TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

DeepKAF: A Knowledge Intensive Framework for Heterogeneous Case-Based Reasoning in Textual Domains

Thesis approved by the Department of Computer Science
Technische Universität Kaiserslautern for the award of the
Doctoral Degree Doctor of
Engineering (Dr.-Ing.)

to

Kareem Amin

Date of Defense: 07th September 2021
Dean: Prof. Dr. Jens Schmitt
Reviewer: Prof. Dr. Andreas Dengel
Reviewer: Prof. Dr. Klaus-Dieter Althoff
Reviewer: Prof. Dr. Juan Antonio Recio García

D 386

ACKNOWLEDGMENTS

First and foremost, I have to thank my parents for their undivided love they gave me. I would have never been writing this thesis now without their hard work throughout many years. My brother and role-model, sister and second mother deserve my wholehearted thanks as well.

It is my deepest gratitude and warmest affection to dedicate this thesis to my supervisor Prof. Althoff who has been a constant source of scientific knowledge and inspiration. Thanks for the guidance and emotional support he gave me during this study.

Also, no words can express my gratitude to the coincidence that introduced me to my mentor who became my dearest friend, Dr. Kapetanakis. Dr. Kapetanakis gave me the confidence and help in all aspects around my thesis and played a major role to get to this stage of my study.

Thanks go to Prof. Dengel as well for giving me a unique opportunity to be part of his research group which gave me the opportunity to learn a lot from him and my colleagues.

Special thanks to my friends, Doaa & Abdel-Meguid for their help, support, and believing in me and my goals since I was pursuing my bachelors degree till now. It is a gift to have such persons like you in my life.

Thank Allah for the strength he gave me throughout every day that helped me to attain the goal I have been aspiring for.

This thesis is only a start to a very long journey!

ABSTRACT

Business-relevant domain knowledge can be found in plain text across message exchanges among customer support tickets, employee message exchanges and other business transactions. Decoding text-based domain knowledge can be a very demanding task since traditional methods focus on a comprehensive representation of the business and its relevant paths. Such a process can be highly complex, time-costly and of high maintenance effort, especially in environments that change dynamically.

In this thesis, a novel approach is presented for developing hybrid case-based reasoning (CBR) systems that bring together the benefits of deep learning approaches with CBR advantages. Deep Knowledge Acquisition Framework (**DeepKAF**) is a domain-independent framework that features the usage of deep neural networks and big data technologies to decode the domain knowledge with the minimum involvement from the domain experts. While this thesis is focusing more on the textual data because of the availability of the datasets, the target CBR systems based on **DeepKAF** are able to deal with heterogeneous data where a case can be represented by different attribute types and automatically extract the necessary domain knowledge while keeping the ability to provide an adequate level of explainability. The main focus within this thesis are automatic knowledge acquisition, building similarity measures and cases retrieval.

Throughout the progress of this research, several sets of experiments have been conducted and validated by domain experts. Past textual data produced over around 15 years have been used for the needs of the conducted experiments. The text produced is a mixture between English and German texts that were used to describe specific domain problems with a lot of abbreviations. Based on these, the necessary knowledge repositories were built and used afterwards in order to evaluate the suggested approach towards effective monitoring and diagnosis of business workflows. Another public dataset has been used, the CaseLaw dataset, to validate **DeepKAF** when dealing with longer text and cases with more attributes.

The CaseLaw dataset represents around 22 million cases from different US states.

Further work motivated by this thesis could investigate how different deep learning models can be used within the CBR paradigm to solve some of the chronic CBR challenges and be of benefit to large-scale multi-dimensional enterprises.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
BPM	Business Process Management
CCC	Computer Cooking Competition
CER	Comparative Effectiveness Research
CNN	Convolutional Neural Network
DeepKAF	Deep Knowledge Acquisition Framework
DeepTMS	Deep Ticket Management System
DNN	Deep Neural Networks
HRR	Holographic Reduced Representations
IoT	Internet of Things
IR	Information Retrieval
IML	Interactive Machine Learning
KRL	Knowledge Repository Layer
LR	Logistic Regression
LSTM	Long Short-Term Memory
MSE	Mean Squared Error
MAS	Multi-Agent Systems
NLP	Natural Language Processing
NN	Neural Network
PMML	Predictive Model Markup Language
PST	Power-set Tree
RBES	Rule-Based Expert Systems

RL	Reinforcement Learning
RNN	Recurrent Neural Network
RTS	Real-time Strategy
SBR	Similarity-based Retrieval
SNN	Siamese Neural Network
SVM	Support Vector Machines
TCBR	Textual Case-based Reasoning
TRG	Text Reasoning Graph
WIMS	Workflow Intelligent Monitoring System

Dedication

This dissertation is dedicated to all people who intentionally and unintentionally supported me. Any "I" that is going to be mentioned in this thesis hides so many "We". Thank you very much to all my supporters.

Chapter One

Introduction & Motivation

"Case-based reasoning is a methodology, not a technology" I. Watson, 1999. Case-based reasoning (CBR) is a way of thinking, imitating how the human brain works naturally. Every person has a unique way of thinking and reasoning by recalling memories and relating them to the present. Problems or challenges can be solved similarly regardless of the domain, for example, whether they are in medicine, law or a specialized technical field. Similar situations from the past are recalled and "adapted" to find an appropriate solution to the experienced case and its context. This ability, combined with the knowledge of a plethora of cases usually obtained from a long professional life, defines the human experience. CBR is an artificial intelligence technique and methodology that can provide a system with the ability to simulate exactly the aforementioned behavior (Aamodt and E. Plaza, 1994). For a newly emerging problem, the most similar known problem from the "empirical knowledge" - the so-called case base - is sought. CBR then applies a solution analogous to the known case to the new problem. Then, like a human expert would do, each new problem together with its successful solution is added to the case base, so that future problem-solving would be based on an even greater experience pool and thus a future problem could converge to a solution faster.

CBR is efficient in building structured procedures to simulate typical human behaviors. However, it is prone to failures in text domains if the case representation is not robust

enough to support its operations (cases representation, retrieval, adaptation and similarity assessment). Therefore, finding an ideal textual case representation is the top priority for a CBR system. The choice of an ideal representation can be "extracted" from the domain characteristics and the complexity of its cases, an area where traditional text representation and machine learning techniques struggle to cope with. Recently, the explosion of deep learning techniques and other forms of vectorized representations have provided a new source for case insights. Richer text features can be extracted and used for each case if required, indicating a potential solution to a long-standing problem.

This thesis presents a novel framework (Deep Knowledge Acquisition Framework (**DeepKAF**)) based on recent work in the area of combining deep learning approaches with the CBR paradigm to generate richer case representations and automatically acquiring domain knowledge from unstructured multilingual textual data.

This thesis describes how hybrid CBR systems can obtain their representation vectors from stemmed words and improve these vectors iteratively, suggesting high quality outputs and relevance to domain experts based on the available past knowledge. For the needs of the evaluation of this research extensive work has been done in the area of automotive engineering conducting several sets of experiments and using a combination of different approaches. For generalization purposes, the legal domain was chosen in addition to demonstrate how **DeepKAF** performs in long, complex and hard-to-read documents. The results from both domains show an enhanced CBR approach that is more efficient compared to existing implementations in the literature, whereas still is able to provide substantial explainability compared to pure deep learning approaches. The evaluation of this work shows that CBR can be applied efficiently to complex textual challenges that were not easy to tackle prior to this work. Using **DeepKAF**, highly usable text representation and similarity measures were built automatically with minimum efforts from the experts. This has been based on the assumption that the provided cases are accompanied by the necessary past experience given by domain experts. These experiments along with their produced results will be thoroughly

discussed throughout this thesis.

This work investigates whether there can be a sufficiently generic methodology to dynamically inject deep learning models into the CBR paradigm without affecting the explainability advantage that CBR systems have. This approach along with its architecture can be generalized and integrated with other non-CBR systems without affecting their structural integrity as well as their functionality.

1.1 Rationale - CBR Applications in the Industrial Domain

In the course of obtaining my master's degree, I worked on building an indexing algorithm demonstrating that retrieval processes can be improved substantially (400%) compared to traditional retrieval process within a case base of 300,000 cases (El-Bahnasy, Amin, and Aref, 2014). However, demonstrable challenges were identified during the implementation and motivation behind the indexing algorithm. A prominent challenge emerging from that work was to be able to scale-up traditional CBR systems by indexing cases. Further to scaling, prominent Artificial Intelligence (AI) solutions like neural networks are considered black-box approaches since they cannot explain their results in a human-readable approach. Compared to theses, CBR has the advantage of being able to provide explainability. Later on, when I started my PhD studies, I spotted the gap that AI black-box approaches have and the advantage that CBR is having, namely **explainability**. I started to think differently in the way of how to handle large amounts of data and being able to answer queries in real-time by building distributed and large-scale CBR systems. Motivated by the findings and challenges from (El-Bahnasy, Amin, and Aref, 2014) the work presented in this thesis presents a different approach in the way of how to handle large amounts of data and being able to query and explain findings in real-time by building distributed and large-scale CBR

systems.

The growth of intensive data-driven decision-making has led to extensive use of AI technologies, (Brynjolfsson and McElheran, 2016) since they seem capable to augment it further. CBR has several examples where data-driven methods can improve business decision-making dynamically. Following the business needs, industries that deal with customer experience (e.g., customer support) have also adopted data-centric approaches to optimize their business workflows and the quality of their services(Brynjolfsson and McElheran, 2016).

Customer support relies on teams of qualified experts and technicians who define, monitor and decide what is the best action / re-action to a problem and what should be avoided. Their role is to pin down with accuracy how tasks, products and company resources should be used to achieve the organization goals. Customer support relies on intelligent systems and data to provide high quality services as well as identify potential bottlenecks, improve processes and rationalize complex cases.

CBR provides a variety of options and architectures to implement applications. That is the reason it holds a prominent position within the intelligent systems sector and has been used by several organizations. Since "reasoning" is a top-sought feature and AI applications tend to **XAI (Explainable AI)** (Arrieta et al., 2020) CBR is a high profile solution since it offers explainability as its core feature out-of-the-box.

CBR is regarded a bridge to machine learning [automated knowledge generation] and knowledge-based systems [manual and semi-automatic knowledge modelling]. As such, it is regarded a natural candidate for finding domain or task-specific approaches to integrate automated knowledge generation [using machine learning] with manual knowledge modeling [using knowledge-intensive CBR]. The following list shows potential application areas on which CBR can have a substantial impact, provided there can be a way to handle the challenges of dealing with big data.

1. Internet of Things (IoT) applications (autonomous cars).

2. Ticketing systems and customer support applications.

3. Medical diagnosis systems (from images or text).

4. Server logs anomaly detection applications.

5. Condition monitoring applications like what most oil and gas companies have.

This work is motivated by the opportunities CBR offers to industrial domains and the challenges it experiences when applied to real applications. CBR is highly applicable to novel applications, however limited when being confronted with real industry requirements. This work investigates how CBR can be enhanced and augmented to be a solid methodology for high intensity tasks in the area of Natural Language Processing (NLP). This work intends to be a reference point for future research in the areas of: CBR, natural language processing and big data. It provides enhanced evaluation in two different industry domains, which can be regarded a further contribution of this thesis. Motivated by the current CBR limitations, the following section will present the research questions this work is addressing.

1.2 Research Questions

This work investigates whether: "do we need to understand the text before processing it?", or "whether is it possible to process and understand text with minimal understanding". The research questions that are going to be answered in the next chapters are:

1. Can we have automatic knowledge acquisition from fuzzy, incomplete and unstructured knowledge?

2. Can we achieve sufficient explainability using knowledge-intensive techniques for heterogeneous textual domains?

3. Can we automatically decode domain knowledge with the minimum possible effort from domain experts?

1.3 Thesis Contributions

The main contribution of this thesis is the conclusion that ensemble learning based on text-based CBR and Deep Learning can be used to process large amounts of complex textual data in real-time and provide accurate solutions. This work proves that deep learning models can decode textual domain knowledge automatically and build effective similarity measures. It is shown that similarity can be measured across different text cases with minimum involvement required from the domain experts.

A further contribution is the formulation, design, and implementation of a novel framework which is used in explainability of heterogeneous domains based on extending existing big data frameworks with **ElasticSearch** and **Keras** to build deep learning models. This framework provides a generic environment to facilitate building an integrated **CBR-BigData-DeepLearning** system in addition to its Knowledge Repository Layer (KRL). The KRL can contain limitless experience regarding the inspected cases and serves as a reference case base that can be used for reasoning and explanation.

A final contribution of this thesis is the extensive and thorough literature research of current CBR challenges, explainable AI, deep learning, and big data architectures covering the amalgamated cross-discipline areas. This review has been exhaustive and detailed and has led to a lot of insights.

1.4 The Approach

This thesis presents Deep Knowledge Acquisition Framework (**DeepKAF**), which was designed for developing distributed large-scale CBR systems. **DeepKAF** employs several deep learning models to overcome the limitations of a textual-CBR system. The deep learning models have been used for two main purposes, namely to decode textual domain knowledge (mixed-language text), and semi-automate the building of similarity measures. The

knowledge acquisition layer is based on a word embeddings model that is an unsupervised learning approach to build relationships within text. Later on, the word embeddings model is later on used in the similarity measures layer where a Siamese network architecture is being applied. The pre-processing layer used a Long Short-Term Memory (LSTM) model to eliminate unnecessary parts from the text and made it ready to be fed to the Siamese network, which is responsible for the retrieval. The LSTM has been replaced with an autoencoder in a later stage of the implementation when we changed the use-case and faced a long text that needed to be summarized somehow. Autoencoders helped in finding a lower dimensional representation to text and thus improving the retrieval results with less effort. The Siamese model main function was to measure similarity between two distinctive objects. Siamese networks are the state of the art in the area of comparing two objects, and showed significant improvements concerning the retrieved results. The Siamese architecture was selected after many experiments using other architectures and a lot of user experience influencing the entire approach. The **DeepKAF** approach tries to minimize the required effort of domain experts to decode the domain knowledge.

The entire approach looks like a long deep learning pipeline where the output of one model is an input to the next one, and this pipeline is contained and controlled within a CBR system that rules how the models are being used together.

In the course of this thesis, state of the art (and beyond) technologies have been integrated in a novel architecture for creating similarity assessment methods (semi-)automatically, rather than consulting experts. Instead, I am using deep learning models to understand the domain knowledge and retrieve similar cases fast. Experts were involved to evaluate and validate the retrieved results and give recommendations on how to improve the retrieved results.

1.5 Required Knowledge

The reader of this thesis will be introduced to CBR challenges, deep learning models and the big data Hadoop eco-system. However, basic knowledge in the areas of software engineering and deep learning is assumed. In-depth knowledge of CBR systems and challenges would be of great benefit.

1.6 Thesis Structure

This thesis is divided into **eight chapters**. **Chapter 1** introduces the thesis topic with an abstract motivation and challenges that CBR currently is facing. Since this thesis is bridging gaps among CBR, deep learning and big data, **Chapters 2, 3 and 4** define the state of the art in CBR, deep learning and big data. They are showing the advantages of intensively using deep learning within the CBR paradigm to solve complex problems, which might be faced by any CBR system implementation in the industrial domain. **Chapter 2** answers the question of "Can CBR be effectively applied in industrial domains?"; to answer this question, some literature and traditional approaches have to be mentioned and introduced because they will be used later on in this thesis. **Chapter 2** also covers the literature of the traditional CBR cycle, along with the challenges that CBR faces when it to comes to industrial implementations. **Chapter 2** also presents three CBR approaches, namely distributed, large-scale and hybrid systems. These approaches are the main focus of this thesis, since **DeepKAF** is using "big data and deep learning as a hybrid approach" to build distributed and large-scale CBR systems.

Chapter 3 is an in-depth literature survey of CBR sub-processes, with the most popular frameworks and architectures tackling the same problem **DeepKAF** is covering. The state of the art in the CBR area is defined in **Chapter 3** by showing how recent work relates to **DeepKAF**, what the differences are between one another, and showing how **DeepKAF**

can overcome repetitive chronic challenges presented in this literature.

Starting from **Chapter 4**, the "bridge" between the three areas (CBR, big data and deep learning) is being built. **Chapter 4** focuses on showing why CBR can be a successful approach in industrial domains.**Chapter 4** also introduces the deep learning models that being used within **DeepKAF** implementation, including justifications why specific models are chosen.

Chapters 5 and 6 are the core chapters describing **DeepKAF** and presenting the findings in relation to the most relevant related work. **Chapter 5** looks at **DeepKAF** from an architectural perspective. The architecture describes where to use the deep learning models and for which purposes, along with their training processes. The state-of-the-art literature in **Chapter 5** is related to building similarity measures and hybrid approaches that used deep learning in CBR. **Chapter 5** also explains the differences between the former literature approaches and **DeepKAF**. In addition, **DeepKAF** explainability is going to be explained and compared with ML approaches and how **DeepKAF** is a generic methodology and can be applied in different use-cases with the same or different models depends on the type of the problem. , Lastly, **Chapter 5** explains the competitive advantage of **DeepKAF** over the other approaches and provide ideas for future uses of **DeepKAF** Based on the architecture described. **Chapter 6** continues with the experiments and validation of **DeepKAF**. **Chapter 6** also focuses on the history of the experiments that were carried out amid the implementation of **DeepKAF**. These experiments include the final version that gives the optimal results. Finally, the technical details, frameworks, and cloud provider that were used throughout the implementation. **Chapter 7** concludes the entire work that has been done in this thesis and presents ideas for the future work with **DeepKAF**.

Chapter Two

CBR Applications in the Industrial Domain

Decisions under pressure are a common characteristic of modern processes and real-time industrial applications. With a constant flow of valuable data points, processes are called to perform better and faster whilst the decision time remains the same if not shorter. Over the past years we have seen impressive work in the areas of neural networks and deep learning, enabling faster reasoning over constantly working data flows. However, there has been less work done in the areas of reasoning under fuzziness, incomplete information and uncertainty. For example, decision support systems dealing with customers based on text information have a high degree of fuzziness and uncertainty. This situation becomes more challenging when their available text contains abbreviations, missing words, or is multilingual. Motivated from the above challenges, this work presents the **Deep Knowledge Acquisition Framework (DeepKAF)**, a case-based reasoning framework for rapid application prototyping on natural language processing workflows. **DeepKAF** presents an end-to-end natural language framework, and it has remarkable applicability and acceptance in industrial applications that share multi-languages, mixed notations, and content fuzziness. The key idea behind its inception has been the observation that processes can be represented as workflows and workflow states can be inferred from text flows (signals) using historical knowledge (cases).

Workflow signals can be inferred to workflow processes (Kapetanakis, Miltos Petridis, et al., 2010), (Kapetanakis and Miltos Petridis, 2014) and processes to experiences leading to appropriate mitigation policies. This work has been evolved over a number of real applications (Amin et al., 2018c), (Amin et al., 2018a), (Amin et al., 2020) and this thesis presents its core concepts, architecture and implementation using deep learning models and similarity metrics.

2.1 About this Chapter

This chapter presents the motivation behind **DeepKAF** by describing the CBR potential in the industrial domain, as well as explaining the current challenges that a CBR system is facing. It is important to discuss the challenges that any CBR implementation may face in the industrial domain, since **DeepKAF's** work investigates how CBR can scale in an industrial domain and against traditional ML approaches. This chapter investigates three generic types of CBR systems, namely distributed CBR, large-scale CBR, and hybrid CBR approaches. These three types are the most relevant to what **DeepKAF** is doing and what would help a CBR system to "survive" in an industrial domain.

The following sections present the general CBR methodology, architecture as well as the challenges that face CBR shown within the CBR literature and challenges that the CBR community has faced over the years. Several approaches, tools and architectures were introduced to solve some common problems, but they are all very domain-specific. **DeepKAF** is an approach towards building a more generic CBR architecture that can handle the current requirements for any industrial implementation.

2.2 Case-Based Reasoning

When systems grow in size and become more complex in nature, Artificial Intelligence (AI) is seen as a mean of automation, efficient management and coordination. AI brings together the underlying infrastructure, the models built and the fundamental machine intelligence to handle resources and workload efficiently. AI techniques are being used widely in the fields of banking, finance, engineering, medical, military and education in order to solve complex problems. Among the AI techniques, the Rule-Based Systems (RBSs) can also be referred to as Rule-Based Expert Systems (RBESs) and the Artificial Neural Networks (ANNs) are the most prominent in the development of applications. These methods, while popular and well known, are difficult to avoid downsides.

Starting with the RBESs and briefly exploring their application model, it is found that, in order for the RBES to function effectively, a series of rules must be specified that depicts the overall domain As a consequence of the RBES method, the entire domain knowledge has to be compiled in advance in order to be able to function. This can be a daunting challenge for applications that have been in service for a number of years. Existing knowledge can be transferred to the RBES by very basic guidelines, something that cannot be readily achieved if the system spans a broad operational range. In those cases, the number of rules will be huge. Maintaining such a system could prove to be a difficult challenge, too, if the rules continually change or new rules appear, not to mention the rules that include overlaps between them. Over all, if there is no fixed rule for a given problem, there is no chance of finding a solution to that particular problem (Kersten and Meister, 1996). Artificial neural networks do not require the entire domain knowledge acquisition from the domain model in order to present a solution to the addressed problem, an advantage over the RBSs. However, they are faced with substantial limitations since they are numeric restricted (Arditi and Tokdemir, 1999) and their way of operating remains hidden while producing the solution. As a consequence, the solution given cannot be easily verified as crucial information is lacking

such as: the technical information, the parameter weights, the learning rules and the internal functionality of the ANN. Therefore, it is not straightforward to provide explanations, making ANNs general applicability challenging in many domains.

Case- based Reasoning (CBR), a modern computational model, could be proposed towards the elimination of the above-mentioned drawbacks. CBR is fast in construction compared to typical RBESs, easier to maintain and can cope with complex structures. This is an advantage in comparison with ANNs that use numeric input or symbolic patterns to deal with the complexity of structures. CBR can also work with some cases, an advantage compared to the domain-knowledge acquisition requirements of the RBESs (Prentzasm and Hatzilygeroudis, 2007) and the exhaustive domain requirements of the ANN learning phase.

2.2.1 CBR Methodology – Architecture

AI systems are proven to be an efficient approach to handling, evaluating and supporting large-scale enterprise systems. CBR is an AI set of techniques that work together to solve problems similarly to the human brain, based on previous experience. Ian Watson (1997) in his book Applying Case-Based Reasoning: Techniques for Enterprise Systems states: "Case-based reasoning (CBR) is an intelligent-systems method that enables information managers to increase efficiency and reduce cost by substantially automating processes such as diagnosis, scheduling and design". As Watson indicates, CBR can be regarded as a computational methodology rather than a technique. CBR can use a variety of AI techniques towards the solution attainment during its Retrieval and Adaptation phases. There are innumerable examples in real life that show how experts in technology fields work: familiarize themselves with the domain, collect specimens, define a methodology, etc. These techniques could be generalized. A medical practitioner gathers the symptoms of a disease, and based on prior experiences, is likely to determine the potential cause of the disease. Depending on the experiences of the past, a medicine can be created, and if the condition appears to be

approached, the confirmation comes, and the rest of the medicine is developed with certainty. In the case of non-successful treatment, the drug switches to the next potential condition suggested by the clinical practice, often depending on the experienced symptoms.

Professionals in the fields of law, mechanical engineering, industrial environments and even ordinary people in everyday life find solutions to problems based on analogy making (Mitchell, 1993; Hofstadter, 1985). CBR is based on the experience obtained over the years, rather than the overall knowledge available in the investigated domain. As a consequence, the cases involve either problems with proven solutions or problems with unsuccessful attempts to fix them. In all cases, the user of the CBR system is of value as an explanation is always given. As a result, he/she knows what to do if the proposed solution was good or what to do in some other situation.

2.2.2 Case-based Reasoning Cycle

CBR is a series of artificial intelligence techniques that uses existing and past knowledge in order to find a possible solution to a given problem. This knowledge can be stored in repositories and formed in definite cases. In CBR, problems are characterized from regularity and recurrence (Aamodt and E. Plaza, 1994). Regularity means that, if a problem reappears, the knowledge learned from one previous problem will potentially be applied to the new problem. Re-occurrence refers to the fact that problems tend to have recurred over time, either continuously or occasionally. These problems can be classified by their type. The solution that works for one problem may, after applying the necessary changes, work for another problem of the same type.

