xsd;double"/>
        <rdfs:domain>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#DressingTool"/>
                    <rdf:Description rdf:about="#FiltrationSystem"/>
                    <rdf:Description rdf:about="#CoolantNozzle"/>
                </owl:intersectionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-
ontology.owl#hasMinWorkPieceLength -->

    <owl:DatatypeProperty rdf:about="#hasMinWorkPieceLength">
        <rdfs:domain rdf:resource="#WorkPiece"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-
ontology.owl#hasMinWorkpieceDiameter -->

    <owl:DatatypeProperty rdf:about="#hasMinWorkpieceDiameter">
        <rdfs:domain rdf:resource="#WorkPiece"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-ontology.owl#hasModel -->

    <owl:DatatypeProperty rdf:about="#hasModel">
        <rdfs:range rdf:resource="&xsd;string"/>
        <rdfs:domain>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#CoolantNozzle"/>
```

```
                    <rdf:Description rdf:about="#FiltrationSystem"/>
                    <rdf:Description rdf:about="#Machine"/>
                </owl:intersectionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-ontology.owl#hasNominalPower -
->

    <owl:DatatypeProperty rdf:about="#hasNominalPower">
        <rdfs:domain rdf:resource="#Machine"/>
        <rdfs:range rdf:resource="&xsd;positiveInteger"/>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-
ontology.owl#hasRotationalSpeed -->

    <owl:DatatypeProperty rdf:about="#hasRotationalSpeed">
        <rdfs:domain rdf:resource="#Wheel"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-ontology.owl#hasSpeed -->

    <owl:DatatypeProperty rdf:about="#hasSpeed">
        <rdfs:domain rdf:resource="#Wheel"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-ontology.owl#hasThickness -->

    <owl:DatatypeProperty rdf:about="#hasThickness">
        <rdfs:range rdf:resource="&xsd;double"/>
        <rdfs:domain>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Material"/>
                    <rdf:Description rdf:about="#WorkPiece"/>
                </owl:intersectionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-ontology.owl#hasTotalInfeed --
>
```

```
    <owl:DatatypeProperty rdf:about="#hasTotalInfeed">
        <rdfs:range rdf:resource="&xsd;double"/>
        <rdfs:domain>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#CoolantNozzle"/>
                    <rdf:Description rdf:about="#DressingTool"/>
                    <rdf:Description rdf:about="#FiltrationSystem"/>
                </owl:intersectionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-ontology.owl#hasViscosity -->

    <owl:DatatypeProperty rdf:about="#hasViscosity">
        <rdfs:domain rdf:resource="#Coolant"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-
ontology.owl#hasYearOfManufacture -->

    <owl:DatatypeProperty rdf:about="#hasYearOfManufacture">
        <rdfs:range rdf:resource="&xsd;gYear"/>
        <rdfs:domain>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#CoolantNozzle"/>
                    <rdf:Description rdf:about="#FiltrationSystem"/>
                    <rdf:Description rdf:about="#Machine"/>
                    <rdf:Description rdf:about="#Material"/>
                </owl:intersectionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-ontology.owl#isCDCapable -->

    <owl:DatatypeProperty rdf:about="#isCDCapable">
        <rdfs:domain rdf:resource="#Material"/>
        <rdfs:range rdf:resource="&xsd;boolean"/>
    </owl:DatatypeProperty>



    <!-- http://www.grindaix.de/grinding-ontology.owl#iscBNCapable -->

    <owl:DatatypeProperty rdf:about="#iscBNCapable">
        <rdfs:domain rdf:resource="#Material"/>
        <rdfs:range rdf:resource="&xsd;boolean"/>
```

```
    </owl:DatatypeProperty>
     <!--

//////////////////////////////////////////////////////////////////////
////////////////
    //
    // Classes
    //

//////////////////////////////////////////////////////////////////////
////////////////
     -->



    <!-- http://www.grindaix.de/grinding-ontology.owl#BandFilter -->

    <owl:Class rdf:about="#BandFilter">
        <rdfs:subClassOf rdf:resource="#FiltrationSystem"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#BearingSteel -->

    <owl:Class rdf:about="#BearingSteel">
        <rdfs:subClassOf rdf:resource="#Material"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#BoringMillMachine
-->

    <owl:Class rdf:about="#BoringMillMachine">
        <rdfs:subClassOf rdf:resource="#RotatingMachine"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#CBN -->

    <owl:Class rdf:about="#CBN">
        <rdfs:subClassOf rdf:resource="#GrainMaterial"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#Coolant -->

    <owl:Class rdf:about="#Coolant"/>



    <!-- http://www.grindaix.de/grinding-ontology.owl#CoolantNozzle -->
```

```
    <owl:Class rdf:about="#CoolantNozzle"/>
    <!-- http://www.grindaix.de/grinding-ontology.owl#Corundum -->

    <owl:Class rdf:about="#Corundum">
        <rdfs:subClassOf rdf:resource="#GrainMaterial"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#CupWheel -->

    <owl:Class rdf:about="#CupWheel">
        <rdfs:subClassOf rdf:resource="#DressingTool"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#Diamond -->

    <owl:Class rdf:about="#Diamond">
        <rdfs:subClassOf rdf:resource="#GrainMaterial"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#DressingTile -->