CBR techniques can be generalized in a formal way into four-step processes (Aamodt and E. Plaza, 1994). This is generally referred to as the CBR cycle or the four R's processes where R's stand for Retrieve, Reuse, Revise and Retain (See Figure 2.1. The procedures carried out by each stage deals with knowledge memory represented in the form of cases. A

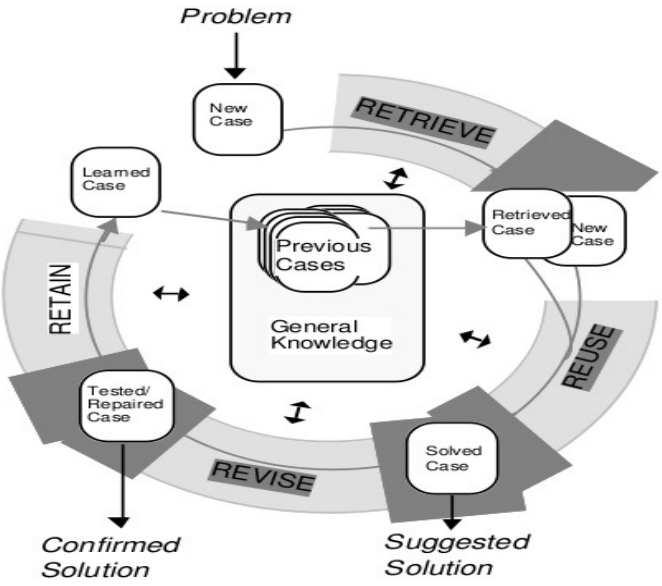

Figure 2.1 The CBR Cycle [Aamodt & Plaza, 1994]

case in CBR usually involves a problem and a known solution for that, as well as the related information that contributed to the addressed solution.

The stages of a CBR process are explained briefly below.

1. **Retrieve**: Whenever a problem is forwarded to the CBR system, the system treats it as a new case - problem with an undefined solution. The role of the CBR system, when investigating a new case, is to scan the knowledge repository for related cases to the one under review. The method then retrieves cases from the repository on the basis of the highest similarity to the case under review. The similarity measures applied will range from very basic ones (if they have to do with numbers, marks, and dates) to sophisticated ones (if the cases represent whole architectural structures or mechanical models).

2. **Reuse**: If a similar case has been identified from the repository, the solution suggested in that case would be followed as the one recommended for the case under review. There is a strong possibility that the desired solution would not be specifically applicable to the incoming case. In any case, the alternative solution must be tailored to satisfy the

particular criteria of the case under review.

3. **Revise**: If a solution has been provided for the case in query, this solution must be verified either in the context of a simulation or directly in a real-world test scenario. This solution frequently has to be revised in order to satisfy the needs of the case under review.

4. **Retain**: If the above processes have been successfully completed and a validated solution to the problem under review has been given, a new case can be formulated. This new case would be made up of the imported problem, its proven solution, plus any available knowledge on the environment. This latest, complete case can then be retained in the knowledge repository for potential use.

The CBR cycle (four R's) can be used across a wide variety of applications due to its generic features and the versatility of adaptation across systems. The next section refers to the relevant work as it is being illustrated from the literature.

2.3 CBR Challenges in the Industrial Domain

Each industrial domain has its unique challenges, however there are some common challenges that most of the applications will face regardless of what domain it is being implemented at. In this thesis, different industrial challenges will be mentioned. Those challenges drove the implementation of **DeepKAF**.

1. Heterogeneous Data types. In industrial applications, data that should be used to build an AI system can have different types. There could be a data attribute that is textual and another attribute that is an image or set of images. CBR systems need to be able to handle different data types within one case and still being able to retrieve most similar cases (Amin et al., 2019).

2. CBR systems require a huge effort to be exerted from Domain Experts to help in understanding domain knowledge, which costs organization a lot of money.

3. CBR systems are slow to respond to queries, comparing to other AI approaches. The querying process involves many underlying sub-processes in order to answer the query at the end.

4. CBR systems are computationally expensive because the algorithm has to search the entire case base to find accurate solutions, then adapt them to the query.

5. The case base needs to regularly maintained and cleansed from noisy data or unnecessary cases.

6. CBR systems consume a huge storage space to store the entire case base and the new cases that are coming.

7. CBR systems are not designed to cope with the data-in-move as most of the industrial applications need (Jalali and D. Leake, 2015)

8. CBR literature does not show a scalable CBR system implementation in the industrial domain that satisfies the current Big Data Era challenges.

9. Last but not least, CBR approaches are not popular because it does not have a big community as other approaches. The smaller community leads to less research and less implementations accordingly.

2.4 Distributed CBR

Distributed computation is a general topic in Computer Science. In CBR, there is a prominent direction towards building distributed CBR systems. In principle, the distribution of resources within any system can have a significant impact on the overall system performance.

CBR needs efficient techniques to manage its sub-tasks such as collecting and formatting data, case base maintenance, cases retrieval, cases adaptation and retaining new cases. From this point of view, the need to build distributed CBR systems for maximum efficiency increased. Multiple Agent CBR systems are widely used and very well known in Distributed CBR systems area, a lot of frameworks and related work have been carried out elaborating different architectures and techniques to manage the CBR sub-tasks.

Most researches in Distributed CBR concentrated more on distributing resources within the CBR architecture, but not on distributing the case base itself. One of the successful distributed CBR platforms is jCOLIBRI (Bello-Tomás, González-Calero, and Díaz-Agudo, 2004). It supports the development of wide range of CBR software, it provides the required infrastructure to implement CBR systems (Belén Diaz-Agudo et al., 2007). jCOLIRBI is depending on multiple agents to perform the subtasks associated with CBR. Multi-Agent Systems (MAS) distribute the case base itself and/or some aspects of the reasoning among several agents (E. Plaza and Mcginty, 2005). I can categorize the research efforts in the area of distributed CBR using two criteria:

1. How knowledge is organized/managed within the system (i.e. single vs. multiple case bases).

2. How knowledge is processed by the system (i.e. single vs. multiple processing agents).

There are many architectures and frameworks that built their strategy based on a distributed approach like **SEASALT** architecture Reichle, Bach, and K.-D. Althoff, 2009 and **jCOLIBRI** (Bello-Tomás, González-Calero, and Díaz-Agudo, 2004) which are going to be discussed in more details in **Chapter 3**.

DeepKAF is not mainly designed as a distributed framework, however, we have done some tests to build small agents that are responsible for specific tasks. For example, having an agent who is the coordinator that will get the query and decide to which agent this query is going to be passed on some parameters. we have built an agent that is responsible for the

text pre-processing before communicating the results to other agents). It was a successful small experiment without moving forward and implement a wider experiment across a set of agents. Conceptually, **DeepKAF** can be applied in a distributed approach with slight tweaks.

In the literature, there are many successful distributed CBR implementations (E. Plaza and Mcginty, 2005). Interesting research has explored distributed, multi-agent system frameworks whereby individual agents are willing, but are expected, to draw on the experiences of other agents for problems which do not lie within their own field of expertise. Plaza et al. (E. Plaza and Ontanon, 2001) has suggested many forms of coordination between CBR agents. In CBR, Plaza et al. (E. Plaza and Ontanon, 2001) ensemble, they research the "ensemble effect" which improves the accuracy of any person by the selection of agents with unrelated cases. Ensemble effect decreases when individual agents have a case base with a partial sample of the case region, but it shows that agents may barter cases using different policies to enhance output individually and collectively.

Generally, CBR and MAS have proved efficiency with different successful distributed CBR systems. Currently, systems are getting more complex and agents need to be smarter to be able to deal with its environments. MAS bring a lot of advantages and benefits to CBR, but also have a lot of challenges and issues that should be taken into consideration while building systems:

1. How do we design our algorithms to decompose tasks to agents and allocate problems to them?

2. If systems are widely distributed, how are agents going to communicate and what communication protocols will they use?

3. What if we lost the communication between agents?

4. How do we ensure that agents are working properly, and every single agent is doing its

task in the perfect manner?

5. How do we troubleshoot issues across all agents?

More literature in the area of Distributed CBR is being discussed in (E. Plaza and Mcginty, 2005)

2.5 Large-Scale CBR

Data is being generated extremely fast — a process that never stops, like the data generated from social media networks. Facebook, the most active of social network, with over 1.4 billion active monthly users, generates the most amount of social data – users like over 4 million posts every minute – 4,166,667 to be exact, which adds up to 250 million posts per hour (Carey-Simos, 2019). With the volume of data growing at unprecedented rate; CBR has a new challenge to deal with this large amount of data. In the industrial domain, all companies now have a lot and different data sources, over the days the size of data is getting bigger and the need to deal with this amount of data seamlessly is also increasing (e.g., electronic manuals, history of failure, electronic medical records) (G.-H. Kim, Trimi, and Chung, 2014) (Jalali and D. Leake, 2015). Therefore, the key factor for the next generation of CBR applications is the ability to deal with the complex and large amount of data that is generated every day and every moment.

Vahid Jalali and David Leake have explained in details the idea behind Large-Scale CBR (Jalali and D. Leake, 2015). In the CBR literature, most of the work focused on compressing the Case Base and other Case Base maintenance activities which are essential activities, however, by compressing or deleting some cases, we might lose an opportunity to retrieve a case that can help in giving a potential solution for adaptation. Therefore, from this thesis point of view, we look at large-scale CBR systems as CBR systems that can work efficiently with the Big Data Challenges which are going to be discussed in **Chapter 4**. Instead of

trying to compress cases only, with the new emerging big data technologies, we are able to retain more cases without affecting the retrieval time (Amin et al., 2018a).

2.6 Hybrid CBR Approaches

Conceptually, hybrid approaches are approaches where we bring the best from two or more different worlds and put them together to solve a specific problem. There are so many examples of Hybrid CBR approaches that have been used to build different applications, especially Recommender Systems (Alshammari et al., 2017) and (Gedikli, Jannach, and Ge, 2014) (Im and Park, 2007) (Simić et al., 2018).

DeepKAF as a hybrid CBR approach is bringing the best from ANN world to the CBR world.

2.6.1 Combined CBR and ANN Approaches

Combination between CBR and ANN has been explored over the last 30 years. (Lees and Corchado, 1999) explained in **1999** a hybrid-CBR approach where ANN was employed to do specific tasks. By that time in 1999 the amount of available data and knowledge along with the processing power were not enough to give good results. Nevertheless, the available systems at that time that were presented in (Lees and Corchado, 1999) gave very promising results and revealed the power of using CBR-ANN approaches, which become very popular since the revolution of Deep Learning and NVIDIA GPUs started.

ANNs are commonly used for learning and the generalization of knowledge and patterns. They are not appropriate for expert reasoning, and their abilities for explanation are extremely weak. Therefore, many applications of ANNs in CBR systems tend to employ a loosely integrated approach where the separate ANN components have specific objectives such as classification and pattern matching. Neural networks offer benefits when used for retrieving cases because case retrieval is essentially the matching of patterns.

During the implementation of **DeepKAF**, the main focus was on integrating ANN and use the advantages that this combination brings to CBR retrieval and similarity measure processes. The use of ANN within the CBR is not new and have been used for certain purposes.In (Dieterle and Bergmann, 2014), authors used CBR with ANN to predict internet domain name price which applied an innovative hybrid-CBR approach and used ANN for tasks to decode domain knowledge, which in this case is the domain names. Authors used ANN to perform the core text pre-processing tasks like stemming, eliminating stopping words and then transform the case base into to a numeric vector that have improved the retrieval process and helped in measuring similarities.

The problem about ANNs that they are very application-dependent. There is no one architecture that is going to be always applied. However, there are some recommendations for architectures that can help in fixing specific problems. **DeepKAF** is providing an entire integrated CBR architecture that shows where ANN can be used efficiently. Additionally, **DeepKAF** (Amin et al., 2019) is also introducing a combination of ANN that can help in decoding a textual domain knowledge. However, ANN domain is moving fast, and new innovations are coming every day. The idea is to bring ANN within the CBR paradigm without hiding the wining explainability feature of CBR.

2.7 Summary

This chapter elaborated the idea behind the CBR paradigm with focus on challenges that any CBR system will face if it is being implemented in an industrial domain. We have discussed distributed CBR approaches with MAS and large-scale CBR as an introduction to the new approaches in the CBR world that can handle big data, which are going to be discussed in details in **Chapter 4**. The approaches being mentioned last are the hybrid CBR approaches that are currently sky-rocketing according to CBR literature. Although CBR includes many other approaches, I want to focus here on the approaches that **DeepKAF** benefits from

mostly.

In the remainder of this thesis, the chapters will focus on what could stop CBR from being a prominent AI approach in the industrial domain, along with defining how to overcome some of these issues by merging other AI approaches within the CBR paradigm.

Chapter Three

CBR Foundations and State of the Art

3.1 About this Chapter

This chapter discusses the CBR state of the art and foundational concepts. **DeepKAF** experiments are focused on Textual CBR, however, I believe that it could be applied to other CBR types as well. Therefore, the common CBR types are described and explained in this chapter, along with the state of the art work related to the implementations that approached those specific types. At the end of each section, it is explained how the presented work (theoretically) relates to **DeepKAF** and what are the advantages/disadvantages of using **DeepKAF**. It also explains in detail the popular CBR architectures, frameworks, and tools being on the top of the state of the art CBR approaches and can be directly compared to **DeepKAF**.

3.2 CBR Types

CBR can be categorized in a number of types depending on their operational domain, context, and data. These CBR types could be generalized in three categories. These are: Structural, Textual and Conversational CBR. These categories are drawn from the essence of the CBR application domain and briefly illustrated below.

3.2.1 Structural CBR

Structural CBR approaches are based on cases that represented using attributes and their corresponding values. Structural CBR systems are responsible for organizing attributes in different ways, e.g., as flat attribute lists, as relational tables, or in an object-oriented way T. R. Roth-Berghofer, 2004a. For cases where additional information, such as complex similarity measures, is needed, the structural CBR approach is helpful in order to produce good outcomes and where the domain model is easily obtainable.

3.2.2 Textual CBR

Text is used to express knowledge. Text is a collection of words in any well-known language that can convey a meaning (i.e., ideas) when interpreted in aggregation(Richter and Weber, 2013). Text processing and analysis is considered a "must-have" capability due to the immense amount of text data available on the internet. Textual CBR is Structural CBR at its core, where semi-structured text are transformed into a structured representation. The text representation brings several challenges when the text is unstructured or has grammatically incorrect sentences. The task of the approach described in our research can be compared to the work presented in (Stram, Reuss, and K. Althoff, 2017; Reuss, Witzke, and K. Althoff, 2017; Reuss, Stram, et al., 2016), the authors used a hybrid CBR approach where they combined CBR with NLP frameworks to be able to process the knowledge written in free text. They mentioned to the issues they faced with the free text or to extract features and build accurate similarity measures. In our work, NLP frameworks were not able to process text spanned across different languages and there were several issues related to accurate sentence parsing. Therefore, we applied a different approach using Deep Neural Networks (DNN) to ease the task of finding similarities between cases and automate the knowledge from textual cases.

Finding the relation between text and extract features are key criteria in the success

of any textual CBR system. These tasks require a lot of effort and normally can take a long time to be done accurately. Different approaches have been presented to build text similarities and find higher order relationships (Öztürk, Prasath, and Moen, 2010). The work of automating knowledge extraction using Neural Networks can be compared to the work presented in (Sizov, Öztürk, and Štyrák, 2014) where authors represented the text using dubbed Text Reasoning Relevant work has been seen in Graph (TRG), a graph-based representation with expressive power to represent the chain of reasoning underlying the analysis as well as facilitate the adaptation of a past analysis to a new problem. The atuhors have used manually constructed lexico-syntactic patterns developed by Khoo (S. G. Khoo, 1996) to extract the relations between texts.

Nowadays, most real-world applications are more complex than the examples given above. They are typically composed of several attributes in nature, which are not easy to interpret.

3.2.3 Conversational CBR

A considerable amount of research in CBR has recently focused on conversational CBR as a means of providing more effective support for interactive problem-solving. Conversational CBR is a type of CBR systems where users can have a kind of "conversation" with the system before submitting the query. In typical CBR systems, users are given a form where they fill some details and description, then submit it to get the answer. If the user is not able to describe the problem in a way that the system can understand, the system will not answer the query correctly. Conversational CBR systems are more flexible and dynamic in building the query process. The user is given an opportunity to get some questions and answers to formulate the query before submitting it to the CBR system (Richter and Weber, 2013). In other words, as Aha et al (Aha, Breslow, and Muñoz-Avila, 2001). defined Conversational CBR, "in conversational CBR (CCBR), a query describing a target problem is incrementally elicited in an interactive dialogue with the user, often with the aim of minimizing the number

System/Tool	Description	Features
Adaptive Place Advisor (Thompson *et al.*, 2004)	CCBR Recommender System	• Spoken dialogue interface • Question selection based on information gain • Dialogue operators • Query relaxation
CaseAdvisor (Carrick *et al.*, 1999)	CCBR Tool for Fault Diagnosis	• Free text entry of problem descriptions • Cost sensitive question selection • Automated information gathering
CBR Strategist (McSherry, 2001a)	CCBR Tool for Fault Diagnosis	• Goal driven dialogue • Explanation of question relevance
ExpertClerk (Shimazu *et al.*, 2001)	CCBR Tool for Knowledge Space Mentoring	• Question selection based on information gain • Script-based indexing of cases
NaCoDAE (Aha *et al.*, 2001)	Generic CCBR Shell	• Free text entry of problem descriptions • Ranking of questions based on frequency of occurrence in competing cases • Dialogue inferencing
Sermo (Bridge, 2002)	Conversational Recommender System	• Dialogue grammar • Query relaxation
ShowMe (McSherry, 2004)	Conversational Recommender System	• Explanation of query failure • Incremental query relaxation
TCRS (Gupta *et al.*, 2002)	Taxonomic CCBR Tool	• Taxonomic representation of cases and focusing of conversations • Dialogue inferencing
Top Case (McSherry, 2005b)	CCBR Recommender System	• Goal driven dialogue • Explanation of question relevance • Solution justification • Dominance criteria for dialogue termination

Figure 3.1 Example CCBR (and related) systems and tools, Reference: (Aha, McSherry, and Yang, 2005)

of questions the user is asked before a solution is reached (McSherry, 2002).

Conversational CBR approaches have been used intensively in the industrial CBR applications due to their ability to formulate the problem correctly based on a dialogue with the user, as described earlier. Figure 3.1 shows different successful CCBR systems.

3.3 Temporal concept in CBR systems

In a certain number of application domains, the concept of temporal reasoning is important, since actions that take place within certain time frames may have a totally different meaning. Time can also define events in terms of importance. An example could be how a monitoring system conceptualizes a 5-minute time span in a police emergency system. If this time span is the response time from a crew conducting a routine patrol, it can be regarded as acceptable. However, if this time span is the response time in the middle of a critical rescue operation , it definitely means that something does not go as well as it was expected. During the latest years, there is an emerging need for temporal reasoning in CBR systems. There are mainly two trends for this (Jære, Aamodt, and Skalle, 2002): The first is that CBR systems are called to solve real, challenging problems where there is need for extensive temporal reasoning (such as medical diagnosis). The second is that CBR systems have become more interactive and transparent to users. Consequently, the problems are moving away from their container, towards user-interactive assistants. In this case, the sequence of actions is of relative importance. The literature can show several examples in the area of CBR systems where the temporal concepts are being taken into consideration. In the area of diagnosis and possible prediction of adverse events before they occur, work has been conducted by (Netten, 1998). The focus of that work was on the incorporation of time-dependent cases and temporal reasoning while attempting to configure and establish the operational conditions for technical applications. The outcome of that was the BRIDGE project whose objective was to improve performance of operational diagnosis systems as a means to improve safety, availability, reliability, maintainability and life cycle costs of large technical applications" (Netten, 1998). (Likhachev, Kaess, and Arkin, 2002) have conducted research work on the behavioral parameterization of robots using both spatial and temporal CBR. The aim of their research was to help robots learn the optimum behavior for autonomous navigation tasks, either by having the robots under (ordinary) training or by

following a mission-based training. During the latter, the robots could learn while trying and making mistakes during their assigned missions. Both temporal and spatial characteristics were taken into account in this research in order to be able to match the best case for a robot within its operating environment. In the area of oil drilling, temporal sequences can indicate a number of situations where call to action may be necessary. An example could be a situation where a drill string gets stuck in the borehole, and stops the drilling process. This can be an exceedingly costly problem since drilling operation hours are of high cost and the freeing of a stuck pipe is a lengthy process. As a result, a stuck pipe is one of the costliest drilling problems. (Jære, Aamodt, and Skalle, 2002) have used a CBR system which based on temporal domain representation could advise the users where and when to stop a drilling operation. Experience could show similar cases that led to stuck pipes. Possible explanation for the cause of that case was also given based on the general domain knowledge. The implemented system has used Allen's temporal theory which is based on intervals in order to have representation close to the way the human expert "reasons in domains where qualitative changes of parameters over time are important" (Jære, Aamodt, and Skalle, 2002). A more advanced approach to the problems faced in drilling industry has been given by (Skalle and Aamodt, 2005). This approach has combined both CBR components and model based components within the Troll Creek architecture. The overall objective of the work was to increase the efficiency and safety of the drilling process. In order to do that, past temporal experience was used to support fault diagnosis and prediction of possible undesirable events in a domain that is characterized from "uncertainty, incompleteness, and change" (Skalle and Aamodt, 2005). Additional research on Temporal CBR (TCBR) was conducted to support dam technicians in decision-making (Mohd-Hassin, Md Norwawi, and Ab Aziz, 2006). The decision to open or shut one or more spillway gates for water reservoirs is vital to safety. Dam -Specialized technicians also have to determine which reservoir spillway gate should be opened or closed to release excess water in order to ensure a safe water level. (Mohd-Hassin, Md Norwawi, and Ab Aziz, 2006) have designed a CBR engine that could support

temporal data mining. The implemented prototype has integrated CBR techniques used in a temporal data application for decision recommendation, based on historical data. Temporal hydrology data was used to evaluate the decision recommended by the prototype versus the dam's expert. For this research, in order to capture the time pattern plus delays, sliding window was used as a segmentation technique for this analysis.

In the area of CBR, expert systems (Floyd and Esfandiari, 2011) have worked on extracting the temporal relationships between sensory stimuli and expert actions. Their investigation was focused on how a CBR agent system (Wooldridge, 2009) could learn from the reactions of experts to specific events by observing them over a period of time. Behavior learning would be the optimum when the agent was able to perform like an expert on a specific input. (Julián and Corchado Rodríguez, 2012)have investigated the provision of temporal bounded reasoning in multi-agent systems that manage security in industrial environments. Their proposed approach facilitates the automatic reorganization of tasks in combination with possible faced environmental changes. Key insight of the research is the optimization of security tasks performed and the solution of problems with temporal constraints.

3.4 Uncertainty in CBR

The strength of CBR is utilized for building a situation dependent decision model without complete domain knowledge (Xiong and Funk, 2009). This strength comes from the ability of deriving decisions based on similarity measures and general utility function from the case library. In opposite to other machine learning techniques, which always come with the question "how much should we trust this decision?" (Hammer and Villmann, 2007). Artificial software for enterprises faces the problem of dealing with partial and often uncertain information. The information available in a system may be unreliable, e.g. "a patient may mis-remember when a disease started, or may not have noticed a symptom that is important

to a diagnosis" (D. B. Leake, 2002). Additionally, the rules governing a system cannot take into account all the possible parameters that may make their conclusions inapplicable, e.g. "the correctness of basing a diagnosis on a lab test depends on whether there were conditions that might have caused a false positive, on the test being done correctly, on the results being associated with the right patient, etc." (D. B. Leake, 2002). As a consequence, an artificial monitoring system should be able to have adequate reasoning for the probability of events based on their available knowledge.

The effective artificial intelligence application faces challenges in the case of flexible and adaptable work environment. The management team of a can find any information available in the form of timed event logs of actions that have been generated during its execution. Any workflow actions, communication messages as well as system generated messages can be found there. A challenge in the monitoring of workflows is that even with well-designed, well-defined workflows, the actual contextual information that affected the decisions taken can be missing. This can happen in many cases, although the system has managed to capture any provided information or even the actual event path that the system has followed. Another factor that makes more difficult the monitoring of workflows, with interfered human roles, is the inability to capture the overall contextual information and communications behind any posterior actions. Some system actions may take place in an unofficial way (e.g., manual interventions) based on informal verbal communications or meetings among actors. These actions may not be captured from the system in the form of a logged event. The structure and orchestration of a workflow does not necessarily define exclusively the final choreography and operation of the monitored workflow. As a result, in order to achieve effective monitoring of workflows, the factor of uncertainty has to be taken into consideration and effectively be dealt. The case of uncertainty will be extensively discussed later on in **Section 3.5**.