    <owl:Class rdf:about="#DressingTile">
        <rdfs:subClassOf rdf:resource="#DressingTool"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#DressingTool -->

    <owl:Class rdf:about="#DressingTool"/>



    <!-- http://www.grindaix.de/grinding-ontology.owl#DrillMachine -->

    <owl:Class rdf:about="#DrillMachine">
        <rdfs:subClassOf rdf:resource="#RotatingMachine"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#Emulsion -->

    <owl:Class rdf:about="#Emulsion">
        <rdfs:subClassOf rdf:resource="#Coolant"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#FiltrationSystem
-->
```

```
    <owl:Class rdf:about="#FiltrationSystem"/>



    <!-- http://www.grindaix.de/grinding-ontology.owl#FormRoller -->

    <owl:Class rdf:about="#FormRoller">
        <rdfs:subClassOf rdf:resource="#DressingTool"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#FreeFlowNozzle --
>

    <owl:Class rdf:about="#FreeFlowNozzle">
        <rdfs:subClassOf rdf:resource="#CoolantNozzle"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#GrainMaterial -->

    <owl:Class rdf:about="#GrainMaterial">
        <rdfs:subClassOf rdf:resource="#Material"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#GrindingMachine -
->

    <owl:Class rdf:about="#GrindingMachine">
        <rdfs:subClassOf rdf:resource="#RotatingMachine"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#GrindingWheel -->

    <owl:Class rdf:about="#GrindingWheel">
        <rdfs:subClassOf rdf:resource="#Wheel"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#Hydrocyclone -->

    <owl:Class rdf:about="#Hydrocyclone">
        <rdfs:subClassOf rdf:resource="#FiltrationSystem"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#LatheMachine -->
```

```
    <owl:Class rdf:about="#LatheMachine">
        <rdfs:subClassOf rdf:resource="#RotatingMachine"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#Machine -->

    <owl:Class rdf:about="#Machine"/>



    <!-- http://www.grindaix.de/grinding-ontology.owl#MagneticSeparator
-->

    <owl:Class rdf:about="#MagneticSeparator">
        <rdfs:subClassOf rdf:resource="#FiltrationSystem"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#Material -->

    <owl:Class rdf:about="#Material"/>



    <!-- http://www.grindaix.de/grinding-ontology.owl#MetalicWorkPiece
-->

    <owl:Class rdf:about="#MetalicWorkPiece">
        <rdfs:subClassOf rdf:resource="#WorkPiece"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#MultiGrainDresser
-->

    <owl:Class rdf:about="#MultiGrainDresser">
        <rdfs:subClassOf rdf:resource="#DressingTool"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#NeedleNozzle -->

    <owl:Class rdf:about="#NeedleNozzle">
        <rdfs:subClassOf rdf:resource="#CoolantNozzle"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#Oil -->

    <owl:Class rdf:about="#Oil">
        <rdfs:subClassOf rdf:resource="#Coolant"/>
```

```
    </owl:Class>

    <!-- http://www.grindaix.de/grinding-ontology.owl#PlanerMachine -->

    <owl:Class rdf:about="#PlanerMachine">
        <rdfs:subClassOf rdf:resource="#ReciprocatingMachine"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#ProfileRoller -->

    <owl:Class rdf:about="#ProfileRoller">
        <rdfs:subClassOf rdf:resource="#DressingTool"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-
ontology.owl#ReciprocatingMachine -->

    <owl:Class rdf:about="#ReciprocatingMachine">
        <rdfs:subClassOf rdf:resource="#Machine"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#RotatingMachine -
->

    <owl:Class rdf:about="#RotatingMachine">
        <rdfs:subClassOf rdf:resource="#Machine"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#ShaperMachine -->

    <owl:Class rdf:about="#ShaperMachine">
        <rdfs:subClassOf rdf:resource="#ReciprocatingMachine"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#ShoeNozzle -->

    <owl:Class rdf:about="#ShoeNozzle">
        <rdfs:subClassOf rdf:resource="#CoolantNozzle"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#SiliconCarbide --
>

    <owl:Class rdf:about="#SiliconCarbide">
        <rdfs:subClassOf rdf:resource="#GrainMaterial"/>
```

```
    </owl:Class>
    <!-- http://www.grindaix.de/grinding-
ontology.owl#SingleGrainDresser -->

    <owl:Class rdf:about="#SingleGrainDresser">
        <rdfs:subClassOf rdf:resource="#DressingTool"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#SlotterMachine --
>

    <owl:Class rdf:about="#SlotterMachine">
        <rdfs:subClassOf rdf:resource="#ReciprocatingMachine"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-
ontology.owl#WaterBasedSolution -->

    <owl:Class rdf:about="#WaterBasedSolution">
        <rdfs:subClassOf rdf:resource="#Coolant"/>
    </owl:Class>



    <!-- http://www.grindaix.de/grinding-ontology.owl#Wheel -->

    <owl:Class rdf:about="#Wheel"/>


    <!-- http://www.grindaix.de/grinding-ontology.owl#WorkPiece -->

    <owl:Class rdf:about="#WorkPiece"/>
</rdf:RDF>


<!-- Generated by the OWL API (version 2.2.1.1138)
http://owlapi.sourceforge.net -->
```