3.5 CBR Explainability

The explanation topic is an essential part in this thesis. Since, **DeepKAF** is combining an "explainable approach" (CBR) (M. T. Keane and Kenny, 2019) with an "unexplainable approach" (Deep Learning models) (Weerts, Ipenburg, and Pechenizkiy, 2019). Explanation is an important field in the area of intelligent systems. An approximate definition of system explanation is a set of logical arguments that could persuade the end-user to follow proposed conclusions, suggestions and even decisions to a certain degree.

In the field of artificial intelligence, and in particular in the field of case-based reasoning, systems are capable of extracting the knowledge of explanation from the available past knowledge. This can be presented afterwards to their human stakeholders in order to provide reasoning and justification for any system recommendations and/or decisions. The availability of explanations improves confidence in any artificial intelligence system and helps to build trust between AI and its users. Everyone's personal experience shows that building trust between people is a challenging job.

Explainability is one of the main advantages of CBR (Craw, 2010). **DeepKAF** investigates the explainability part of CBR because by using ANN, there will some sub-process within the CBR paradigm that are not explainable. However, **DeepKAF** and by applying the CBR paradigm will be able to provide adequate level of explainability that is enough to validate the retrieved solution (see Chapter 5).

3.5.1 The Concept of Explanation

Explanation presents a prominent topic in science (Schurz, 2000), often referred to as scientific explanations. Scientific explanations attempt to give answers to why questions by using existing facts and applying general rules. Nonetheless, these answers can be different regarding the application domain and as a result, explanations can differ. Also, there is a distinction between cause giving explanations and reason giving justifications (Schurz, 2000),

referring to the equivalent 'why' questions respectively. The first category of the above, questions why something has taken place like that, what was its cause or reason for being. An example of that could be:

Child's Question: Why is the water in the lake frozen?

Father's Answer: Because the water temperature is below 0º Celsius.

The second category explores the explanation for the cause of an incident. Explanations in this category may be applied as the explanation of the cause for why an incident occurred in a particular manner. An example follows:

Wife's Question: Why did not you call me the whole day?

Husband's Answer: Because I had too much work in the office, darling. I did not have the time even to get some lunch...

Although explanation seems a self-descriptive concept which reveals the answer to intended knowledge questions, in reality explanation can be literally false (T. Roth-Berghofer, 2004). This is being done deliberately due to moral, pedagogical, social and other context-dependent reasons (Cohnitz, 2002). These explanations are being provided in the context of satisfying the questioner, and not necessarily fulfil the purpose the questioner expects them to. An example can be a question from a seven-year old girl to her mother on how her little brother came to life. The most prominent answer that she could get is that the stork brought him home, when everybody was asleep, or some variants of the above. Explanations are heavily dependent on the background context of a given environment. Schank has characterized explanation as the most common method used by humans to support understanding and decision-making. An explanation should describe a solution to a problem, as well as which is the path that has to be followed in order to reach the solution. Therefore, explanations are characterized as both inclusive and instructive (T. Roth-Berghofer, 2004). A system can explain its actions both to humans and/or services that inquire how it works, as well as to itself. In such a way, according to Schank, it becomes an understanding system. The range of cognitive understanding of such a system can vary from making sense to com-

plete empathy (C. Schank, 1986). AI reasoning stands in the direction of the logical edge of the above spectrum, seeking to persuade its users rather than blindly leading them.

Explanation, according to **Schank**, can be divided into three main classes. These are the physical world, the social world and the individual behavior patterns. An artificial agent can give sufficient reasoning to a case that fits in to one of the above classes. Especially for the first class of explanation (physical world) reasoning can be derived from the laws of physics. Having those as a basis, more complicated explanations can derive for other (presumably) more complicated fields. Explanations for the social world and behavior patterns seem the most complicated ones being of high importance in the modern world. For the needs of the explanation to those areas, advanced techniques like data mining, user profiling, and machine learning could be used. These techniques attempt to extract the followed behavior patterns in an investigated discipline. The reasoning given by itself is not adequate to persuade a user of the clarity and accuracy of the decision made by an AI system. Therefore, trust in any AI system should be established in order to be effective for its individual users. Intelligent systems should include meaningful explanations before providing findings or feedback in order to improve the user trust in them. In order to do this with any particular device, output should be sent to its consumers in order to provide sufficient rationale for output. Users are often more persuaded of the accuracy of the approach because, next to the output (whether it is positive or bad), there is proof of how this output was derived (R. Swartout, 1983). While attempting to formulate explanation, certain goals should be fulfilled. To achieve that, (Sørmo, Cassens, and Aamodt, 2005) have identified five distinct explanation goals. These goals either isolated or in combination could be used to construct explanations. These are:

1. How did the System reach the answer? (Transparency)

2. Explain why the answer is a good answer. (Justification)

3. Explain why a question asked is relevant. (Relevance)

4. Clarify the meaning of concepts. (Conceptualization)

5. Teach the user about the domain. (Learning)

Usually, explainable systems are satisfying one or more of the above goals while attempting to provide sufficient explanation.

3.5.2 Explanation in Systems

As seen in the previous section, an explanation is required to address questions about the cause, the reason for existence or the outcome of an event. Artificial intelligence technologies are now being used in the field of problem-solving, in the same manner as human experts. Trust must be established for individual users in order to be able to determine whether to approve or deny an automated AI-based recommendation. The user must be able to dig down to a particular system-output and be able to verify, if necessary, what led the system to that direction. Due mainly to the requirement for adequate reasoning, the need for an effective explanation has been intensified.

The explanation that can be extracted from a system is based on the nature and the contextual background of the system itself. An example can be an artificial neural network, which by its nature contains the knowledge inside its internal structure. As a result, it works as a black box and cannot provide sufficient explanation to support its outcomes, since its knowledge cannot be extracted. A system that uses genetic algorithms to calculate its output and come up with recommendations faces similar explanation limitations. Rule-based systems can perform better since they can resort to the reasoning provided by their rules. However, several restrictions apply: In order for this to be efficient, the domain should be limited for the user to be able to follow and evaluate the explanation. In any different case, the complexity of the correlated rules can be unmanageable, since even experienced users are not usually able to follow such stated explanations (T. Roth-Berghofer, 2004). Another constraint rule-based systems face is a substantial growth in the number of rules

they have to contain in order to operate in a dynamic domain. This number, as well as the effort to collect them, makes them a relatively complicated approach to offer adequate explanations. Case-based reasoning systems can offer a solution to the faced limitations of ANNs and rule-based systems. This is due to their ability to present the information of their cases in order to provide support to a certain system output. Therefore, cases can be descriptive in terms of a problem and its surrounding information. However, cases do not provide information in terms of their selection criteria and the rationale behind that. As a result, a different entity is required that connects the related knowledge to a case and its available structure. This can be related to the concept of knowledge containers. Knowledge containers, introduced by Richter (1995), refer to both the knowledge in cases as well as the structure of that knowledge. Due to its structural generalization, a knowledge container can refer to a number of tasks combining their related characteristics to possibly one schema. Richter has introduced four knowledge categories for CBR systems. These are presented in **Section 3.9.1** and are: the vocabulary, the similarity measures, the solution transformation (or adaptation) containers, and the case base.

As an example for an explainable application that can be related to CBR in general and similarity measures in specific, Björn Forcher et al. (Forcher et al., 2014), presented a generic explanation facility ***Kalliope***, which is integrated in a search engine ***KOIOS++*** to enable semantic search on medical Wikipedia articles. **Kalliope** uses reconstructive explanations in order to provide justifications for medical semantic search results. This work is deeply connected with CBR from the similarity measures side and how to build similarities from domain-specific knowledge. **Kalliope** focuses on the selection of the most understandable rationale for medical lay-people. Björn Forcher also demonstrated that there are many possibilities for explaining one and the same search outcomes due to several marking alternatives and the inference of more relations between terms (transitive closure of is-a and part-of). The authors designed an interpretation of the weighting mechanism based on two basic principles, namely the semantic frequency class and semantic co-occurrence class.

The first can be used to predict if users are aware of the preceding concept, and the second can be used to predict if users are aware of relations between concepts. Both hypotheses have been verified by means of various user experiments, which have also been mentioned in this work.

3.6 Case Representation

In a representative paper on case representation (Bergmann, Kolodner, and E. Plaza, 2005), "Improving human decision-making through case-based decision aiding," Janet Kolodner focuses on the role cases can play in helping people make decisions and on the content cases need to have to play that role. Kolodner does not try to comment on the form that a case should take, but rather focuses on the kinds of things that should be represented in a case so that it can be productively used for reasoning. Kolodner recommends that cases include a problem situation description, the solution that was proposed (often including how that solution was derived), and the outcome, including the state of the world after the solution was carried out, how close that was to what was expected, and explanations, if necessary and available, of why it might not have worked as well as expected.

Case representation is the first step in any CBR system. The case-based reasoner has to decide how domain knowledge is going to be represented for reasoning purposes. The case representation stage is critical because it has direct effect on the retrieval process. The representation technique should be able to accommodate all pieces of information and store this information in a way that makes it easy to build relationships with each other (Similarity Measures).

3.6.1 Basic Representation Methodologies

Feature Vector Representation is used for domains with weak or intractable theories. A category is extensionally represented as a collection of cases called exemplars. A new case

is classified into a category if a match can be found between an exemplar and the new case. This matching process is knowledge intensive and tries to build an explanation that connects the features of the new case with an exemplar. Since each explanation is a path constructed inside a semantic net, retrieval is the process of explaining the (similarity) relation between a new case and an exemplar (Bergmann, Kolodner, and E. Plaza, 2005).

Object-oriented representation make use of the data modeling approach of the object-oriented paradigm, including is-a and part-of relations as well as the inheritance principle. Cases are represented as collections of objects, each of which is described by a set of attribute-value pairs. The structure of an object is described by an object class. Several object-oriented case representation languages have been developed. Object-oriented representations are particularly suitable for complex domains in which cases with different structures occur (Bergmann, Kolodner, and E. Plaza, 2005).

Textual Representation decomposes the text that constitutes a case into information entities (IEs). An IE is a word or a phrase contained in the text that is relevant to determine the reusability of the episode captured in the case. Given a vocabulary of the relevant words or phrases, text cases can be mined for IEs, allowing case acquisition to be automated. The set of cases that form the case base is organized in the form of a case retrieval net (CRN), which is a directed graph with nodes representing cases and their IEs. These nodes are linked according to their similarity. Hence, knowledge about similarity is encoded into the strength of the links between the nodes in the CRN (Bergmann, Kolodner, and E. Plaza, 2005).

Frame-based representation Frame based representations have been (partially) formalized by description logic. The notion of "cases as terms" argues that viewing structured cases as terms in feature logic (a particular brand of description logic) helps to better understand several aspects of case-based reasoning. Domain knowledge can be implemented by means of a sort hierarchy, and the problem of composite cases (cases that tie together certain ob-

jects or sub-cases) is clarified by the assumption that a sub-term is often a term. Lastly, the notion of similarity between two cases is linked to the concepts of subsumption and anti-unification of terms. This notion also bridges relational cases in CBR with relational learning in inductive ML, where subsumption and anti-unification are basic building blocks.

3.7 Similarity Measures

Similarity measures are the heart of CBR because retrieval is dependent on it. Building similarity measures process is highly domain-dependent and used to describe how cases are related to each other. In CBR, comparison of cases can be performed along multiple important dimensions (Ashley, 1991; Brüninghaus and Ashley, 1998). Cases that only match partially, can be adapted to a problem situation, using domain knowledge contained in the system (Aleven, 1998). Thus, methods, like in particular Information Retrieval, which are based only on statistical inferences over word vectors, are not appropriate or sufficient. Instead, mechanisms for mapping textual cases onto a structured representation are required. A basic assumption for applying the principle for similarity measures is that both arguments of the measure follow the same construction process. This allows us comparing the corresponding sub-objects in a systematic way. For our system, we defined the two types of similarity measures: Local Similarity Measures and Global Similarity Measures. Local Similarity Measures describe the similarity between two attributes, and the Global Similarity Measures describe the similarity between two complete cases. In the next section, we elaborate how we applied the Local Similarity Measures followed by the Global Similarity Measures.

Local Similarity Measures: Based on the collected data and the discussions with experts, we defined the local similarity measures. We have mainly four attributes which are distinctive, except for the email subject and content. For the Priority (integer) and Sending

Groups (distinctive strings) we used distance functions. For the email subject and content, we counted upon the Word2Vec model to give us the similarity degrees between different texts, after applying all the aforementioned prepossessing tasks.

Global Similarity Measures: The Global Similarity Measure defines the relations between attributes and gives an overall weight to the retrieved case. The weight of each attribute demonstrates its importance within the case. The weight of each attribute has been defined in collaboration with the domain experts.

3.7.1 Taxonomic Similarities

Taxonomies are commonly used structures to approach a variety of domains like ontologies. Nevertheless, building similarities using taxonomic graphs have been widely used in CBR. Taxonomies are used to link objects to each other in a hierarchical representation. The intention is to do this in a hierarchical way, from general to more specific objects. This means that branching leads to objects that have more in common (Richter and Weber, 2013). Reuss et al. (Reuss, Stram, et al., 2016)have used taxonomies to build similarities in the aircraft domain. They used the keywords, phrases, and hypernyms to generate taxonomy similarity measures that are used in the retrieval process. The work presented in FEATURE-TAK framework is highly relevant to **DeepKAF**. Despite taxonomies similarity measures are efficient in the ambiguous domains, the effort needed to decode domain knowledge and build the taxonomies was a big obstacle in building the CBR system. **DeepKAF** intention as described in this thesis is mainly to minimize the effort required to build CBR system compared to other approaches and methodologies.

3.7.2 Graph Similarities

In certain applications, it is also required to compare objects represented as graphs and to decide how similar the objects are.

Similarities between graphs representing molecules have an important role to play in many areas of chemistry and, increasingly, biology. Examples include the database searching (Peter Willett, 1999), the analysis of biological activity (Carletti, Foggia, and Vento, 2013), the architecture of mixture synthesis (Lewis, Mason, and McLay, 1997) and the interpretation of molecular spectra(L. Chen and Robien, 1994). In (Raymond, Gardiner, and P. Willett, 2002), Raymond et al. proposed a brilliant inexact graph matching procedure, RASCAL (Rapid Similarity CALculation). The screening procedure is intended to determine rapidly whether the graphs being compared exceed some specified minimum similarity threshold (without resulting in any false dismissals) in order to avoid unnecessary calls to the more computationally demanding, graph matching procedure. The screening procedure is described in **Section 2** along with an example calculation. The latter graph matching process consists of an efficient determination of the MCS using the graph similarity concept. Graph similarities is extremely relevant to **DeepKAF** since the entire framework depends on representing text as graph and calculate distances. The main difference in **DeepKAF** is how **DeepKAF** is using ANN to find the best graph representation to the text before comparing them.

3.8 Retrieval

The main objective behind any problem-solving technique is to attain an accurate solution to the problem, ideally even the best. In CBR, retrieval is the process of attaining suitable solutions for the new case. That retrieved case should have similarities based on the similarities measures to the new case. Although retrieval process always implies the "finding" process of the cases, it differs from one CBR system to another based on the CBR system

type and problem domain. Therefore, several retrieval approaches have been introduced to cope with different problems from different domains.

3.8.1 Index-Based Retrieval

Case indexing refers to assigning indexes to cases for future retrieval and comparison. The choice of indexes is important to enable retrieval of the right case at the right time. This is because the indexes of a case will determine in which context it will be retrieved in the future, indexes must be predictive in a useful manner. This means that indexes should reflect the important features of a case and the attributes that influence the outcome of the case, and describe the circumstances in which a case is expected to be retrieved in the future (Shiu and Pal, 2004).

Indexes should be abstract enough to allow retrieval in all the circumstances in which a case will be useful, but not too abstract. When a case's indexes are too abstract, the case may be retrieved in too many situations or too much processing is required to match cases. Use inductive techniques for learning local weights of features by comparing similar cases in a case base. This method can determine the features that are more important in predicting outcomes and improving retrieval (Shiu and Pal, 2004).

Indexing cases is one of the most challenging tasks in CBR. The index structure is supposed to guide the search for the most similar case(s) (see Figure 3.2 (Richter and Weber, 2013). Therefore, two steps have to be performed: 1. Generating the index, 2. The retrieval itself. The index generation for the whole case base takes place offline while the retrieval is online. The performance of retrieving similar cases in large-scale CBR was seldom been discussed. When the number of cases in the case base becomes large, the processing time for retrieving similar cases rapidly increases. The process of retrieving similar cases thus becomes a critical task of CBR. That brings the index-based retrieval to the table.

Cases indexing can significantly improve the retrieval process in two dimensions: 1. The

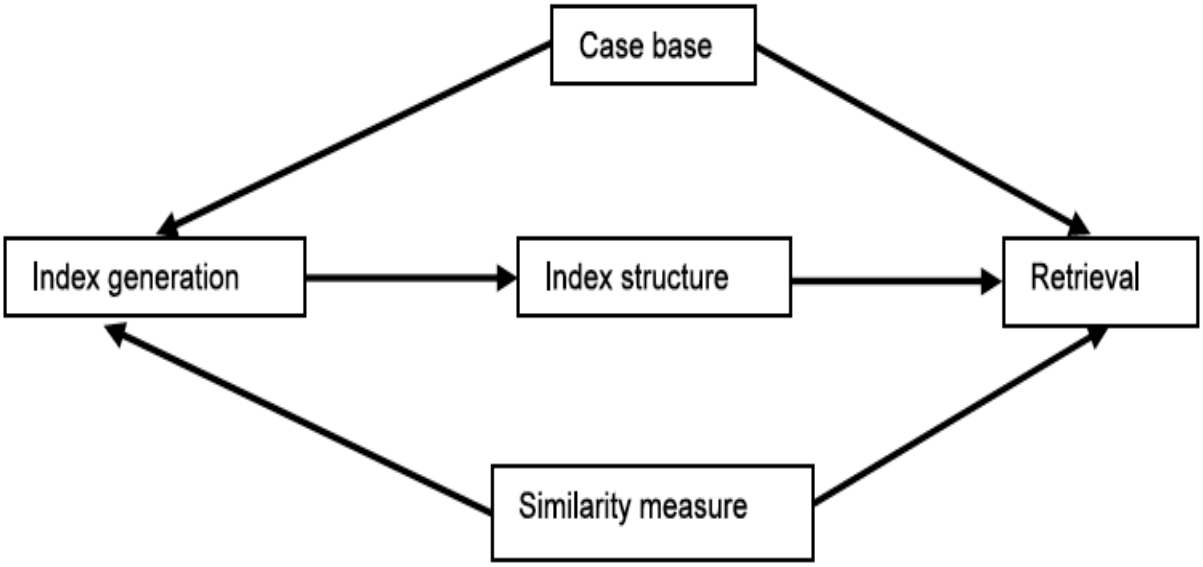

Figure 3.2 Cases Indexing Design Steps
Reference: (Richter and Weber, 2013)

speed of retrieval 2. If it is a conversational CBR, a good indexing approach can minimize the number of questions required to be able to retrieve the most similar cases

In (El-Bahnasy, Amin, and Aref, 2014), the authors introduced a novel CBR indexing approach based on Power Set Tree. Cases were represented in an attribute-value form. Power Set tree was built to find the unique combination of attributes-values per each case, and use that unique combination as an index to the entire case. A solution to find the unique combinations for each case in a Case Base has been designed and built, and then use these unique combinations to build the Indexed Case Base (ICB). Using that approach, authors were able to make the retrieval process three times faster than the case base without index.

3.8.2 kd-Trees

Trees are classic examples of Index Structure discussed in the former section. The basic idea is searching top-down and eliminating all other branches when taking a specific branch (Richter and Weber, 2013). Trees are not directly related to CBR. Using a tree structure is a huge advantage, since it is possible to eliminate so many candidate cases by only one

step in the tree. In the previous section, the work presented by Kareem Amin et al. (El-Bahnasy, Amin, and Aref, 2014) used Power Set Tree (PST) for both indexing and retrieval, which has significantly improved the retrieval process. The challenge with building Trees comes from the way of how the tree is going to be structured and what attributes will be in the leaves and if the tree connectors will have weights or relevancy or not (El-Bahnasy, Amin, and Aref, 2014). In CBR, there are so many tree types have been used like kd-trees (Wess, K.-D. Althoff, and Derwand, 1994) and binary trees (Bentley, 1975) , but kd-Trees are the most common (Richter and Weber, 2013). The kd-tree is the index structure that one constructs in the preprocessing step. Building a tree can be a very exhaustive and time-consuming process (Wess, K.-D. Althoff, and Derwand, 1994). The average case effort (Mehlhorn, 1984) to generate a k-d tree is $O(k * n \log_2 n)$, for the worst case $O(\log_2 n)$. The average costs for retrieving the most similar case costs are $O(n)$. For small case bases, trees can be of high benefit to the retrieval process while with larger CBR systems, building the tree itself takes long time (El-Bahnasy, Amin, and Aref, 2014). Therefore, while using trees in the retrieval process has many advantages, the time and effort consumed to build them accurately are a drawback.

3.9 Adaptation

CBR systems are greatly relying on adaptation in order to be able to provide firm reasoning to a current state of the system. Since CBR systems relate to analogy, it is essential to be able to adapt past cases in order to provide profound human-acceptable arguments. However, this states a relative challenge in complex systems, since adaptation is getting more difficult. Adaptation has been always a hot and challenging area of CBR (A. Goel and Belen Diaz-Agudo, 2017). Advanced adaptation strategies have been introduced in the last years of CBR research, focusing on CBR in the big data era (Jalali and D. Leake, 2015). While retrieval is mainly focusing on narrowing down the candidate solutions based on similarity

measures, adaptation ensures that the most similar cases are being adapted correctly to the new case based on adaptation rules.

The literature provides a number of CBR systems where adaptation is prominent. Especially over the last years the Computer Cooking Competition (CCC), present in International Conferences (International Conference for Case-Based Reasoning have shown several systems that are keen to successive adaptation of their cases. Cooking CBR systems are using sets of ontologies along with their cases in order to be able to rectify and adjust their existing recipes to the user requirements.

The literature provides a number of CBR systems where adaptation is prominent. Especially over the last years the CCC, present among others at ICCBR 2009/2010/2011, has shown several systems that are keen to successive adaptation of their cases. Cooking CBR systems are using sets of ontologies along with their cases in order to be able to rectify and adjust their existing recipes to the user requirements.

An example given from the CCCs is the **Taaable** system, (Cordier et al., 2014) where a whole architecture has been designed and deployed to support the successful adaptation of recipes. Taaable is backboned by a semantic wiki, which works as a central module to manage all the available data and their inclusive knowledge along the system. On top of that, Taaable is using opportunistic adaptation knowledge discovery in an approach to gain interactive and semi-automatic learning of adaptation knowledge triggered by user-provided feedback.

Another example is ColibriCook software (DeMiguel, L. Plaza, and Díaz-Agudo, 2008), developed for the cooking domain. ColibriCook is an ontology-based CBR system which aims to the retrieval and adaptation of cookingrecipes. The systemis based on the robustness of jCOLIBRI2 (Belén Diaz-Agudo et al., 2007) CBR framework, which provided a basic ontology extension. The ontology was adapted accordingly to meet the requirements of the cooking domain.

DeepKAF as a framework is not focusing on adaptation, but rather retrieval. However,

the used ANN within **DeepKAF** can be reused to implement adaptation rules, and this part from the future work described in **Chapter 8**.

3.9.1 Adaptation Rules

As **Richter** mentioned in his book, (Richter and Weber, 2013) "adaptations are special kinds of actions, described formally by rules". A rigorous formalism is important for the systematic use of adaptation. Adaptation is a form of action that is done based on some preconditions. Those preconditions are the rules controlling what actions are going to be performed. An adaptation rule can be represented in the following form:

$$\phi_1 \wedge \phi_2 \wedge \phi_3 \wedge .... \wedge \phi_n \Rightarrow \textbf{Action}$$

where ϕ_n is the precondition.

3.9.2 Adaptation Approaches

In principle, there are three main types of adaptation: 1.***Structural adaptation,*** where rules are predefined and applied directly to cases, 2.***Derivational adaptation*** where adaptation rules are being derived and generated from the case base and 3.**C*ompostional Adaptation*** where the case solutions is obtained by combining elements from several similar cases, we combine with the most similar one (Chun-Guang et al., 2004). Structural adaptation techniques have been popular by predefined some adaptation rules to be used. However, with the complexity of data and application domains, pre-defining the adaptation rules cannot survive alone as an adaptation technique. It should be combined with other adaptation techniques in order to be able to propose accurate solutions for the new case (Jalali and D. Leake, 2015). As an example of Derivational Adaptation, the approach that Huan Li et al. introduced in (H. Li et al., 2007) where authors used a learning process to update their adaptation rules set with guidance from the domain knowledge experts. In another work by Hanney et al. (Hanney and M. T. Keane, 1997), authors proposed a technique which

learned adaptation knowledge from case comparison and applied the technique in a house price case-base. Another example of derivational adaptation was introduced by Jalali et al (Jalali and D. Leake, 2014). The authors introduced **AGCBM**, an approach to condensing the case base size for domains in which adaptation knowledge is generated from cases, in which retention decisions reflect both cases' usefulness as source cases and their usefulness for generating adaptation rules on demand.

Adaptation approaches are still a hot topic to research in CBR. In the last years, new approaches have been introduced that use ML and DL approaches to combine adaptation rules or generate new solutions that were not existing before.

3.10 CBR Systems and Applications

we need to do something for that. Due to the generalization of processes and the CBR- cycle portability across systems, the concept of CBR can be applied to a remarkable number of applications in a wide range of environments. This can be seen from the early days of CBR, in the early 1980s, where its application included a variety of fields like those of dynamic memory (Schank, 1983), analogy (Carbonell, 1983) and legal reasoning (Rissland, 1983). This work on CBR has been continued with Hammond and Collins on case-based planning (Hammond, 1986; Hammond, 1990; Hammond, 1989) and the work of Ashley and Rissland on HYPO legal reasoning system (Ashley and Rissland, 1987).

At the same time, CBR started being used in systems like the **CYRUS** computer system. **CYRUS** was a question-answering system that contained information regarding the life (travels, meetings) of the former United States Secretary of State Cyrus Vance. CBR has been used in the CYRUS system to represent the memory cases that were afterwards retrieved to answer a relevant question. Based on the implementation presented in CYRUS the systems of **CASEY** (Phyllis, 1988) and **MEDIATOR** (Simpson, 1985) were built. CASEY, a pioneer system in the application field of diagnosis, has been used for diagnosing heart problems on patients. Its CBR mechanism was based on the existing knowledge of known heart-problem diagnoses. MEDIATOR was specialized in mediation, trying to solve disputes by adapting existing dispute solutions over the new proposed ones. Michael Lebowitz has also used a CBR system to create the Integrated Partial Parser (IPP) which was "a computer system designed to read and generalize from large numbers of news stories" (Lebowitz, 1983). () Modern industrial approaches that use CBR can also be seen in the literature. An indicative example of those is the **CLAVIER** system. **CLAVIER** CBR system (Hinkle and Toomey, 1994) has been developed and used from Lockheed Corporate aircraft manufacturing company as an advisory system for their qualified personnel. The role of the system was to maximize their efficiency and ensuring the quality of composite

aerospace parts before finalizing them and sending them to the convection oven. Other systems like JULIA, an earlier system, has been designed to plan meals (Hinrichs, 1988; Hinrichs, 1989); KRITIK was designing mechanical assemblies by using a combination of CBR and MBR (Model-based reasoning)(A. K. Goel, 1989) whereas **CYCLOPS** has been used for landscape design (Navinchandra, 1988). Form Tool, (Cheetham, 2005) a decision support application based on CBR, was designed for the Plastic Industry to retrieve the specified color formulas that meet colors requests from customers (Plastic Color Matching tool). **ShapeCBR** a system designed for metal casting was created to "automate the process of creation and selection of cases to populate a CBR system for retrieval of 3D shapes to assist with the design of metal castings" (Miltos Petridis, Saeed, and Knight, 2010). **ShapeCBR** was using graph similarity algorithms towards the efficient retrieval of similar components from the case repository, based heavily on its CBR mechanism. Another family of applications that heavily uses CBR are the so-called help-desk applications. These applications are designed to support company customers by providing relative services, starting primarily with the contact security which is frequently needed by customers. Help may also be needed regarding the issues that they face. The importance of help desk applications can be rather significant since apart from supporting customers, it can work as a monitoring tool for the company to a large extent (Marcella and Middleton, 1996). The operational spectrum of help-desk applications can vary: starting from the detailed area of technical issues and going towards the areas of customer satisfaction, user interface experience, etc. Applications like the SMART: Support Management Automated Reasoning Technology (Acorn and Walden, 1992), the Appliance Call Centre automation at General Electric (Cheetham and Goebel, 2007) and the **HOMER**: CAD/CAM help-desk application at Daimler Chrysler(Bergmann, 2002a) were built with the focus on the customer help and support. CBR can be extremely useful in systems that require a direct answer to a number of predefined questions. Literature has already shown a number of systems using it towards that direction.

3.11 Architectures, Frameworks, Tools

The development of a simple case-based reasoning application includes a mixture of steps, such as case and background knowledge collection, modeling the appropriate case representation, implementing the retrieval functionality and implementing user interfaces. Compared to other AI methods, CBR reduces the work needed for knowledge acquisition and representation, which is undoubtedly one of the key factors for the market success of CBR applications. To standardize the good practices of implementing CBR domains, several frameworks and architectures were introduced with advantages and disadvantage for each approach. This section covers the most popular CBR frameworks and architectures that can be compared to **DeepKAF**. The idea of providing an architecture and common practices to build CBR systems is not new. It started with CBR becoming well-known as an AI sub-field, which brought the need to share experiences and practices to make CBR systems able to survive in the research field along with the industrial domain.

3.11.1 SEASALT Architecture

Agent-based systems technology has generated lots of excitement in recent years because of its promise as a new paradigm for conceptualizing, designing, and implementing software systems (Sycara, 1998). In (Bach, Reichle, and K.-D. Althoff, 2007) and (Reichle, Bach, and K.-D. Althoff, 2011) the **SEASALT** architecture is an application-independent architecture to work with heterogeneous data repositories and modularizing knowledge to be structured. It was proposed based on the CoMES approach to develop collaborative multi-expert systems. SEASALT aims to provide a coherent multi-agent CBR architecture that can define the outlines and interactions to develop multi-agent CBR systems (see Figure 3.3). The **SEASALT** team has applied it in (Reichle, Bach, and K.-D. Althoff, 2011) to travel medicine as part of the **docQuery** project. It was as a textual CBR application domain to showcase how SEASALT could be used. In (Pla et al., 2013) Albert Pla et al. have provided

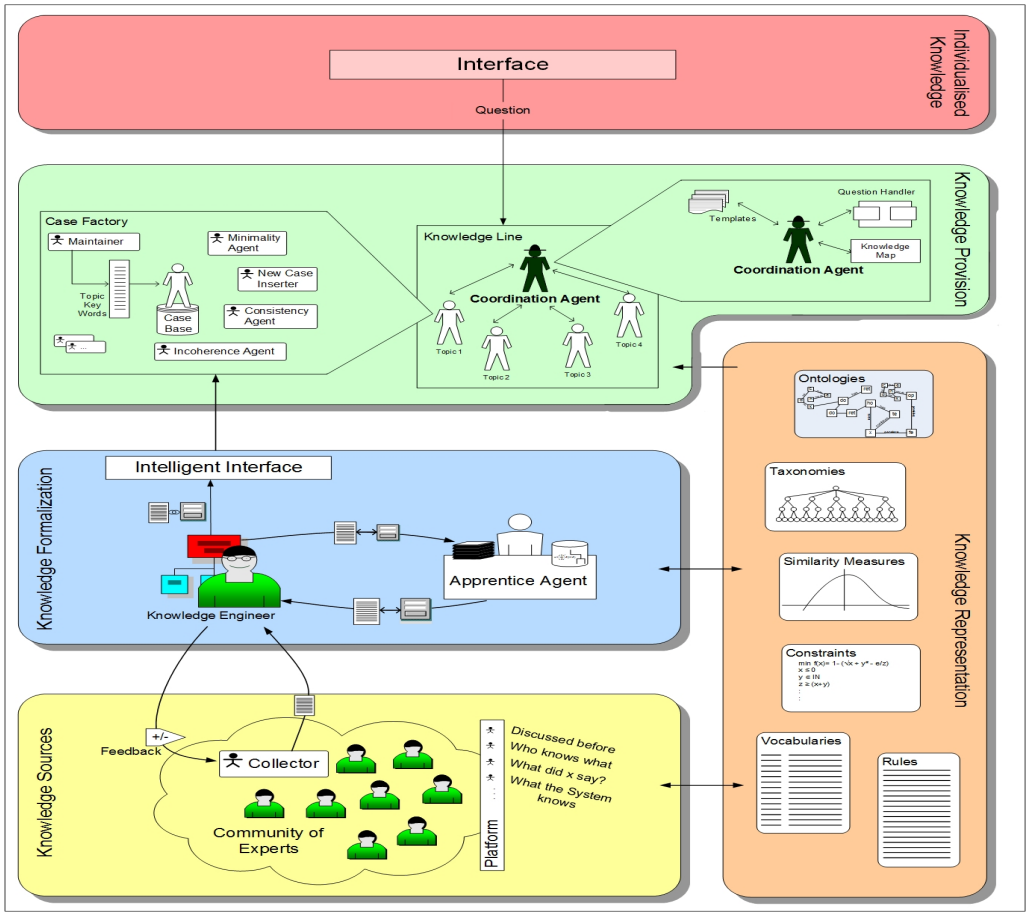

Figure 3.3 The SEASALT Architecture

a user-friendly tool for medical prognosis (eXiT*CBRv2). They proposed an innovative multi-agent system architecture, in which they have a coordinator agent that is responsible for receiving new cases, then pass it to **n** agents. Each agent is connected with case base to retrieve cases based on different retrieval calculations. Afterwards, they all pass the results again to the coordinator agent to assess and compare results and at the end it gives the final results. They illustrated the use of the tool through several experiments carried out with a breast cancer database, and they show how easy it is to compare distributed approaches that maintain naturally distributed clinical organization, compared to centralized systems.

Generally, CBR and MAS have proved efficiency with different successful distributed CBR systems. Currently, systems are getting more complex and agents need to be smarter to be able to deal with its environments. MAS bring a lot of advantages and benefits to

CBR, but also have a lot of challenges and issues that should be taken into consideration while building systems:

1. How do we design our algorithms to decompose tasks to agents and allocate problems to them?

2. If systems are widely distributed, how are agents going to communicate and what communication protocols will they use?

3. What if we lost the communication between agents?

4. How do we ensure that agents are working properly, and every single agent is doing its task in the perfect manner?

5. How do we troubleshoot issues across all agents?

SEASALT consists mainly of five layers, Knowledge Source, Knowledge Formalization, Knowledge Representation, Knowledge Provision, and Individualized Knowledge. Every layer contains several software agents designated for several tasks. The following paragraphs describe SEASALT in a bottom-up manner (Reichle, Bach, and K.-D. Althoff, 2011).

Knowledge Sources Layer Connecting to different data sources and being able to collect data is essential for any software. In addition, SEASALT relies on Web 2.0 platforms as main data sources rather than the traditional data sources such as databases and static web pages.

The SEASALT architecture is especially tailored for the acquisition, handling and provision of experiential knowledge as provided by the communities of experience and represented on Web 2.0 platforms. The SEASALT architecture may also provide external knowledge sources by equipping the individual collectors with database or web server protocols or HTML crawling capabilities.

Knowledge Formalization Layer In order for the collected knowledge in the knowledge sources layer to be easily usable within the knowledge line, the collected contributions have to be formalized from their original representation into a more modular, structured representation. This task is mainly carried out by the knowledge engineer. This Layer is big intersection point between **DeepKAF** and **SESALT**. **DeepKAF** uses several deep learning approaches in order to formalize the collected knowledge and make it usable with minimum effort from the knowledge engineer.

Knowledge Provisioning Layer The knowledge provisioning layer of SEASALT is carried out using the knowledge line method described in (Bach, Reichle, Reichle-Schmehl, et al., 2008). The fundamental concept of the Knowledge Line is the modularization of knowledge, analogous to the modularization of applications in the product line approach within Software Engineering. Within the SEASALT architecture, this knowledge modularization takes place with respect to the individual subjects represented in the respective knowledge domain.

Knowledge Representation Layer The knowledge representation in SEASALT has been created based on general resources like WordNet and further developed during the runtime of the application. **SEASALT**'s knowledge representation includes rules, vocabulary, ontologies and taxonomies. Since the multiple agents working on the community platform. The tools of the Knowledge Engineer and the CBR systems for the individual topic agents handle the same Information Domain(s). This does not only significantly encourage the preservation of the information model, but also promotes the interoperability of the individual components.

Individualized Knowledge Layer Any application that is built based on **SEASALT** has a web-based user interface. The web-based interface offers a semi-structured input in the form of different text fields used for entering information about the case. That interface

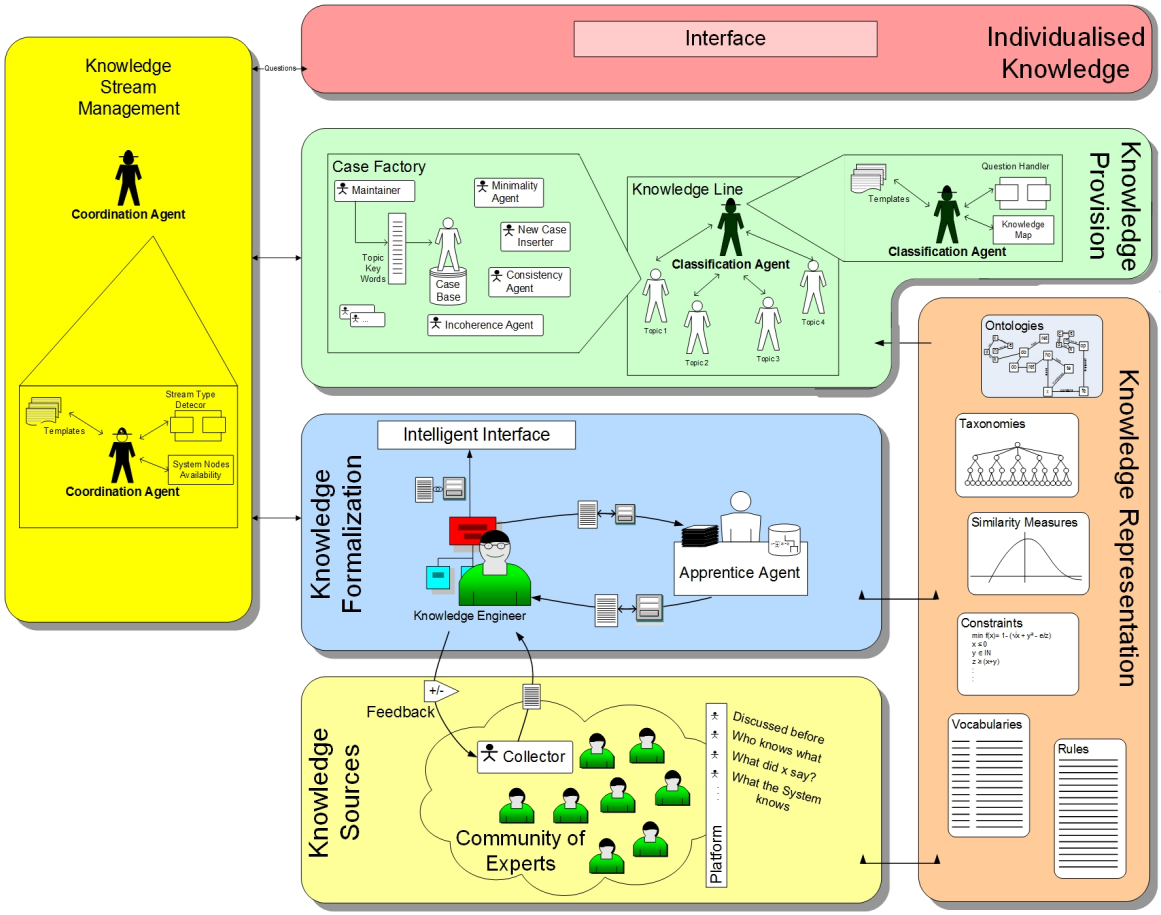

Figure 3.4 The SEASALT – Big Data Oriented Architecture

is used to get queries from the CBR system user and go through the Knowledge Provision Layer to get the most relevant results.

SEASALT is the heart of this **DeepKAF** and this thesis, and the idea of that thesis was to extend the SEASALT architecture with a Knowledge-stream layer (see Figure 3.4). **DeepKAF** can be integrated with **SEASALT** by adding more agents to handle the deep learning and data streaming parts.

The Knowledge Stream Management layer (see Figure 3.4 has two main tasks, the first is processing the streams of data coming from Knowledge Sources in real time, and the second is to give real-time analysis to data patterns found within the streams. The Knowledge Stream Management layer will contain software agents designated for the prescribed tasks.

One key innovation of the integration between **SEASALT** and **DeepKAF** is to enhance

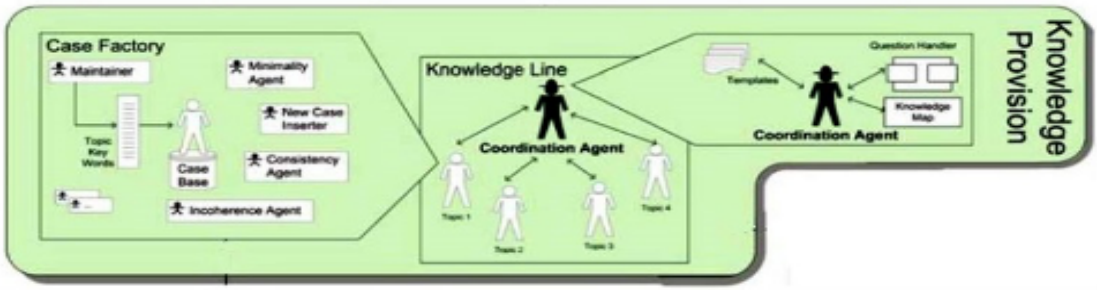

Figure 3.5 Knowledge Provision Layer

the CBR systems that are built based on **SEASALT** and be able to tackle problems that come with any Big Data implementation. **DeepKAF** adds more flexibility and scalability to **SEASALT** by applying the state of the art techniques to handle different types of data.

Figure 3.5 depicts the Knowledge Provision Layer in SEASALT, in which the coordination agent is connected with a number of topic agents. Every topic agent is considered as a standalone CBR system with its own case factory. Figure 2. Illustrates the changes made to the Knowledge Provision layer and the new Knowledge Stream Management layer. The Knowledge Stream Management layer is getting two types of input, one from the Knowledge Formalization layer that is related to new data insertion, and the other input is coming as a stream of questions from the Individualized Knowledge layer. According to our architecture, system nodes would be the available processing power – Node = Single Processing Power. The Knowledge Provision layer will be distributed across several nodes, and hence each node contains Knowledge Provision agents. The Coordination Agent will act as the system manager who is aware of all the system nodes and responsible for the whole system control. He will be the data tap that uses the underlying framework to distribute the incoming requests across the system nodes. Normally, I have two kinds of nodes, one for Queries processing to retrieve results and the second for New Cases processing. I can have up to N nodes in the system according to the hardware availability. The more nodes I have, the better performance I can get. In every node, I have a Classification Agent to classify the received data and assign it to the intended Topic Agent. Each Classification Agent is aware

of the knowledge map gathered from knowledge sources and classify the incoming requests according to predefined classes. Then, the Classification Agent assigns the request to the intended Topic Agent(s). The Topic Agent is performing queries to retrieve the most similar cases. Since I use distributed nodes in the hardware cluster, the Case Base will be replicated to avoid data integrity, using the replication channels to replicate data between the Case Base instances (see Figure 3.4). Since our Case Base will be distributed among several nodes, the Case Factory agents will be centralized. Therefore, the Case Factory will have only one instance that performs case maintenance on a single Case Base. Afterwards, the results will be distributed to the whole system nodes using the replication channels.

3.11.2 CBR-WIMS Framework

Any system that can intelligently monitor workflows has to be able to provide flexibility and agility in a continuously changing environment. A suitable architecture is needed to be able to accommodate that. As a plausible consequence of the above, a proposed architecture for a monitoring system has to be flexible, agile and easily adapted to a workflow managed process. Usually such architecture should be accompanied by a resilient software framework which could provide generic functionality. Subsequently, this functionality should be selectively adapted on demand, hence providing a modular software approach which at the same time is application explicit. In the case of business processes, such a framework should be able to accommodate different systems with equivalently different business rules. Modern enterprise systems are advanced, that is why a proposed architecture has to be equivalently radical in order to satisfy the escalated complexity of business processes at its utmost. In order to deal with the above stated needs of the intelligent business process monitoring, a new framework has been developed. The framework, named **Case-Based Reasoning Workflow Intelligent Monitoring System (CBR-WIMS)** (Kapetanakis, Miltiadis Petridis, Ma, and Bacon, 2009; Kapetanakis, Miltiadis Petridis, Ma, and Bacon, 2010; Kapetanakis,

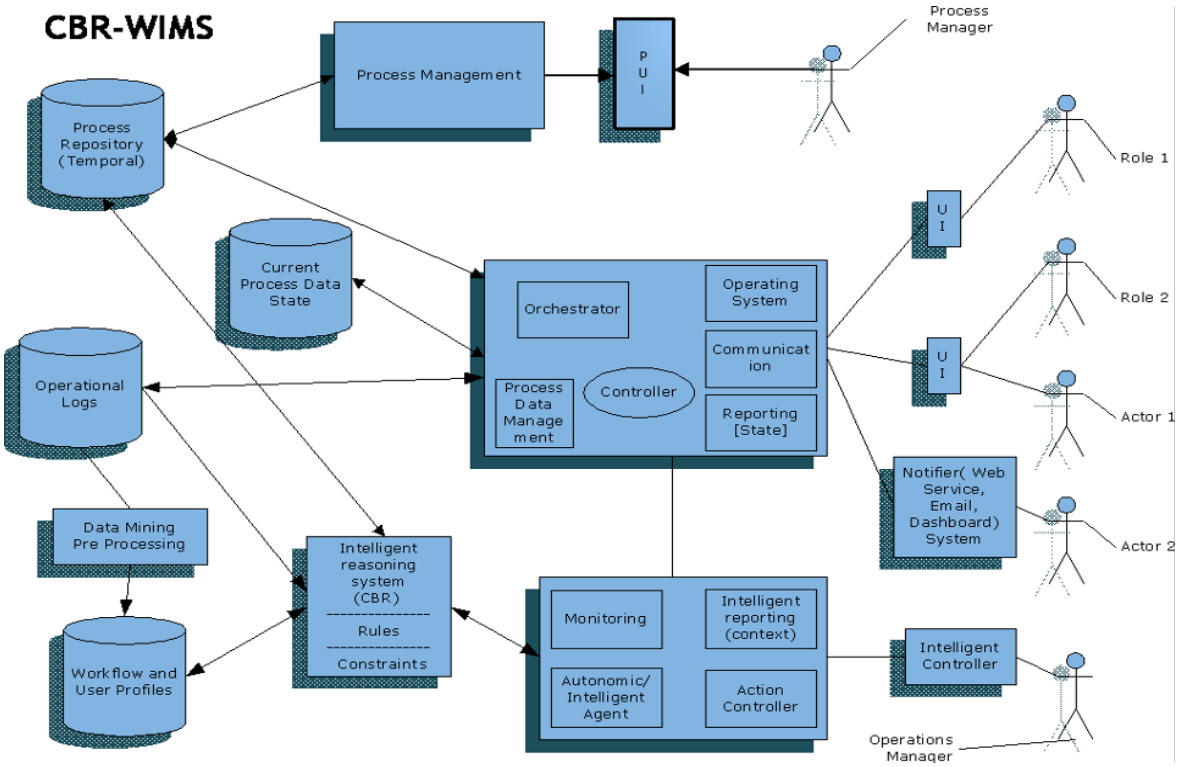

Figure 3.6 The Intelligent Workflow Management System Architecture

Miltiadis Petridis, Ma, and Knight, 2010), consists of a series of tools and services that are encapsulated and presented under one framework to its specialized users. The aim of the development of **CBR–WIMS** is to provide automatic monitoring, diagnosis and explanation to workflow managers and stakeholders (see Figure 3.6. The framework gives emphasis on a number of facets such as:

1. the identification of potential problems within a workflow
2. the analysis of workflow information
3. the retrieval of past information which is similar to an investigated case
4. the assembly of suggestions, recommendations that could lead to the restoration of the workflow state to an acceptable, stable (healthy) condition.

CBR-WIMS has been designed as a generic workflow monitoring framework which facilitates a number of flexible services, control modules and adaptors. **CBR-WIMS** can

offer an interoperable environment for workflow monitoring and diagnosis. This chapter has presented the architectural backbone and has revealed how its collaborative components could accommodate that. An alternative system orchestrated, choreographed and operated via a workflow has been used as a test bed for the evaluation of **CBR-WIMS**. **WIMS** have shown efficient in terms of its adaptation capabilities on the different system. The applicability of CBR-WIMS to different business processes has shown that it can establish intelligent workflow monitoring. An interesting finding from its application on BTS is that heedful knowledge acquisition leads to its reusability across workflows.

CBR-WIMS has revealed how it can be expanded, adapting its monitoring functionality across systems. However, the provided functionality was accompanied by a number of explanation elements that enhanced its monitoring, reasoning and diagnosis on workflow systems.

CBR-WIMSis able to explain the context or the probable cause of a problem through the clustering classification. This has shown to enhance the confidence of its users overall on making successful monitoring and diagnosis on particular cases. WIMS was called to explain its provisional judgements in more than one system that both of which were dealing with uncertainty. This can be seen from the conducted experiments, which demonstrated with success the explanation provision at both micro level (individual cases) and macro level (case justification among the case-base overall). Several challenges have been raised while attempting to offer interoperable explanation provision, but were dealt successfully with the application of diverse techniques.

3.11.3 COLIBRI

COLIBRI as platform provides a well-defined design architecture for a multi-annual experience CBR system, a reference to that architecture: the jCOLIBRI framework and several development tools which help users implement and share new CBR systems and components.

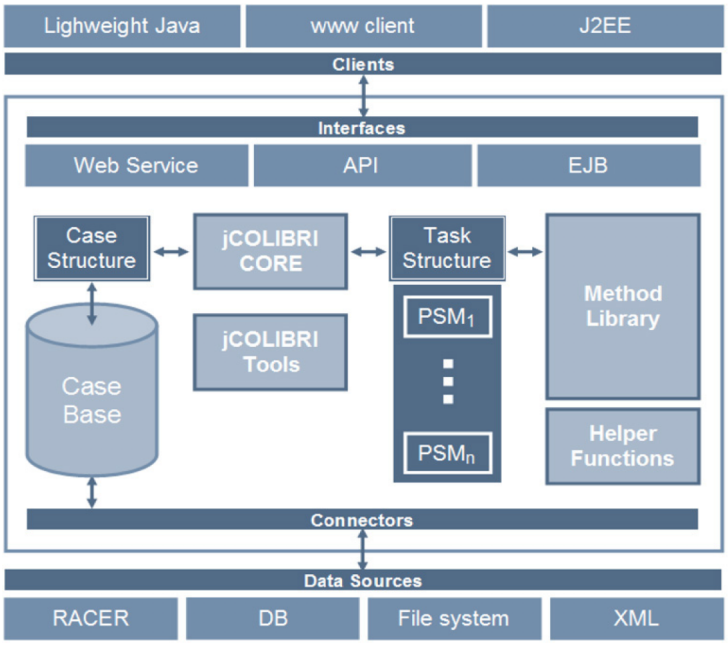

Figure 3.7 jCOLIBRI Framework Structure
Reference: (Belén Diaz-Agudo et al., 2007)

The methods were incorporated into the production sense of COLIBRI Studio. All instruments are built by the COLIBRI architecture of two layers (Bello-Tomás, González-Calero, and Díaz-Agudo, 2004). **COLIBRI** is designed to provide a collaborative environment, in which users can engage in the implementation of CBR applications. It's an open platform where users can contribute to various designs or components that other users can reuse (see Figure 3.7

COLIBRI installation or Interface based configuration tools allow the building of the CBR system without the need to write a line of code. Nonetheless, COLIBRI is flexible and could program new methods and integrate them into the framework if we were to construct a highly complex CBR system or if we wanted problem-solving methods that are unable to be used in the framework. As depicted in COLIBRI Architecture, the tool has several layers of building any CBR system to ensure consistency and persistence of any implemented system. It provides different connectors to structured and unstructured data sources, then a layer to build similarity measures. The tool also provides different ways to structure cases and

problem-solving, as explained in their publication (Bello-Tomás, González-Calero, and Díaz-Agudo, 2004). The following bullets explain in details the philosophy behind how COLIBRI provides an eco-system to build CBR applications.

1. CBR systems must provide secure and appropriate access to the stored cases with the increasing size of the case base. In two separate yet related concerns, COLIBRI splits the Case Base management problem: persistence mechanism through connectors and in memory organization. Various in-memory logins and data structures are given.

2. Cases can be depicted, depending on the system, as simple plain attribute-value cases, text cases or complex hierarchical structures (object-oriented), where their attributes are interconnected. Different similarity functions can be used to compare attributes of cases, depending on the case structure.

3. At the knowledge level, a CBR application is described in COLIBRI as a sequence of tasks that have to be resolved through the resolution or decomposition process. Methods of decomposition divide a job into several tasks. Task examples include Pre-Process Cases, Query, Recall, Reuse, Review, Retain, Similarity Calculators and many more. Their approach to PSM competent specifications (post-condition) and criteria (pre-condition) incorporates ontologies and offers all major advantages. For each mission, the designer of CBR configures the method that addresses them. First, it allows for formal specifications that add accurate meaning and support reasoning. Furthermore, it gives us valuable benefits in terms of reuse, as activities and methodologies can be shared by different systems.

4. COLIBRI provides a method library that includes at the knowledge level the operating level (i.e. implementation) of the problem-solving methods. If the library does not contain the required resolution method, a system designer must write Java code. The aid functions promote the development of new methods and functions of similarity.

5. Note that this is both the application generator and architecture for the CBR application constructed with COLIBRI. This is an application generator architecture. By specifying case structure, login settings and task structure, COLIBRI tools allow one to create a new application. In XML files, configuration data will be saved as an input to the CBR application's system heart. The framework core also plays a role in the design process because it can be interactively tested by a partially configured system.

Based on the understanding of how COLIBRI as a framework works to implement CBR systems, **DeepKAF** is providing the technical details of how to approach CBR systems in the industrial domain without forcing a structure like how COLIBRI works. **DeepKAF** can be considered as a complement to how COLIBRI works in general by providing the technical details and models that help in collecting data and building similarities.

3.11.4 myCBR

myCBR is an open-source, software development (SDK) tool for similarity-based retrieval. With myCBR Workbench, highly advanced, knowledge-intensive similarity measures can be modeled and checked in a powerful Interface and easily integrated into your own applications using myCBR SDK (Bahls and T. Roth-Berghofer, 2007). **myCBR** is a joint effort of the Competence Centre CBR at German Research Center for Artificial Intelligence (DFKI), Germany, and the School of Computing and Technology at University of West London (UWL), UK.

myCBR Features:

1. Powerful GUIs for modelling knowledge-intensive similarity measures

2. Perspectives: If you have used the integrated development environment eclipse, you are already familiar with this feature. The myCBR Workbench provides task-oriented configurations for modelling your knowledge model, information extraction, and case base handling.

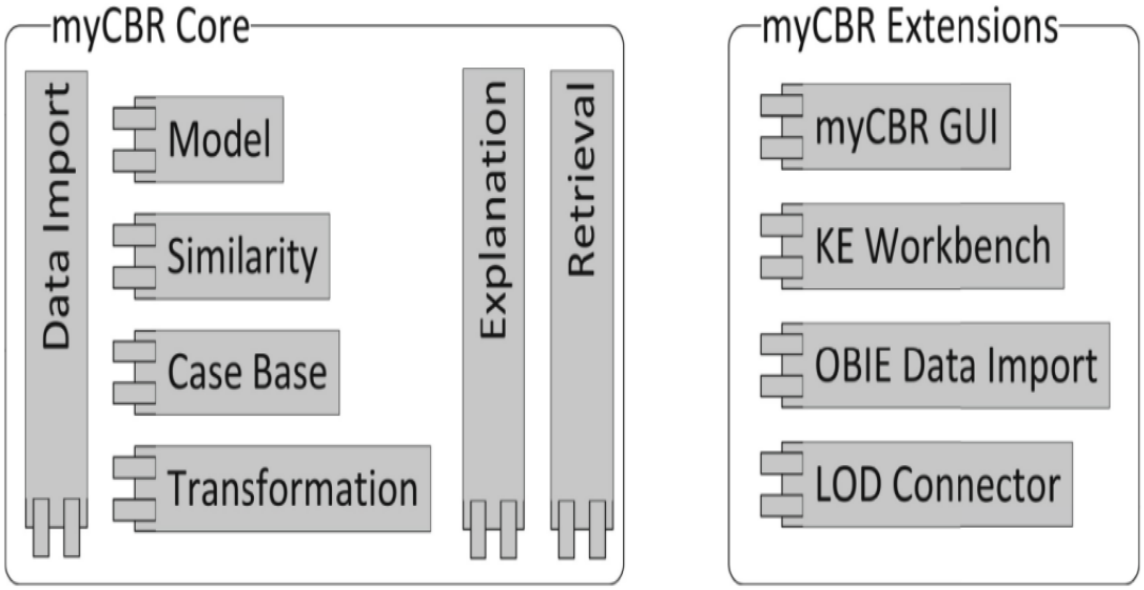

Figure 3.8 myCBR Architecture - Reference: (Bach and K.-D. Althoff, 2012a)

3. Similarity-based retrieval functionality

4. Extension to structured object-oriented case representations, including helpful taxonomy editors

5. Rapid prototyping via CSV

The last release of **myCBR** was in 2014 (myCBR 3) (Bach and K.-D. Althoff, 2012b). In myCBR 3, Workbench offers efficient GUIs for knowledge intensive similarity measures modeling. The Workbench also offers task-oriented settings for modeling the knowledge model, information retrieval, and case-base management. A similarity-based retrieval mechanism for knowledge model validation is included within the Workbench. Editing a knowledge model enables the ability to use standardized object-oriented case representations, including helpful taxonomy editors, as well as case import using CSV files. The myCBR 3 Software Development Kit (SDK) provides a simple-to-use data model that makes it easy to create applications on. The retrieval cycle as well as case loading, even from significantly large case

bases, is fast and thus makes seamless use in applications designed on top of the knowledge model of myCBR 3. Each attribute can have multiple similarity measures within myCBR 3. This function enables the exploration and testing of similarity measures in order to record variations. Because you can pick a suitable similarity measure via the API at runtime, you can easily accommodate divergent circumstances or various types of users (Bach and K.-D. Althoff, 2012b).

3.11.5 CAKE

Collaborative Agent-based Knowledge Engine (CAKE) provides unified access to knowledge available within an organization, and CBR technology is used throughout the system to distribute this knowledge to agents as required (Bergmann, Freßmann, et al., 2006). Since 2004, CAKE has been started as an ongoing research group project. Many R&D projects supported by various organizations have so far contributed to its growth. CAKE's principal components are (Bergmann, Freßmann, et al., 2006):

1. The agile workflow engine that interweaves modeling and execution of workflows and enables versatile collaboration between different workflow participants.

2. A knowledge engine that uses case-based process-based thought methods to retrieve and adapt semanticized workflows to enable business users to create and adapt workflows according to demand.

3. A common storage layer to access and update all workflow information consistently and to obtain web contents workflows.

4. A browser-based user interface for streamlined access to all CAKE elements.

CBR technology is used for two different purposes within CAKE. First, when a new job has to begin, it is used for the collection of appropriate jobs. Furthermore, when a conversation with an agent is initiated, it is used to pick suitable agents. To order to resolve

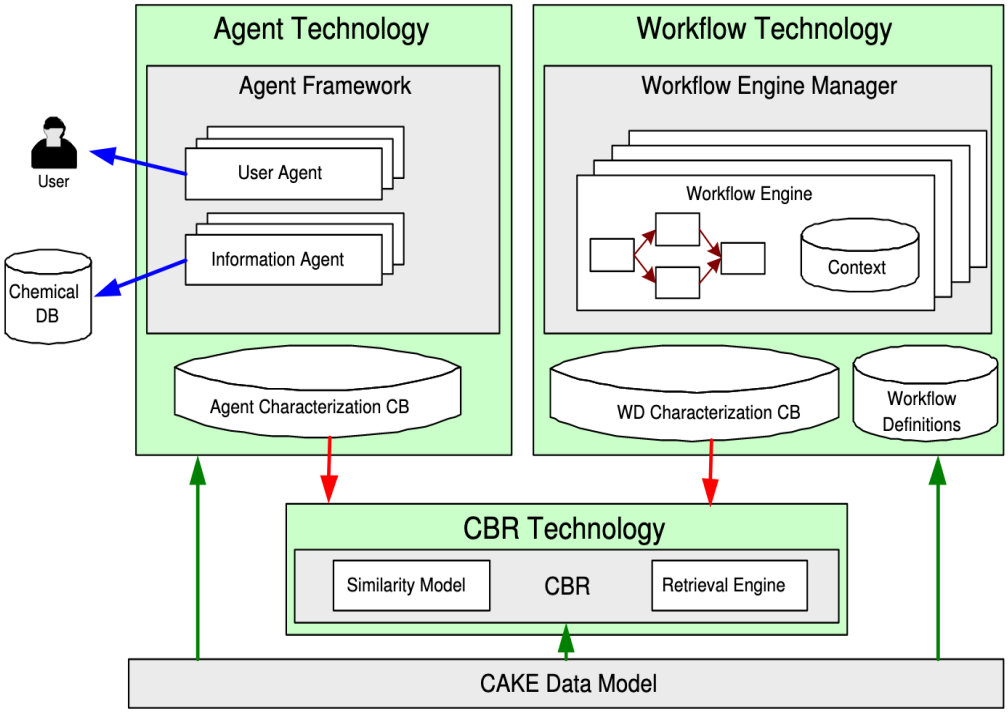

Figure 3.9 CAKE Architecture - Reference: (Bergmann, Freßmann, et al., 2006)

the need to rely on static job assignments for workers, CAKE uses CBR for these tasks. Job properties and agents are thus known as cases, whose characteristics are stored in the WD and Agent Classification Case Base (see Figure 3.9. In order to execute both retrieval tasks, CAKE implements the same generic CBR feature. The study of demands strongly supported the need for an overall domain ontology for a systematic definition of the relevant domains. Its aim is to attach meaning to different sources of knowledge, things and activities which have to be taken into account.

ProCAKE There is another more specific release of CAKE for process oriented CBR called ProCAKE (Bergmann, Grumbach, et al., 2019). **ProCAKE**, a generic framework for building structural and process-oriented CBR applications. The software is developed at the Department of Business Information Systems II at the University of Trier. ProCAKE integrates various similarity syntactics and semantics for different types of data. Most of the acts are defined officially in (Bergmann, 2002b). For instance, the measures used for numerical classes use linear, exponential and threshold functions while Levenshtein or regular expressions apply to String classes measures. For semanticized similarity evaluation, there are many taxonomic steps. A metric that graphs with an A * search algorithm is implemented for the NESTGraph class (Bergmann and Gil, 2014). Multiple algorithms are implemented for recovery: In order to facilitate retrieval with a broad case base, multiple MAC/FAC approaches and an A * parallel recovery method are introduced besides k-NN recovery. According to the research team, there is no other framework that provides the same functionality that ProCAKE is providing in the area of POCBR.

ProCAKE Features:

1. generic and domain-independent framework
2. tailored for developing structural and process-oriented CBR applications
3. comprehensive and extendable data types for knowledge modeling

4. various syntactic and semantic similarity measures

5. several retrieval algorithms (k-NN, MAC/FAC, . . .)

6. generic adaptation framework for integrating adaptation methods

7. modular and pattern-driven architecture

To summarize, CAKE is a powerful well-developed and maintained multiple-agent framework that can be used in different domains. The main intersection point between CAKE and **DeepKAF** is the similarity measures and how they are built. **DeepKAF** is more flexible in the area of domain knowledge acquisition using different deep learning models, then use the acquired knowledge to train other models to build similarity measures, while CAKE is counting on the traditional search algorithms like A* or a classification algorithm like KNN. On the other hand, CAKE is more complete and mature than **DeepKAF**, since it has an ongoing project for years with accumulated research experience.

3.12 Summary

Chapter 3 showed some CBR concepts that are a must to mention before going through the rest of this thesis. This chapter summarized CBR types and differences, with focus on Textual CBR that **DeepKAF** is targeting. The chapter showed different research in the literature defining the state of the art architectures and frameworks that have been developed over the past years in the CBR community. The literature presented in this chapter was chosen based on the popularity of the work in the CBR community, along with successful industrial implementations that used those architectures and tools. The main goal of this chapter was to highlight the concepts that **DeepKAF** will focus on in the next chapters, along with the main strengths that **DeepKAF** is adding, compared to the work presented in the literature.

Chapter Four

CBR, Big Data, and Deep Learning

Artificial Intelligence (AI) has been getting increasingly popular throughout the last decade. As big data is a key trend in the tech industry at the time this thesis is written, machine learning approaches seem adequately positioned to make predictions and automated suggestions based on large amounts of data. Common examples of machine learning can be seen in Netflix's algorithms for movie recommendations based on the user's favorite genre(s) or Amazon's algorithms for book recommendations based on previous purchases and others. In such a demanding and constantly evolving environment, CBR appears as a competing technique that can deal with the demands of a data-intensive era (Aamodt and E. Plaza, 1994; A. Goel and Belen Diaz-Agudo, 2017). Large-scale real-time data collection creates a need for advanced data acquisition, management, and rigid mechanisms for analytics.

Large-scale systems are part of most medium and large enterprises following an ongoing demand for building systems that are able to process data streams. Current big data strategies tend to process in-motion data and can offer a variety of scenarios to work with. The term big data refers to dynamic, large, structured and unstructured volumes of digitized data, generated from different sources that can differ substantially in formats (Labrinidis and Jagadish, 2012a). CBR is a problem-solving paradigm that is different from the major AI approaches since: Unlike other approaches that rely solely on the general knowledge of a problem domain or associate along inferred relationships between problem descriptors and

conclusions, CBR utilizes specific knowledge of previously experienced problem situations. CBR has been applied successfully in a variety of applied solutions across different domains. Recent AI trends clearly focus on data-driven methods. Yet, many decision support systems are regarded usually as "black boxes", since they provide little or no reasoning evidence for their processes (Herlocker, Konstan, and Riedl, 2001). CBR eliminates the "black-box" perspective since by design it is able to provide sufficient arguments for its outputs, allowing domain experts to reason upon its decisions.

The ability of CBR to reason upon individual examples and its inertia-free learning makes it a natural approach for big-data problems such as predicting on top of very large example sets (Smyth and McKenna, 1999). The literature shows increasing work on CBR and big data with the majority of cases on case base maintenance methods, aiming to reduce the case-base size while preserving competence (Smyth and M. Keane, 1997; Smyth and McKenna, 1999). Few CBR projects have focused on the case base size, considering scales up to a million of cases (Daengdej et al., 1996; Beaver and Dumoulin, 2013).

4.1 About this Chapter

As shown in **Chapter 3**, with the different approaches and architectures that the main goal was always to standardize the CBR implementation process along with some tools that can facilitate the similarity measures building and retrieval processes. However, none of those frameworks tackled any deep learning or big data frameworks that can leverage CBR implementations on large-scale. This chapter differentiates CBR from black-box approaches, which motivated the presented work in this thesis. The following sections present different neural network approaches that were extensively tested throughout this thesis, and advocate the selected approaches and techniques used in **DeepKAF**.

Details about each deep learning approach are provided because these approaches are the ones that eventually have been used in building **DeepKAF**. The deep learning architectures

used in **DeepKAF** are the outcomes of the implementation journey. Before selecting any of those approaches, an extensible literature survey has been done and presented on the three dimensions that **DeepKAF** is covering, namely CBR, big data, and deep learning. This survey goes through the research that has been carried out in these directions, presenting the rising impact of integrating these three methodologies and how they were used all together up to this point of time.

4.2 CBR VS Black-box Approaches

Pressing requests for precise and reliable classification tools in machine learning research has led to increasingly complex algorithms. Ensemble approaches and algorithms like Support Vector Machines (SVM) and neural networks have achieved an unpredictable level of complexity. These approaches and those like them, because they are not transparent in the logic behind the predictions they create, are commonly referred to as **"black-box"** Algorithms. The motivation behind most CBR systems is to provide some form of evidence or argument for any proposed solution. In (Nugent and Cunningham, 2005), authors showed the importance of CBR systems and the advantage of providing a proof for each proposed solution. Authors built a CBR explanation system for regression tasks and compared the results with other "black-box" regression approaches, and CBR excelled the other approaches in performance while keeping the ability to provide explanation. CBR is best suited in a lot of domains where experts will never follow an AI approach without having explanation.

CBR still suffers from the chronic complications when it comes to build CBR systems in industrial domains as described in **Chapters 1 and 4**, which **DeepKAF** is solving by incorporating "black-box" approaches within the CBR paradigm instead of avoiding using such approaches.

Black-box approaches have so many advantages that cannot be ignored, and that's why they are heavily applied in so many industrial applications, regardless of the inability to

provide accurate explanations. Therefore, we believe that **DeepKAF** as a framework can leverage CBR systems to be used in industrial application along with "black-box" approaches. This combination between CBR and black-box approaches brings the best of both worlds. CBR with its strength to make use of the historical data, and black-box approaches with the proven track of how to decode a domain knowledge.

4.3 Big Data

Big data systems have gained substantial focus in recent years and still continue having rapid development. Such systems cover many industrial and public service areas such as search engines, social networks, e-commerce sites and multimedia, as well as a variety of scientific research areas such as bioinformatics, environment, meteorology, and complex simulations of physics. Conceptually, big data are characterized by very large data volume and velocity, highly variety (diversity) in data types and sources, and stringent requirements of data veracity (fidelity) (R. Han et al., 2015).

Technically, a big data system can be characterized in terms of its data (4V's, as we will explain) and its related workflows on top of its input data. We first explain the 4V's properties as follows. **Volume -** Volume represents the amount/size of data in terms of units such as Terabytes (TBs), Petabytes (PBs) or Zetabytes (ZBs). Today, data are generated faster than ever. For example, about 2.5 quintillion bytes of data are created every day and this speed is expected to increase exponentially over the next decade according to research from the International Data Corporation (IDC). **Velocity -** Data arrival rate is real-time or almost real-time. **Variety -** Data can vary significantly in terms of format, including non-standard schema as well as BLOBs and CLOBs inside data. **Veracity -** one of the most challenging aspects in data generation. In many benchmarks such as GridMix , SWIM, HiBench and YCSB, the generation process of synthetic data is independent of real raw data (Huang et al., 2010; Y. Chen, Alspaugh, and Katz, 2012; Cooper et al., 2010).

4.3.1 Big Data, Big Challenges

Big data management has very big challenges and opportunities, since almost every-thing today is generating data. Big data management is not any more driven by computer scientists or researchers; it is a must for companies today to benefit from their data. Otherwise, they will not be able to survive in the current fast changing business environment. Big Data management plays a major role in the competition between companies.

Large enterprises like SAP, IBM, Google, Microsoft, SAS, and EMC are running now with maximum speed to build the most advanced big data platforms, not only from the software side, but also from the hardware one. They aim to attract new customers and help companies to gain the maximum benefit from their data. The new amount of data requires new, innovative technologies to be able to give the results in a reasonable time (Desarkar and Das, 2017). The big data term refers to dynamic, large, structured and unstructured volumes of data generated from (Labrinidis and Jagadish, 2012b):

1. Traditional data sources – includes the transnational data created from ERP systems, CRMs, web store transactions, etc.

2. Machine generated data – includes sensors data, smart meters, web logs, etc.

3. Social data – includes data generated from social networks like Facebook, Twitter, LinkedIn, etc.

4.3.2 CBR and Big Data

In most of the up-to-date CBR research, with regard to the increased data sizes, the primary focus has been on the compression of existing data rather than scaling-up. Considerable CBR research has focused on the efficiency issues arising from case base growth. As the case base grows, the swamping utility problem can adversely affect case retrieval times, degrading system performance (Jalali and D. Leake, 2015). CBR and big data collaboration

is an emerging topic, some researches have been carried out focusing mainly on case base maintenance methods, aiming to reduce the case base size while preserving competence (Smyth and M. Keane, 1997; Smyth and McKenna, 1999).

Few CBR projects have considered scales up to a million of cases (Daengdej et al., 1996; Beaver and Dumoulin, 2013; Agorgianitis, Kapetanakis, et al., 2017). The ability of CBR to reason from individual examples and its inertia-free learning makes it a natural approach to be applied to big-data problems such as predicting on top of very large example sets (Jalali and D. Leake, 2015). A capability to handle very large data sets could facilitate CBR research on very large data sources already identified as interesting to CBR, such as cases harvested from the Experience Web (E. Plaza, 2008), cases resulting from large-scale real-time capture of case data from instrumented systems (Ontanon et al., 2014), or cases arising from case capture in trace-based reasoning (Mille et al., 2013).

Since the growth of digital data is widely heralded. A 2014 article estimates that "Almost 90% of the world's data was generated during the past two years, with 2.5 quintillion bytes of data added each day" (G.-H. Kim, Trimi, and Chung, 2014). In (Xu and Tian, 2015/11), Yu-Hui X & Xiao-Yun Tian provided a CBR model NT-CBR based on the data mining technology NT-SMOTE. They have tried to solve the problems associated with enterprise risk man-agement and compared their results with different methodologies. The NT-CBR model used big internet data to do the forecasting of risks and give smarter and faster solutions to the risk. In (Jalali and D. Leake, 2015) Vahid Jalali & David Leake have initially developed ensembles of adaptation for regression (EAR), a family of methods for generating and applying ensembles of adaptation rules for case-based regression. That model suffered from high computational complexity and therefore they decided to go to big data techniques (Map Reduce) to improve their model performance, and they called it BEAR. BEAR uses MapReduce and Locality Sensitive Hashing (LSH) for finding nearest neighbors of the input query. It consists of two main modules: LSH for retrieving similar cases and EAR for rule generation and value estimation. As a conclusion, they got very promising results

that encourage them to perform bigger experiments to ensure that the model is reacting in the perfect manner.

From the CBR-big data literature, the CBR-big data systems can be classified as large-scale, real-time, stream management systems and hybrid techniques in combination with artificial neural networks. The following sections will summarize the literature in chronological order, some CBR systems developed over the years that tackle the current big data arising issues. Also, it classifies these systems according to their objectives and attempts to find out the extent up to which CBR and big data are used in these systems.

The following two classes describe the CBR systems that tackle the aforementioned Big Data V's:

1. **Large-scale CBR (Volume)**: Systems that focus on dealing with large case bases and trying to improve retrieval and/or indexing processes.

2. **Real-time CBR (Velocity, Veracity)**: Systems that focus on processing stream of data coming from different sources in real-time. These systems deal with problems like fast retrieval and pattern matching to carry out some proactive tasks.

Large-scale CBR

This section shows CBR systems that were designed to work with large case bases. I will show the main focus of the CBR systems and the problems that authors tried to solve.

RICAD — In RICAD (Daengdej et al., 1996), authors have built a solution that applies an efficient indexing and retrieving methodologies. They worked with a real world case base consists of 2 million incomplete insurance cases with thirty different attributes per case. Authors got inspired by database engines and how data is being queried, and aimed to build a system that can intelligently improve the retrieval processes with noisy data.

Bit-wise — Bit-wise (W.-C. Chen et al., 2002) is designed to work with large-scale CBR, and focused more on an indexing method that that is able to speed up the retrieving process for similar cases in case of large case bases.

PST-Indexing — In (El-Bahnasy, Amin, and Aref, 2014), I worked on an indexing approach that depends on the Power Set Tree (PST). I used the PST to find a unique combination of attributes for cases and used these unique attributes as a key for each case. Using the PST approach, I was able to build huge trees and search within billions of possible unique combinations. I was able to speed up the retrieval process four times better than the case base without the PST approach.

BEMD-GGD — In (Jai-Andaloussi et al., 2013), authors focused on analyzing medical images and built a CBR system based on a distributed Hadoop computing environment. This CBR system (BEMD-GGD) is used for content based image retrieval. Two methods were proposed to characterize the numerical content of medical images: the first method is BEMD-GGD and the second method is BEMD-HHT. The experiments showed that the MapReduce model is more efficient when the target image data is large.

4DSS The 4DSS (Marling et al., 2015) is a prototypical hybrid-CBR system that aims to help T1D patients achieve and maintain good BG control. As cases and data have accumulated over ten years, the research emphasis has shifted toward using the accumulated data to build machine learning models for BG prediction. Authors mentioned to the potential of collecting data in the future from fitness bands that can help in improving the prediction accuracy.

BEAR The work presented in (Jalali and D. Leake, 2015) illustrated the practicality of a big-data version of ensembles of adaptation for regression, implemented in BEAR, which uses MapReduce and Locality Sensitive Hashing for finding nearest neighbors of the input query.

IDGARD In (Bedué et al., 2015), authors have presented a novel approach to integrate a high security cloud storage (sealed-cloud) in Business Process Management (BPM). A new model has been implemented for using a sealed-cloud multi-media data storage for their workflow contents. Authors focused on building a scalable CBR approach and therefore, they focused on the cloud technologies.

WHAP (M. L. Han et al., 2016), WHAP implementation tackled the retrieval problem with large case bases. Authors implemented WHAP. A profiling system that uses CBR. Authors verified WHAP's usefulness by analyzing large scale of web defacement cases including North Korean hacker's attacks against South Korea, and unveiling a relationship between those attacks and another set of attacks against Sony Pictures Entertainment.

EACH (Jalali and D. Leake, 2017) have presented a different CBR approach based on learning itineraries. Learning itineraries are abstract representations of the user interactions that consider the order in which the problems were solved. In this work, authors have described a case-based recommender that leverages the implicit learning itineraries that emerge from the online judge submissions.

Smart-grid In Smart-grid (Troiano, Vaccaro, and Vitelli, 2016), authors have built a solution based on CBR to optimize smart grid operations. The solutions acquire real time data from grid sensors and analyze them based on the historical data. Due to time constraints, the search of similar patterns requires to face the large size of the historical smart grid data, which increase dynamically once new measured information is available. Therefore, they used the big data technologies to help them achieve the system purposes.

The aforementioned CBR systems were focusing mainly on building a large-scale CBR system. These methodologies have applied different approaches to be able to handle the problems that will come with any large-scale CBR system. The research presented in (Agorgianitis, Miltos Petridis, et al., 2016) argues that the current perspective of distribution in CBR does not take into consideration the importance of data volume as a key prerequisite in the integration of distribution in CBR for business process workflows. The authors proposed a new categorization of distributed CBR systems where the data volume aspect is the focus of the distribution effort, introducing a number of prerequisites.

Real-time CBR

This section focuses on the CBR systems that focused on real-time analysis and cases retrieval. The following systems are the best to my knowledge that showed advanced approaches in handling the real-time CBR systems efficiently.

CBR-RTS	CBR-RTS (Coello and Santos, 1999) has a modular architecture that easily supports evolution. The current structure of the case base and the corresponding retrieval algorithm seem adequate for case bases storing a moderate number of small cases. New organizations and retrieval strategies might have to be considered when dealing with case bases storing a high number of complex cases. The CBR-RTS architecture aims to build CBR systems that supports real-time retrieval. CBR-RTS focused on indexing the case base in order to perform fast retrieval.
D-HS+PSR(II)	In (Patterson, Galushka, and Rooney, 2003), Authors have presented two novel indexing schemes called DHS and D-HS+PSR(II). D-HS is based on a matrix of cases indexed by their discretized attribute values. D-HS+PSR(II) extends D-HS by combining the matrix with an additional tree-like indexing structure to facilitate solution reuse. The experiments showed accuracy, speed and ability to facilitate efficient real-time maintenance of retrieval knowledge as the size of the case base grows.
N/A	The work presented in (Floyd, Davoust, and Esfandiari, 2008) focuses on spatially-aware systems such as mobile robotic applications and the particular challenges in representing the systems' spatial environment. They select and combine techniques for feature selection, clustering and prototyping that are applicable in this particular context. The results demonstrated that prepossessing such case bases can significantly improve the imitative ability of an agent.

ExPeCo PeFoot	ExPeCo is a multi-agent CBR system (De Loor, Bénard, and Chevaillier, 2011). Authors have proposed a solution that stems from arborescent case bases, which enable the similarity between a target situation and all the cases in the base to be calculated at any time. In accordance with psychological considerations, the longer the time allowed, the better the evaluation of a situation. Some preliminary tests confirmed the credibility of the resulting behavior in CoPeFoot.
SIEVE-STREAMING	In comparison to the former papers that focus on the retrieval process. In (Y. Zhang, S. Zhang, and D. Leake, 2017), authors experiment within this paper, authors have extensively evaluated SIEVE-STREAMING algorithm on a Travel Agent Case Base benchmark within the CBR community, with comparison between several traditional methods.
DrillEdge	DrillEdge is a prominent solution in the area of real-time CBR systems. It has been developed and improved by different researchers (Bach, Gundersen, et al., 2014; Gundersen et al., 2013). DrillEdge has showed an advanced technique for automatic case capturing in the oil and gas domain. Authors have showed how the acquisition of cases in a knowledge intensive application can be supported by event detection and clustering. They introduced a method that automatically scans oil-well drilling logs for problematic situations and marks a case window on time series data. They evaluated their approach against domain experts doing the same task and found out that their approach reliably chooses correct areas for a mechanical stuck pipe that can be refined by experts.

N/A	In (Wender and Ian Watson, 2014), authors have presented a navigation component based on a hybrid-CBR and Reinforcement Learning (RL) approach for an AI agent in a Real-time Strategy (RTS) game. Spatial environment information is abstracted into a number of influence maps. These influence maps are then combined into cases that are managed by the CBR component.
CBR-FTIMS	In (Adedoyin et al., 2016), authors have proposed a multi-intelligent fraud detection system using logistic regression, neural network, and CBR. To prove the efficiency of their method, they used synthetic simulated data in evaluating their performance. The recognition performance shown by LR classifier is better compared to NN and CBR, with a steady increase in precision, sensitivity and specificity as the percentage ratio for the training and test data was varied.
O-MaSE	In (Rekik, Elkosantini, and Chabchoub, 2017), authors have introduced a new multi-agent architecture for the real-time container stacking in seaport terminals is presented. The allocation structure is based on CBR. In this context, a CBR system is developed and integrated in the proposed MAS. The proposed approach allows also the control of the storage system in a real-time manner by including the different unexpected events and disturbances that may occur during the allocation process.

4.4 Word Embedding Models and Why they are important

Deep learning models are not able to process strings or plain text. They require numbers as inputs to perform any sort of job, classification, regression, etc. Many current NLP systems

and techniques treat words as atomic units, therefore, in order to apply a deep learning model to NLP, we need to convert words to vectors first (Amin et al., 2018a). Word embedding is the process of converting text into a numerical representation for further processing. The different types of word embeddings can fall into two main categories:

1. **Frequency-based embedding (FBE)**:

   FBE algorithms focus mainly on the number of occurrences for each word, which requires a lot of time to process and exhaustive memory allocation to store the co-occurrence matrix. A severe disadvantage of this approach is that quite important words may be skipped since they may not appear frequently in the text corpus.

2. **Prediction-based embedding (PBE):**

   PBE algorithms are based on neural networks. These methods are prediction based in the sense that they assign probabilities to seen words. PBE algorithms seem the present state of the art for tasks like word analogies and word similarities.

PBE methodologies were known to be limited in their word representations until Mitolov et al. introduced Word2Vec to the NLP community (Mikolov, K. Chen, and Corrado, 2013). Word2vec consists of two neural network language models: A Continuous Bag of Words (CBOW) and skip-gram. In both models, a window of predefined length is moved along the corpus, and in each step the network is trained with the words inside the window. Whereas the CBOW model is trained to predict the word in the center of the window based on the surrounding words, the skip-gram model is trained to predict the context based on the central word. Once the neural network has been trained, the learned linear transformation in the hidden layer is regarded as the word representation. In **DeepKAF**, skip-gram model is being applied since it demonstrates better performance in semantic task identification (Altszyler, Sigman, and Fernández Slezak, 2016).

Text Pre-Processing

In the text pre-processing stage, raw text corpus preparation tasks are taking place in anticipation of text mining or NLP. In **DeepKAF** ticket management implementation, we trained our Word2Vec model over the ticket corpus to build an embedding model that will be used in the similarity measures. As any text pre-processing tasks, we have two main components: 1. Tokenization, 2. Normalization. Tokenization is a step which splits longer strings of text into smaller pieces, or tokens. Normalization generally refers to a series of related tasks meant to put all text on the same standard level: converting all text to the same case (upper or lower), removing punctuation, converting numbers to their word equivalents, and so on. Normalization puts all words on equal footing, and allows processing to proceed uniformly. Normalizing text can mean performing a number of tasks, but for our approach, we will apply normalization in four steps: 1. Stemming, 2. Lemmatization 3. Eliminating any stopping words (German or English) 4. Noise Removal (e.g., greetings and signatures). In essence we can consider the Word2Vec model or any other model that could be built as a substitution to the traditional taxonomies.

4.4.1 2Vec Models

2Vec models is a family of different Word Embeddings models that are being used for different NLP problems. In this section, three 2Vec models are going to be presented and explained because they are the main models that are being used in **DeepKAF**

Word2Vec

Word2Vec was introduced by Mikolov in 2013 (Mikolov, K. Chen, and Corrado, 2013). Word2Vec is a shallow two-layer neural network that processes text by giving a numerical representation, "vectorizing" for each word. This takes a broad corpus of terms as its input and creates a vector space, generally several hundred dimensions, with a corresponding vector

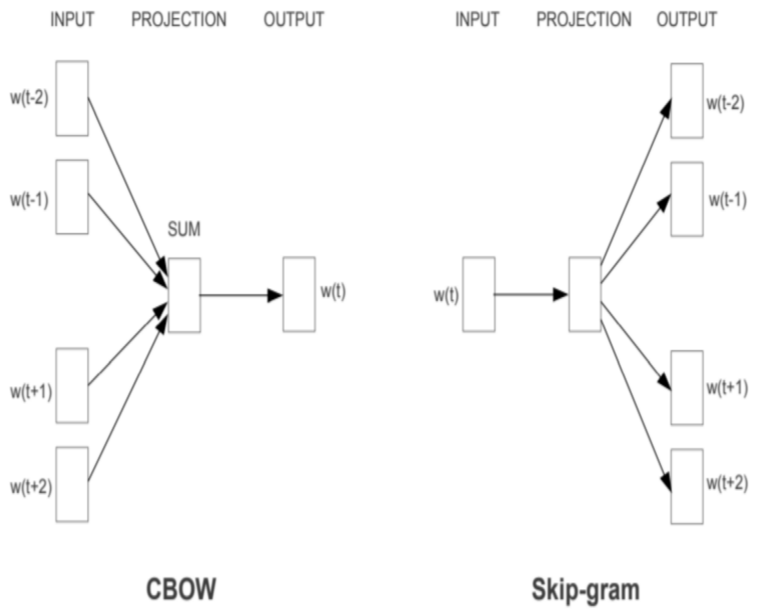

Figure 4.1 CBOW VS Skip-Gram
Source: (Mikolov, K. Chen, and Corrado, 2013)

of space allocated to each specific word in the corpus. Word vectors are arranged in vector space such that terms that share similar meanings in the corpus are placed in close proximity to each other in space. Word2Vec is an especially computationally powerful predictive model for learning word embedding from raw text.

Although Word2vec is not a deep neural network, it transforms text into a numeric form that deep neural networks can understand. Word2Vec representation is built using two popular algorithms: 1. CBOW and 2. Skip-gram model (see Figure 4.1). There are two primary teaching methods, the Distributed Words Bag and the skip-gram model. One involves predicting context terms using a core phrase, while the other involves predicting the term using context words.

Continuous Bag-of-Words The false function in CBOW is very similar to skip-gram, in the sense that we also take a pair of words and show the model that they co-occur, but instead of inserting the mistakes, we apply the input terms to the same target word. The

size of the hidden layer and the output layer will stay the same. Only the size of the input layer and the calculation of hidden layer activation functions will change, if there are four context words for a single target word, there will be four input vectors of 1xV. Each will be multiplied by returning 1xE vectors to the hidden VxE layer. All four 1xE vectors will be combined as an element-wise to achieve the final activation of the layer, which will then be fed to the softmax activation function.

Skip-gram For skip-gram, an alternative to "CBOW" rather than combining background terms, each is used as an example of pairwise instruction. That is, instead of a single CBOW example, such as [predict 'ate' from average('The', 'animal',' 'the',' 'mouse')], the network is faced with four skip-gram examples [predict 'ate' from 'The'], [predict 'ate' from 'cat'], [predict 'ate' from 'the'], [predict 'ate' from 'mouse']. (The same spontaneous window-reduction happens, but half the time it will be just two samples of the closest words.)

Impressions about CBOW and Skip-gram **Skip-gram:** fits best with a limited volume of training data, describing only unusual terms or phrases.

CBOW: many times quicker than skip-gram preparation, marginally higher accuracy for repeated phrases.

It all depends on the use case and the available data to decide which approach to use. Best case would be to build both models to benchmark and continue with the one that has higher accuracy.

Sequence2Sequence (seq2seq)

The Seq2seq model was introduced by Google in 2014 (Sutskever, Vinyals, and Le, 2014). Seq2seq model is designed to compare a fixed-length input with a fixed-length output, where the length of input and output can vary (see Figure 4.2). There are so many popular applications powered by the seq2seq model, for example, Google Translate started using

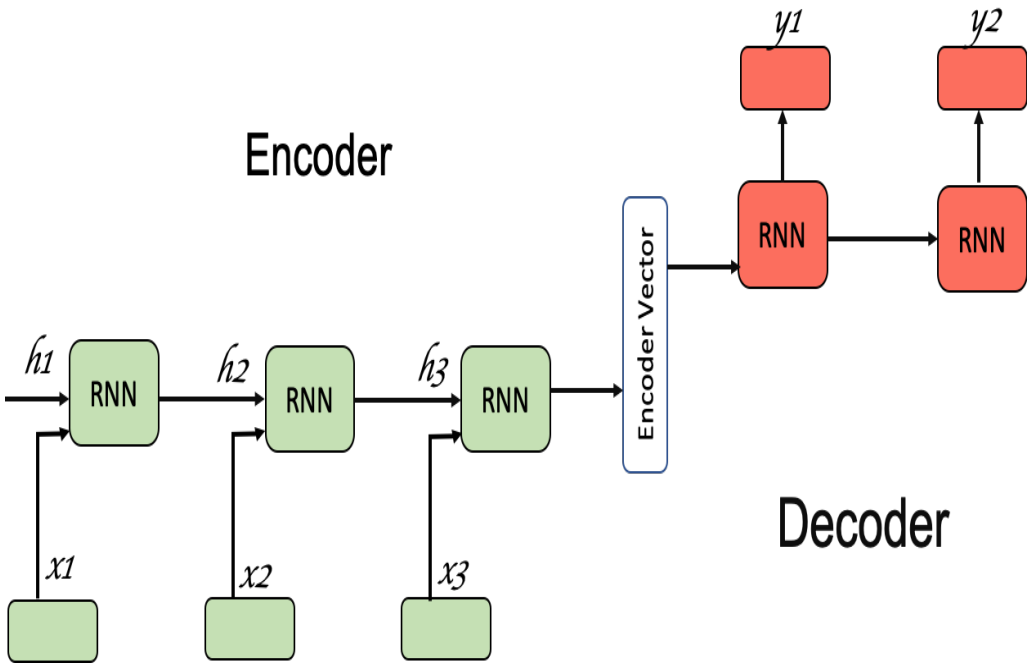

Figure 4.2 Seq2Seq Model

such a model in production in late 2016.

The Seq2seq model consists of three parts: encoder, intermediate (encoder) vector, and decoder. The encoder collects the background of the input sequence in the form of a hidden state vector and sends it to the decoder that produces the output sequence. As the function is focused on the list, both the encoder and the decoder use some types of RNNs, LSTMs, GRUs, etc. The hidden vector can be of any dimension, but in most situations it is used as a power of 2 and a large number (256, 512, 1024) and will in any way reflect the complexity of the full sequence as well as the scope (Sutskever, Vinyals, and Le, 2014).

The outputs are being calculated using the hidden state at the current timestamp together with the corresponding W(S) weight. Softmax activation function is then used to construct a probability vector that will help to evaluate the final outcome (e.g., word in question-answer problem).

The strength of the seq2seq model lies in the fact that it can map sequences of different lengths to one another. As any sequences in different languages, inputs and outputs are not

identical and their lengths can differ. This opens up a whole new collection of problems that can now be solved by such an architecture (Dai et al., 2017). Despite the strength of the seq2seq model, it did not work very well on a mixed-language text as per the case in **DeepTMS** which is one of the **DeepKAF** applications that will be explained in details in **Chapter 6**.

Doc2Vec

The idea behind Doc2vec is inspired by Word2vec and was introduced by the same team (Le & Mikolov) in their publication (Le and Mikolov, 2014a). In other words, Paragraph Vector (Doc2Vec) is intended to be an extension of Word2Vec such that Word2Vec learns to project words into a latent d-dimensional space, while Doc2Vec aims to learn how to project a text into a latent d-dimensional space. In Word2vec, the embeddings model is built based on that words retain a logical structure, but documents do not have any logical structures. To solve that problem, another vector (Paragraph ID) needed to be added to the Word2vec model. That's the only difference between **Word2vec** and **Doc2Vec**. There are two document embeddings models from Paragraph Vector (more popularly known as Doc2Vec): 1. Distributed Memory (PV-DM) and 2. Distributed Bag Of Words (DBOW)

4.4.2 fastText

fastText is another popular method to construct word embeddings that was created by Facebook's AI Research (Joulin et al., 2016). fastText is considered as an extention to Word2Vec and Miklov, who introduced Word2Vec, is the main contributor to the fastText concept as well. Unlike Word2Vec, instead of learning vectors for words directly, fastText represents each word as an n-gram of characters. So the vector for a word is made of the sum of this character n-grams. The character n-gram approach helps to capture the meaning of shorter words and enables the embeddings to recognize suffixes and prefixes. Once the

word is represented using the character n-grams, the skip-gram model is trained to learn the embeddings. This model is known to be a model word bag with a sliding window over a word, since no internal structure of the word is taken into account. As long as the characters are within this frame, the order of the n-grams does not matter.

fastText is working well with unusual words. And even though a term had not been used during preparation, it could be broken down into n-grams to get its embeddings.

Word2vec on the other hand fails to have any vector representation of terms that are not in the standard dictionary. This is a great advantage of fastText over Word2Vec.

4.4.3 Word Embeddings - Conclusion

Word embeddings is an ongoing research field that aims to find better word representations than the current ones. There are different word embeddings available that can be used to solve many NLP tasks. Over time, the embeddings have become broad in number and more complex. There is no one model that can solve all problems, but rather a model that fits the best to a specific problem with specific characteristics. In the word embeddings sections, many word embeddings approaches have been introduced with a comparison of how they work. In **Chapters 5 and 6**, the word embeddings layer in **DeepKAF** is going to be explained along with the models that have been used to benchmark the experimental results' accuracy.

4.5 Deep Learning Architectures

Deep learning algorithms and applications are growing fast. The deep neural network architecture has a proven record of successful implementations in so many domains. The main advantage is that there is always various models and architectures that can fit any data type. This section explains the most popular deep neural networks (CNN, RNN, and LSTM) with

focus on Siamese Networks architecture as it is going to be used within **DeepKAF**.

4.5.1 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (ConvNets or CNNs) are a sub-group of deep neural networks that have proved to be very powerful in fields such as image recognition and classification (Oord, Dieleman, and Schrauwen, 2013). Additionally, ConvNets have been succesfully applied in many other AI applications like recommender systems, natural language processing (Collobert and Weston, 2008), and financial time series (Tsantekidis et al., 2017).

4.5.2 Siamese Network Architecture

Siamese networks usually contain two or more identical sub-networks that share the same configuration with the same parameters and weights (see Figure 4.3. Siamese Neural Networks (SNNs) are known for their ability to find similarities or relationships among distinct objects. Typically, two identical sub-networks are used to process similar inputs, and another module will take their outputs to conclude with a final output (Bromley et al., 1993a). We chose SNN as our experimental setup since they provide the following advantages:

1. They are more robust compared to Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) in processing complex text.

2. They can provide better text embeddings.

3. By sharing weights across sub-networks, they reduce the number of parameters to train for, which in turn means less data required and less tendency to over-fit.

During building **DeepKAF**, several Siamese Networks have been tested, but Siamese Manhattan LSTM Model outperformed all the tested models.

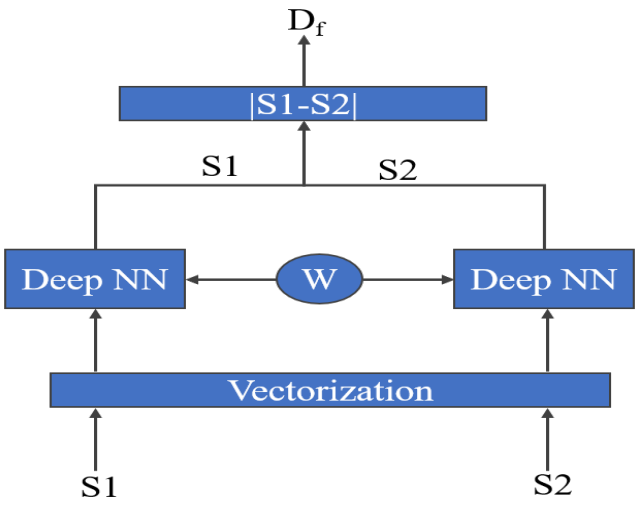

Figure 4.3 Siamese Network Sample

4.5.3 Deep Autoencoders

Autoencoders are artificial neural networks capable of learning efficient representations on data inputs, called codings, in an unsupervised way. Typically, these codings have a substantially lower dimensionality compared to their input(s) (Géron, 2017) by 'interpreting' them into a condensed output via latent representation views. Autoencoders were initially proposed as a method for unsupervised pre-training in (Ballard, 1987). Traditionally, autoencoders were used for data dimensionality reduction or feature extraction. Denoising by following an autoencoder approach was initially introduced by LeCun in (Lecun, 2001) as an alternative to Hopfield neural networks (Hopfield, 2018).

Autoencoders are a kind of neural network designed for dimensionality reduction; in other words, representing the same information with fewer numbers. The basic premise is simple — we take a neural network and train it to put out the same information it is given. By doing so, we ensure that the activation function of each layer must, by definition, be able to represent the entirety of the input data (if it is to be successfully recovered later on). If each layer is the same size as the input, this becomes trivial, and the data can simply be copied over from layer to layer to the output. But if we start changing the size of the layers, the

network inherently learns a new way to represent the data. If the size of one of the hidden layers is smaller than the input data, it has no choice but to find some way to compress the data.

Figure 4.4 shows a typical autoencoder architecture. However, modern applied autoencoder networks can have a substantially more complex layer topology.

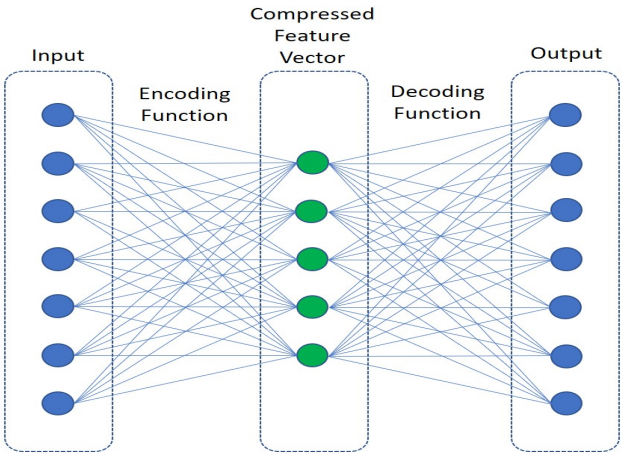

Figure 4.4 Autoencoders Architecture

Typically, an autoencoder architecture comprises three main components:

C1: An Encoding Function: A function that converts the network's input to an internal representation, ultimately reducing the input to a latent view representation.

C2: A Compressed Feature Vector (Latent View Representation): Latent view represents the lowest level space in which the inputs are reduced and information is preserved.

C3: A Decoding Function: A function that mirrors the encoding function and reverses the internal representation to formulate a readable output.

4.5.4 Autoencoders VS Word Embeddings

Both autoencoders and word embedding models have conceptual similarities in terms of how both techniques do the dense representation of text, and they are both fully connected feed forward networks. However, they have several differences in terms of their architecture,

learning method and input data. These differences position each technique to different areas, as problem solvers, since their application performance can vary significantly. **Table 4.1** shows a detailed comparison matrix between Word Embeddings models and Autoencoders.

Criteria	**Autoencoders**	**Neural Word Embeddings**
Architecture	A Deep Autoencoder (DAE) is composed of two, symmetrical deep-belief networks. Typically, they comprise four or five shallow layers representing the encoding, and a second set of four or five layers that make up the decoding.	Neural Word Embeddings (NWEs)are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words or phrases. NWEs can model a large corpus of text as input and can produce a vector space of several hundred dimensions. Throughout the process, each unique word in the corpus being assigned a corresponding vector in the space (Mikolov, K. Chen, and Corrado, 2013)
Learning Type	DAEs are unsupervised learning techniques that use the input dataset as also the output label. Their design focuses on input reconstruction	NWEs are an unsupervised learning technique which uses a corpus of unlabeled text to reconstruct linguistic contexts of words or sentences (Mikolov, K. Chen, and Corrado, 2013)

Input Data	Autoencoder models are 'flexible'. They can be applied to any kind of data where learning dense representation is useful	NWE models can only be used on textual data in NLP where learning the context and the connections between words and sentences is important
Use Cases	- Image Search - Text Search - Topic Modeling - Dimensionality Reduction - Denoising	- Text Similarities - Text Representation

Table 4.1 Autoencoders VS Word Embeddigs

4.5.5 Skip-thought Vectors

Skip-thought autoencoder vectors is an approach that was introduced by (Kiros et al., 2015). The authors describe a methodology for unsupervised learning of a both generic and distributed sentence encoder. That is, an encoder maps words to a sentence vector and a decoder is used to generate the surrounding sentences. In this setting, an encoder is used to map, for example, an English sentence into a vector. The decoder then conditions on this vector to generate a translation for the source English sentence. Sentences that share semantic and syntactic properties are mapped to similar vector representations, see **Figure 6.7** for a typical skip-thought model architecture. Consequently, the authors introduce a simple and rigid vocabulary expansion method to encode words that were not seen as part of training, allowing the skip-thought model the ability to process any unknown words. **Skip-thought** models are inspired by the skip-gram structure used in Word2Vec (Mikolov, K. Chen, and Corrado, 2013), however, the authors propose an objective function that

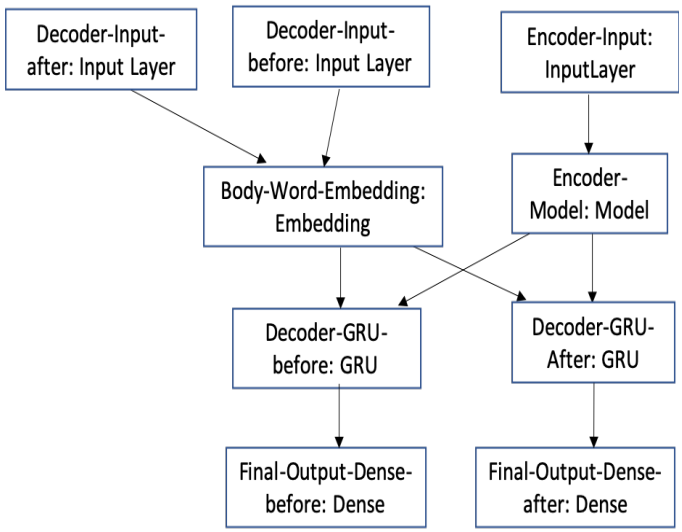

Figure 4.5 Skip-thought Model Architecture

abstracts this idea to sentence level instead of individual words. (see **Figure 6.7**).

Encoder. Let $w_1^i, ..., w_N^i$ be the words in a sentence s_i where N is the number of words in the sentence. At each time step, the encoder produces a hidden state h_t^i which can be interpreted as their presentation of the sequence $w_1^i, ..., w_t^i$. The hidden state h_N^i thus represents the full sentence.

Decoder. The decoder is a neural language model which conditions on the encoder output h_i. The computation is similar to that of the encoder except introducing matrices C_z**,** C_r and C that are used to bias the update gate, reset gate and hidden state computation by the sentence vector. One decoder is used for the next sentence, s_{i+1} while a second decoder is used for the previous sentence s_{i-1}. Separate parameters are used for each decoder, except for the vocabulary matrix **V**, which is the weight matrix connecting the decoder's hidden state for computing a distribution over-words.

4.6 Advantages of Combining CBR with Big Data and Deep Learning - Related Work

As explained in the former sections, deep learning algorithms are effective when dealing with learning from large amounts of structured or unstructured data. Big data represent a large spectrum of problems and techniques used for application domains that collect and maintain large volumes of raw data for domain-specific data analysis. Within the CBR paradigm, deep learning models can benefit from the available amounts of data, but the integration between CBR, big data and deep learning faces challenges that propagated from each research field (X. Chen and Lin, 2014). The age of big data poses novel ways of thinking to address technical challenges. While deep learning can be applied to learn from large volumes of labeled data, it can also be attractive for learning from large amounts of unlabeled/unsupervised data (Yoshua Bengio, 2013; Lecun, 2001; Yoshua Bengio, A. C. Courville, and Pascal Vincent, 2012), making it attractive for extracting meaningful representations and patterns from big data.

Substantial literature ((X. Chen and Lin, 2014; Yoshua Bengio, 2013; Lecun, 2001; Y. Bengio, A. Courville, and P. Vincent, 2013; M. Chen, Mao, and Liu, 2014; Labrinidis and Jagadish, 2012a; Chaudhuri, Dayal, and Narasayya, 2011)) expresses in detail the obstacles in the development of big data and deep learning applications. The key challenges are listed as follows (see Table **??**):

1. High volumes of data bring great challenges in all applications. Big data often possess numerous examples (inputs), large varieties of class types (outputs), and very high dimensionality (attributes).

2. The processing power required to train deep learning models, and the time consumed to get models ready to be used (e.g., DistBelief (Dean et al., 2012) can learn with very large models (more than one billion parameters), its training requires 16,000 CPU

cores, which are not commonly available to most researchers). Furthermore, GPUs are being utilized to implement a parallel scheme model: each GPU is only used for a different part of the model optimization with the same input example but also GPUs are not cheap or available to all researchers.

3. Data quality: Big data is often incomplete, noisy and unlabeled because it comes from different sources with different formats which make it complicated to capture the important data. Advanced deep learning methods are required to deal with noisy data and to be able to tolerate data inconsistencies.

4. Data Representation: Many datasets have certain levels of heterogeneity in type, structure, semantics, organization, granularity, and accessibility. Traditional Relational Database Management Systems (RDBMS) cannot handle the huge volume and heterogeneity of big data, which leads to the distributed File Systems and NoSQL databases as alternatives to the classical data storage systems. The choice of data representation technique varies based on the purpose of the whole solution. NoSQL databases have different types (graph, column based, key value, document, etc...) based on how we need to represent and query the data.

5. Data life cycle management: same like any normal data-based solution, we should define the data retention rate. The difference that in big data applications we are confronted with a lot of pressing challenges, one of which is that the current storage system cannot support extensive volumes of data. Therefore, a data importance principle related to the analytical value should be developed to decide which data shall be stored and which data shall be discarded.

6. Emerging challenges for big data learning also arose from high velocity: data are generated at extremely high speed and need to be processed in a timely manner. One solution for learning from such high velocity data is online learning approaches. Online

learning learns one instance at a time and the true label of each instance will soon be available, which can be used for refining the model (Bottou, 1999) (Blum and Burch, 2000) (Cesa-Bianchi et al., 1996) (Freund and Schapire, 2000) (Littlestone, Warmuth, and Long, 1995) (Shalev-Shwartz, 2012). This sequential learning strategy particularly works for big data, as current machines cannot hold the entire dataset in memory.

7. Data distribution is changing over time: Non-stationary data are normally separated into chunks with data from a small time interval. The assumption is that data close in time are piece-wise stationary and may be characterized by a significant degree of correlation and, therefore, follow the same distribution (Chien and Hsieh, 2013) (Sugiyama and Kawanabe, 2011) (Elwell and Polikar, 2009) (Elwell and Polikar, 2011) (Alippi and Roveri, 2008) (Alippi and Roveri, 2009) (Rutkowski, 2004) (Oliveira, 2007).

CBR and Deep Learning

Most of the CBR with deep learning literature have been produced during 2017 and after. In this section, I give the most recent research papers that used deep learning architectures within the CBR paradigm.

N/A In (O. Li et al., 2017), authors have combined the strength of deep learning and the interpretability of CBR to make an interpretable deep neural network model. The prototypes can provide useful insight into the inner workings of the network, the relationship between classes, and the important aspects of the latent space, as demonstrated here. Although their model does not provide a full solution to problems with accountability and transparency of black box decisions, it does allow to partially trace the path of classification for a new observation.

SelfBACK In (Sani et al., 2017), authors have presented a novel nearest neighbor sampling approach for personalized HAR that selects examples from a subject-independent training set that are most similar to a small number of user provided examples. The model is personalized to the user, and accuracy is improved. Evaluation showed that their approach outperforms a general model by up to 5% of F1 score. Another advantage of their approach is that it avoids the practical limitation of subject-dependent training by reducing the data collection burden on the user.

4.6.1 Potential Modern CBR Data in-Motion Apps

1. Real-Time Cyber-security: protects systems with superior threat detection.

2. Smart Manufacturing: dramatically improves yields by managing more variables in greater detail.

3. Support Ticketing Systems: solve and route voiced or texted customer tickets inside the organization.

4. Future Farming: optimizing soil, seeds and equipment to measured conditions on each square foot.

5. Automatic Recommendation Engines: match products to preferences in milliseconds.

4.7 Summary

The major philosophy behind the **DeepKAF** system was to develop modern CBR applications that can compete with other AI approaches in the industrial domain. This chapter emphasized the intersection area between CBR, big data, and deep learning. Based on the work experience, CBR has a strong edge over a lot of other AI approaches, however, the problem was to develop CBR applications that are scalable and that can handle large amounts of data. **Chapter 4** defined what big data is and when to call data "Big Data" along with various deep learning architectures being used in **DeepKAF**. **Chapter 4** has justified why **DeepKAF** is combining CBR, big data, and deep learning. In this chapter, I have also covered the literature that relates CBR with big data or CBR with deep learning and summarized the contribution for each developed approach. **Chapter 4** is essential in understanding **chapters 5, 6, and 7**, because it justifies many decisions that have been made during the **DeepKAF** research and development.

Chapter Five

DeepKAF: Deep Knowledge Acquisition Framework

Finding an appropriate case representation is important, and finding an appropriate organization of the case base. If, in addition, the available knowledge sources consist (mainly) of textual data, knowledge representation is even more challenging (Aamodt and E. Plaza, 1994). To do this, the choice of an ideal representation is guided by the domain characteristics and the complexity of its cases. Recently, the increased availability of deep learning techniques and other forms of vectorized representations has provided a new source for case insights. Richer text features can be extracted and used for each case if required. **DeepKAF** is built based on continuous research in the area of deep learning and CBR. **DeepKAF** generates richer case representations by automatically acquiring domain knowledge from unstructured sentences using different deep neural networks architecture.

5.1 About this Chapter

This chapter presents the **DeepKAF** architecture that is based on the conclusions from the previous chapter. A more detailed state of the art is provided, closely related to what **DeepKAF** is achieving, to show the main difference between **DeepKAF** and the other

architectures and frameworks presented in the literature. This chapter describes the generic architecture of **DeepKAF**. The main challenge when it comes to the **DeepKAF** implementation is what models to be used, and for what purposes, and how to incorporate all these things within the CBR system without hiding the explainability of the results. Above all the deep learning and big data frameworks used is their orchestration, which is essential for a successful use of **DeepKAF**. It will be described how **DeepKAF** is obtaining its representation vectors from stemmed words, improving these vectors iteratively, and suggesting high quality outputs being relevant for domain experts based on either explicit queries or their experiences.

5.2 Related Work and State of the Art

The literature covers several methods that can help to construct similarity measures for textual CBR, particularly when textual data include abbreviations or domain specific language. The related work in this section has been divided into two main parts: 1. Building similarity measures 2. Using Siamese neural networks (SNNs) to find similarities.

5.2.1 Building Similarity Measures Challenges

Finding relations among different sentences is a key factor in building text similarity measures. This process can encounter different challenges varying from: the complexity of decoding the domain knowledge, the domain-specific abbreviations, incorrect sentences and multi-lingual text to the unavailability of the domain experts (Stram, Reuss, and K. Althoff, 2017; Reuss, Witzke, and K. Althoff, 2017; Reuss, Stram, et al., 2016). The authors tried several approaches to build similarity measures in a domain where there are too many abbreviations and textual descriptions of issues, which were written by technicians and engineers. Finding relations among cases and understanding the textual data provided using the traditional NLP frameworks can be a tedious process since NLP frameworks have limited

functionality in parsing the text accurately due to incorrect sentences or abbreviations.

5.2.2 Cases Similarity using Siamese Neural Networks

The Literature shows a successful approach in using trained Siamese neural networks to build CBR similarity measures. Martin et al.(2017) have used them to identify appropriate similarity metrics for the **selfBACK** dataset (Martin et al., 2017). In this work, convolutional Siamese network implementations were used to build similarities between different cases in a dataset that contained time series data. Results were base-lined against a typical convolutional neural network, and the Siamese architecture outperformed the CNN implementation. The difference between the implementation mentioned in (Martin et al., 2017) and the work presented in this paper is in how Siamese networks are being used and trained. The self-BACK dataset does contain numerical time series data and not textual data. Comparing to **DeepKAF**, in (Martin et al., 2017) the authors have trained the Siamese architecture over the entire case and not on specific attributes. For **DeepKAF**, an explicit model for every attribute has been trained to generate the local similarities and then combine them into a global similarity.

The work of automating knowledge extraction using neural networks can be compared to the work presented in (Sizov, Öztürk, and Štyrák, 2014). In this work, the authors represented the knowledge extracted from text in a graph-based approach which dubbed as Text Reasoning Graph (TRG). Relevant work has been seen in a graph-based representation with expressive power to represent the chain of reasoning underlying the analysis, as well as facilitate the adaptation of a past analysis to a new problem. The authors have used manually constructed lexico-syntactic patterns developed by Khoo (S. G. Khoo, 1996) to extract the relations between text elements. In **DeepKAF**, a Siamese neural network has been used to automatically detect these relations between text elements instead of doing it manually. **Chapter 6** shows an example of how I trained and used the Siamese network to

build similarities along with the evaluation results.

One major advantage of a **DeepKAF** compared to other approaches, is that it uses deep neural networks to build relationships and extract features from text, which contains domain-specific knowledge, or mixed languages and still use CBR to provide justification for the reached conclusions. Although the literature shows some early successes in the field, there is still further potential to decide accurately whether to use Siamese networks either on the attribute level, or on an entire case level basis, or on both.

5.3 DeepKAF Processes

DeepKAF is defining the procedure of implementing a CBR system and injecting deep learning models while still being able to provide an adequate level of explainability (see Figure 5.1).

Automatic knowledge extraction from natural language data can be a challenge since language expressions within business context are usually unstructured, permuted and convoluted. Text can be extracted from various, unrelated sources and can be complex enough to make traditional natural language processing techniques inadequate while processing it. This can be mainly due to grammatically incorrect text, the presence of different languages, or due to implied concepts that lead to domain ambiguity. Examples of such lexical content ambiguity can be differences in intended meaning for the same word, as a result of word synonymity (different terms have similar meaning, or context can vary). Natural text can also contain different languages, which increases the scarcity of satisfactory results while processing it. Unsupervised text processing techniques are increasingly used nowadays to provide users with the right information at the right time. They do this by processing and profiling large quantities of textual information over time and use it to filter out items for presentation based on user queries and preferences that are collected at almost real time. In a customer support setting, unsupervised text pre-processing and processing endeavor to

learn from the past cases in order to identify solutions to present problems that customers wish to solve in an efficient and usually urgent manner. Getting this right can mean improved problem-solving ability for a service provider and increased satisfaction levels for a service user and customer. More satisfied users improve the likelihood of choosing the same provider again, increase the brand loyalty, and result in a win-win case for customer and service provider alike.

Figure 5.1 depicts the CBR processes in **DeepKAF**. Each process has a number with a letter as a prefix. This prefix represents the respective stage: **K** denotes knowledge acquisition stage, **S** denotes building similarities stage, and **R** denotes the retrieval stage. Each process is going to be described in details and the task number will be used for reference purposes. The **DeepKAF** architecture is flexible enough that we can use the processes described below processes to work on building global similarities, local similarities or both. In other words, the deep learning models can be built on an attribute level or a case level based on the domain and task in hand. The recommended approach is to use it on the attribute level. Train models on the attribute level and having a specific models pipeline ensures higher accuracy but a more complex building architecture.

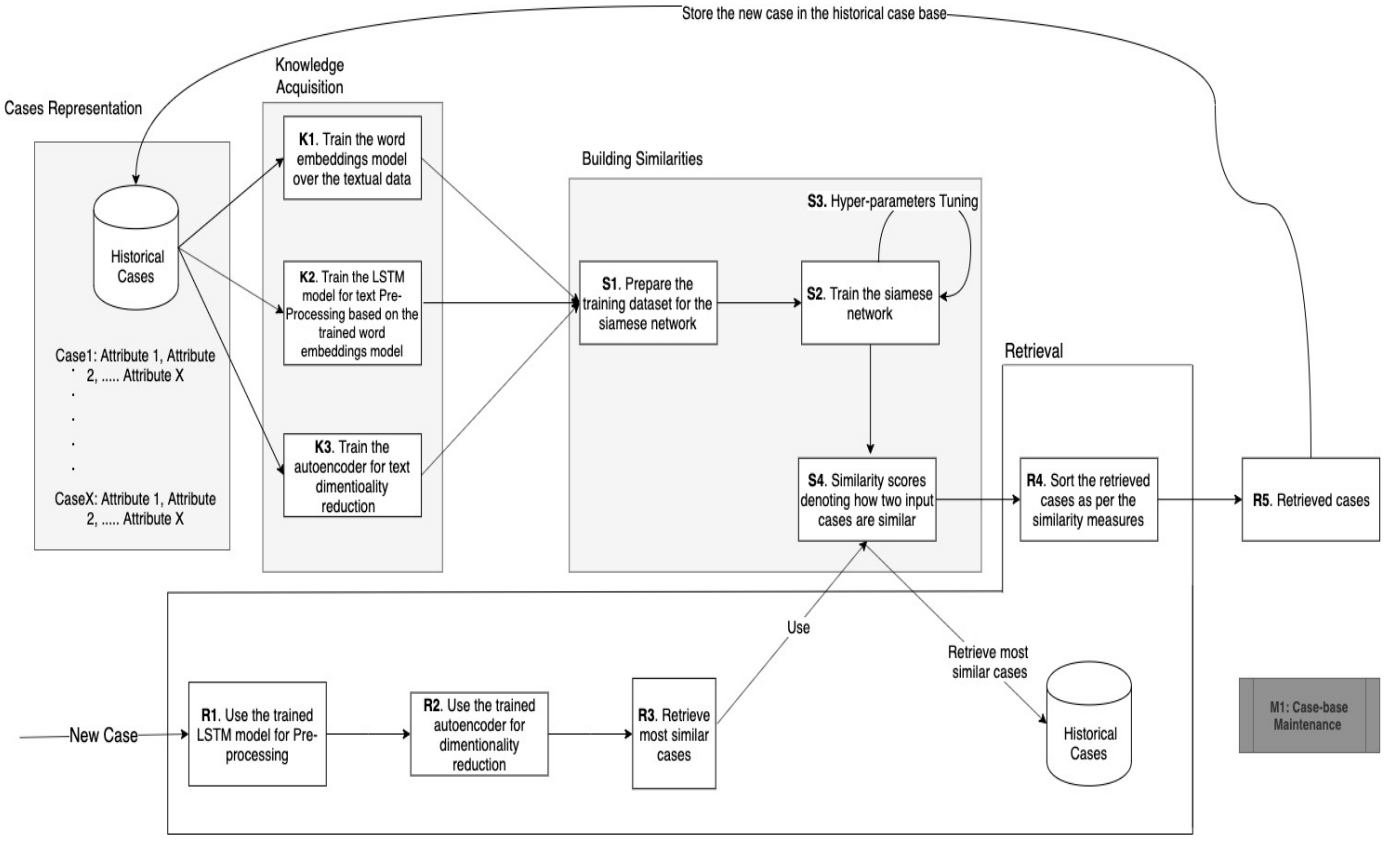

Figure 5.1 DeepKAF Processes

K1: **Objective:** Build the word embeddings model from all available textual data. This word embeddings model will be used with all the other deep learning models that have been used in **DeepKAF**. The word embeddings model is the first step in acquiring the domain knowledge.

Input: The input layer for the word embeddings neural network takes a larger corpus of text to build a vector space that has multiple dimensions. Every unique word in the text corpus is assigned a corresponding vector in the space.

Output: A vector space that contains the vector per word. The context of each word vector influences the distance between the vectors. If some words are coming together very often in sentences, they will have closer vectors.

K2: **Objective:** Do the text pre-processing tasks for the input text. Text can not be processed directly, or it will give bad accuracy. During the pre-processing process, the important parts from the text has to be highlighted and ignore the rest. The aim for **DeepKAF** is to work with real-time applications, and normally the input text will have some noise that needs to be eliminated before processing the text (the experiments in Chapter 6 show this process in details).

Input: The input for this process is the trained word embeddings we built in **K1** and a training dataset that shows the important words or text that the model should look for and ignore the rest. Based on the experiments, a LSTM model showed the best results in being able to identify the most important parts from the text.

Output: A trained model that is able to find the important parts in the text and feed it to the next process.

K3: **Objective:** Generate a low-dimensional representation to the text while keeping the most essential properties. The main objective of this process is to find the optimal representation for the input text without knowing the exact meaning of it. This is important in complicated domains, where the knowledge engineer has to understand the domain jargon in order to decide how to represent the text. However, with process **K3**, the aim is to build a model that is able to find this representation automatically in an unsupervised learning approach. Based on our experiments, auto-encoder neural networks showed exceptional results in finding the best representation of the text.

Input: The important parts from the text based on the trained model in **K2**.

Output: A lower-dimensional text based on the input.

S1: **Objective:** S1 is the first process in building similarities measures. The objective of this process is to start preparing the training dataset that is going to be used to train the Siamese network.

Input: The input for this process will be the low-dimensional text coming from **K3**. The training dataset should contain two texts from the historical cases and a degree of similarity. The target variable for this Siamese network is to be able to predict how similar two texts are.

Output: A dataset with three columns, two texts and one for degree of similarity.

S2: **Objective:** The objective of this process is to train the Siamese network and make it ready to predict text similarities. Throughout my research, I have tried several techniques to build the similarity measures, but the Siamese network model surpassed all the other techniques.

Input: The training dataset that was prepared in **S1** along with the word embeddings model that was built in **K1**.

Output: A trained Siamese network model that is able to identify the domain text and predict the similarity degree between two texts.

S3: **Objective:** The objective of this task is to find the optimal or near optimal hyper-parameters for the Siamese network that would allow the model to achieve good generalization/out-of-sample performance. There are several techniques to find the optimal hyper-parameters, which can be applied based on the type of data and data attributes (Claesen and De Moor, 2015). In **Chapter 6**, I provide the hyper-parameters I used for each implemented deep neural network model.

Input: The model accuracy.

Output: Optimized hyper-parameters.

S4: **Objective:** Store the trained models, the autoecnoder, LSTM, word embeddings and Siamese network in a usable format like PMML (Predictive Model Markup Language). These models will be used later during the retrieval process.

Input: The trained models.

Output: The trained models will be used during the retrieval process to do the text pre-processing, dimensionality reduction, and then similarity prediction.

R1: **Objective:** In the retrieval phase, the trained models are used that have been trained during the knowledge acquisition and building similarities. The objective of this process is to receive the queries and do the pre-processing.

Input: The new query.

Output: The textual data in the new text after performing the pre-processing.

R2: **Objective:** The objective of this process is to find a low-dimensional representation for the incoming query before pushing it to **R3**

Input: The new case after pre-processing.

Output: A low-dimensional representation of the query.

R3: **Objective:** The objective of this process is to use the trained Siamese network model to retrieve the most similar cases.

Input: The new case after pre-processing and dimensionality reduction.

R3 **Output:** A list of retrieved cases that have the highest similarity degrees. We can decide to retrieve the top ten similar cases or any other number based on the needs.

R4: **Objective:** The objective of this process is to sort out the retrieved results based on the similarity degree that was calculated by the Siamese network.

Input: The retrieved cases with similarity degrees.

Output: A sorted cases list.

R5: **Objective:** The objective of this process is to show the retrieved cases for the end user and get feedback. Based on the feedback, a new case is going to be added to the case base

Input: The sorted retrieved cases.

Output: A new case to be added to the case base

M1: **Objective:** The objective of this process is to keep the case-based reasoning system efficient and accurate. Therefore, case base maintenance and models maintenance should be carried away on a regular basis based on how many new cases are being added to the case base. Models have to be re-trained based on the techniques described in **Chapter 4** to ensure high accuracy.

Input: The historical case base

Output: Retrained models

5.4 DeepKAF Technical Architecture

The **DeepKAF** framework has been designed and implemented to leverage case-based reasoning as an explainable AI approach by using various deep learning models to overcome and mitigate the limitations of CBR implementations. According to their survey about what's hot in the CBR domain (A. Goel and Belen Diaz-Agudo, 2017), case acquisition from raw data, including text and diagrams is on the top of the list along with implementing CBR approaches that can deal with high velocity / veracity / data volumes. This section ex-

plains the technical architecture to build a **DeepKAF**-based system. As described earlier, **DeepKAF** involves using approaches from three domains, CBR, big data, and deep learning. Therefore, Figure 5.2) depicts the tools and frameworks I have used to build several **DeepKAF**-based systems. These systems are described in details in **Chapter 6**.

Within specific industrial domains where tacit knowledge is present, deep learning approaches alone cannot cover and satisfy the expert needs completely. **DeepKAF** is showing how CBR can be extended in dealing with up-to-date industrial application challenges, like decoding large amounts of unstructured text, multi-language content, and mixed data types, for example numeric, categorical, etc..

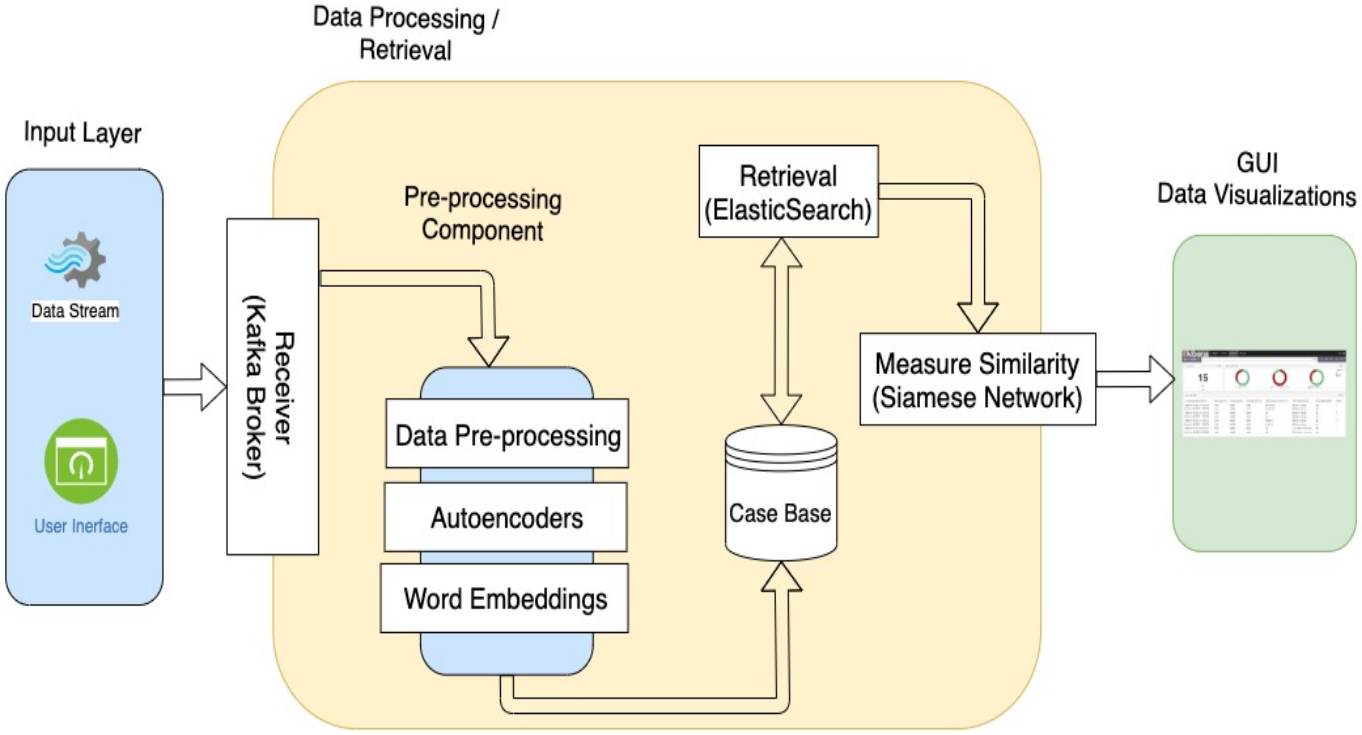

Figure 5.2 DeepKAF Technical Architecture

DeepKAF is mainly improving two key elements of CBR components, namely similarity measurement and retrieval.

5.4.1 Similarity Measures Component

Within any CBR system, building similarity measures is considered one of the most tedious and complicated tasks((Stram, Reuss, and K. Althoff, 2017; Reuss, Witzke, and K. Althoff, 2017; Reuss, Stram, et al., 2016)). Similarity measures are highly domain-dependent and used to describe how cases are related to each other. Therefore, an intensive involvement from the domain expert is required to be able to decode the domain knowledge and being able to describe how cases and attributes can relate to one another. In **DeepKAF**, the main goal is to semi-automate the building of similarity measures in CBR systems using textual data sources.

The similarity assessment component in **DeepKAF** is divided into two stages: 1. Building and training the deep learning models stage; 2. Application stage.

In the building and training stage, several models are used to decode the domain knowledge in a combination of unsupervised (word embeddings), semi-supervised (autoencoders) and supervised (Siamese network) training approaches.

As showed earlier in Figure 5.1, the process of building similarity measures within **DeepKAF** consists of the below steps:

Step 1: Use denoising and dimensionality reduction autoencoders on the input data.

Step 2: Build word embeddings for the text.

Step 3: Split the dataset into training/test data.

Step 4: Train the Siamese neural networks.

Step 5: Check the accuracy with the test data.

Step 6: If the accuracy is not high enough, do hyper-parameters tuning, check the data quality (if possible), and re-run the entire process.

In the application stage, the trained models which were built in stage 1 are being used

to measure similarities between two cases in the retrieval process.

In the next section, the retrieval component is described in detail.

5.4.2 Retrieval Component

In the retrieval component, the models that have been built and trained are going to be used in order to clean the input text first and then retrieve the most similar cases via the trained Siamese network. The Siamese network gives a range between 0 (no similarity) and 1 (identical) to how similar the cases are. Lastly, the retrieved results will be sorted based on the global and local similarity measures.

Figure 5.1 showed the retrieval process within the CBR paradigm combined with the trained deep learning models. In CBR, the typical retrieval strategy uses similarity knowledge and is therefore called similarity-based retrieval (SBR) (Kang, Krishnaswamy, and Zaslavsky, 2011). In **DeepKAF**, the traditional similarity matrix is replaced by the trained models that can give degrees of similarities between two distinctive textual cases. The notion of a metric or distance in similarity measures, *d(c1, c2)*, between cases *c1* and *c2* has been used in many contexts during a CBR system implementation. There are many techniques to calculate this distance between two cases, and the outcomes is a similarity matrix that shows how each attribute relates to each other and defines the relation between the cases (Finnie and Sun, 2002). Normally, building the similarity measures matrix is tedious and involves so much work, therefore, **DeepKAF** uses the trained siamese neural network to give the similarity measures on the attribute level instead of building the traditional similarity matrix.

The retrieval process consists of the following steps:

Step 1: Use denoising and dimensionality reduction autoencoders on the input data (exactly as in the building similarity measures phase).

Step 2: Use the trained Siamese network to retrieve the most similar cases.

Step 3: Calculate the local similarity (attribute-based), and the global similarity (case-based) based on the similarity degrees that are given by the siamese network.

Step 4: Rank the retrieved cases according to the domain preferences.

Step 5: Get the end user feedback on the retrieved results.

Step 6: Retrain the models according to the methodologies described in **"Section 5.4.3"**

Step 7: Store the new case in the case base along with its associated potential solutions.

5.4.3 Feedback and Models Retraining

Means to provide feedback and "back-input" are essential processes for any industrial CBR system. Within a traditional CBR cycle, constant user feedback is used for this purpose. Typically, a new case and its solution are added to the case base as part of the "lazy" learning process that CBR systems use to improve. For **DeepKAF** the feedback process leads to ongoing deep learning model retraining that extends the used vector space. This is being used respectively to measure similarities more accurately and increase the coverage of word embeddings model by increasing its vocabulary.

In this section, different re-training approaches that can be used within **DeepKAF** will be described.

Word Embedding Model Retraining

DeepKAF uses a word embedding constructed during its training phase, which is provided from the available case content, vocabulary, and metadata. Any new case provides acts as feedback input, having as a result the retraining of the existing word embeddings model on new terms, available words and sentences. In other words, continuous training of the word embeddings model with every new case that is added to the case-base is essential to maintain

high accuracy in the retrieval process. However, retraining the word embeddings model with every new case is a tedious and time-consuming process. Therefore, in my opinion, the best approach is usually to expand the initial word corpus with as much domain data as possible, so that the word embeddings model has "enough" examples of every word important to that domain. Several word embeddings models have been used throughout **DeepKAF** (Amin et al., 2018a), with no specific model prevailing and thus recommended over another one. An observation from the authors is that embeddings are affected from the nature of the text. For example, if the text is grammatically correct, has abbreviations, or does not have enough text to sufficiently train the model enough. The training process of any word embeddings model should be based on high coverage of available words and sentences and hence having a strong embeddings model with all necessary links between words and sentences. There are two main retraining approaches on this, namely incremental learning and transfer learning.

Incremental training techniques differ based on the model to be retrained. **DeepKAF** has tested several word embedding models among which word2vec (Mikolov, K. Chen, and Corrado, 2013) and its descendants (e.g., sentence2vec, sentence2sentence etc...) seemed most applicable. Word2vec models do not offer an option for direct, incremental training. They have to be re-implemented extensively and heavily modified from source code to be able to build a model that can be retrained.

Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned (Verwimp and Bellegarda, 2019). In **DeepKAF**, transfer learning has been used in retraining a word2vec model with new words derived from the new cases. Word embeddings have been a key component in the success of the entire approach because they are used by a full range of the neural network topologies to measure similarities like LSTMs and MaLSTM, which will be described in the following sections. **DeepKAF** delegates the training to a new model by propagating back to word vectors (word vector inputs can be fixed, in which case the issue below does not apply). If the LSTM training corpus is sufficiently large such as that the vocabulary in the training

phase retrains all the word2vec vectors, this is regarded as a successful retraining phase. If the training corpus does not retrain all vectors, and we get a sentence in the testing phase that has a word that was not seen before, there is a risk of lowering the model accuracy, since untrained vectors will be mixed along with the ones that were trained. This specific case requires extra attention. In summary, if word vectors are chosen for transfer learning, and the word vectors are retrained in the new space, the framework needs to ensure the training moves all the words in the original vector space - that is the new corpus should have all the words in the original word2vec vector output. To summarize the retraining process, retraining is crucial in **DeepKAF** because there are several neural networks depending on each other, and the common model is the word embeddings model that being used in all the neural networks. Therefore, retraining of one word embeddings model without ensuring that other models are also trained on the same word corpus and having the same vector space can negatively affect the entire **DeepKAF** similarity measures component.

Siamese Networks Retraining

Unlike other neural networks that learn to predict over different classes, Siamese neural networks can learn how similar or dissimilar two objects are. Hence, with every new case, the SNN model needs to be retrained on the whole dataset, which is an exhaustive task in terms of time and processing power. Based on several experiments to establish the most appropriate similarity degree, it was observed that the top ten retrieved results (see **Chapter 6** on **DeepKAF** Evaluation) can be used in data augmentation techniques to reproduce a small dataset that consists of similar objects and use transfer learning for the retraining process.

5.5 How DeepKAF is Generic and Explainable?

The research on **DeepKAF** provides arguments that the complications of building similarity measures, decoding unstructured domain knowledge, and the tricky adaptation methods are road-blockers for CBR to penetrate the industrial market on larger scale. Henceforth, **DeepKAF** is providing an approach that by using it, CBR systems are able to cope with whatever any AI system is confronted with. The essential objective of developing **DeepKAF** is to build an AI solution that is generic and explainable.

This section illustrates how the **DeepKAF** approach is using generic DL models that can be re-used and applied with other use-cases and still being able to provide explanations of its results.

5.5.1 Explainability

Explanation is an important sub-field of intelligent systems. An approximate definition of explanation in systems is the collection of logical arguments that could convince a user to adopt proposed assumptions, recommendations and even decisions up to a certain extent. In artificial intelligence, and especially in the area of case-based reasoning, systems are able to extract explanation knowledge from the available past information. This can be presented afterwards to their human stakeholders in order to provide reasoning and justification for any system recommendations and/or decisions. The explanation provision increases the confidence in an artificial intelligent system and has as a target the development of trust between itself and its users. Everybody's personal experience shows that building trust among humans is a difficult task. Therefore, if this task is applied to an artificial environment, it can be even more difficult.

In **DeepKAF** the deep learning models are being used in a way that makes the retrieved cases traceable. The black-box ML techniques are being used in classification, clustering, or prediction problems, and it is not easy to provide reasons why a specific value was predicted

or chosen (Weerts, Ipenburg, and Pechenizkiy, 2019). On the other hand, the problem to be solved in the CBR retrieval phase is a search problem where there are already historical cases stored, and the most similar cases should be retrieved. That means, whenever cases are retrieved, a similarity degree will be provided of how similar the retrieved cases are to the new case. The Siamese network that is being used in the retrieval layer within **DeepKAF** is trained on finding similar cases from the historical case base and provides a local similarity degree per attribute, and in the end it is the human decision to decide which is the best solution to be applied.

This section will give a detailed view on how explainability works within **DeepKAF** and what **DeepKAF** is adding to this point, however, in **Chapter 6**, there is an experiment on a ticket management system that is receiving new tickets by emails and the CBR system extracts the important parts from the text and finds the most similar cases from the case base.

Figure 5.3 depicts how a ticket management system can be implemented with a CBR based on **DeepKAF**, and a machine-learning only approach.

As an AI Engineer, to build such a system, there are many possible approaches and methodologies to apply. Machine and deep learning approaches are prominent and can successfully be applied. A CBR approach can shine in such use cases as well because of the ability to provide explainability for each solution, which is essential in some domains. To give the same level of explainability for the machine learning models, different approaches have to be used, and more complexity will be added to the system overall.

In the CBR only approach (without **DeepKAF**), we will face problems in extracting important attributes from the input text that will be used in the retrieval phase. Traditional CBR will go through the traditional similarity measures building process that has been described before. Thus, building a CBR system would not be an easy task.

In the machine learning only approach, the system might provide good results eventually, but it is not able to provide any level of explainability to the domain experts. The system

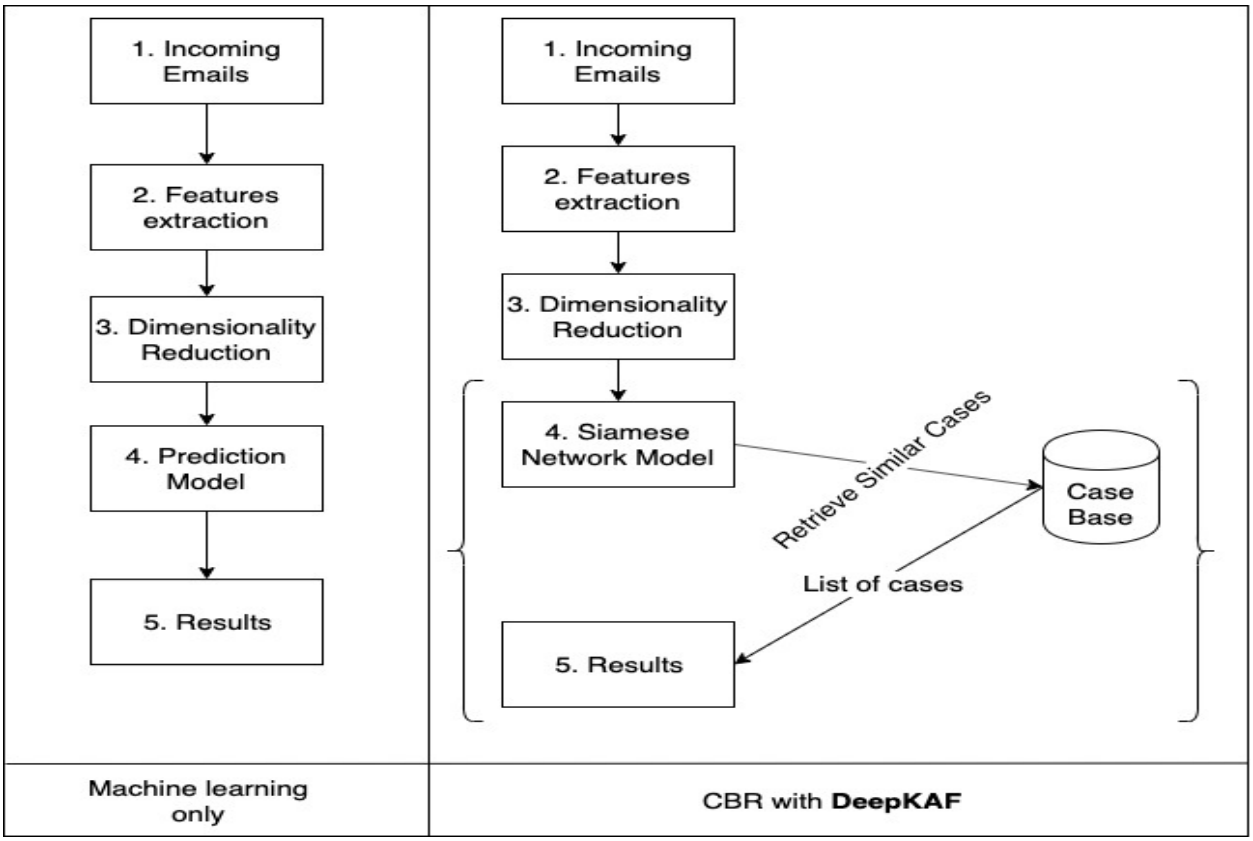

Figure 5.3 A comparison between CBR-DeepKAF and Machine Learning Approaches

will be predicting the solution instead of finding it from the historical cases. The Siamese network would play no role in this approach because there is no comparison process to be included. Machine learning approaches exploit the historical data in the training phase and then ignores them, while CBR approaches exploits every single case stored in the case base.

In the CBR based on **DeepKAF** approach, the trained neural networks extract the important attributes from the incoming email and the Siamese network uses these attributes to retrieve the most similar cases from the case base, and then calculate the similarity degrees as described earlier in the architecture. The main difference here is that in this approach, the CBR system is finding the solution from the historical data and providing a solution based on the findings. If a domain expert wants to validate the retrieved results, a list of relevant cases is provided to prove how the current solution is derived by former solutions stored in the case base. **DeepKAF** as an approach can use Siamese networks to be trained on the attribute level and this supports heterogeneous cases where cases can consist of an attribute that is an image and another attribute is a text. **DeepKAF** uses different deep learning models, which definitely hides some explainability. However, **DeepKAF** uses these models not for the task but for very specific sub-tasks like computing similarity for one attribute. As a consequence, DeepKAF's cases maintain a certain level of transparency and, therefore, the retrieved cases are still providing a sufficient level of explainability and justification to the domain experts.

5.5.2 Genericity

Although the provided models within **DeepKAF** are not generic themselves, however, the generic "part" is that anyone can follow the architecture and tailor each DL model to given domain-specific needs. For instance, autoencoders as an artificial neural network architecture are generic for data dimensionality reduction and denoising, but the skip-thought autoencoder that has been used in the implementations is popular in the NLP domain. The Siamese

networks are very efficient in finding similarity between two objects, but the MaLSTM that has been used is mainly applied for textual data. Word embeddings are more generic because they can be re-used easily if you are dealing with standard textual data from any languages, and there are many pre-trained word embeddings models (Yamada et al., 2020).

To conclude, having a generic approach is never an easy task. If we look at each domain from the outside, we will see each application is very domain-dependent and each domain has its own unique problems and challenges, which is definitely correct to some extent. However, throughout the past years with combined business, technical, and academic experiences, there were specific problems and challenges that always come with specific data types. These problems are not technical challenges. Those problems are related to how to understand this domain data and how to decode such an amount of domain knowledge in a manual way? How to handle data that is moving and being updated fast? How to build a solution that is able to find similarities without a real understanding of the data content; that data could be an image, a text, or a sound?.

DeepKAF is grounded based on the leading questions that brought all that NN models to be used together within the **CBR** paradigm. The main competitive advantage of CBR is, besides its cognitive plausibility, its explainability capability that comes naturally with any CBR application **(see Chapter 3)**. But on the other hand, there are so many challenges that face any CBR system implementation in terms of building similarity measures, and that's what **DeepKAF** aimed to improve.

5.6 DeepKAF Extended View

This section presents an extended view differentiating **DeepKAF** from other techniques in Textual-CBR and explains the competitive advantages of using deep learning models within the CBR paradigm. **Section 1** compares DeepKAF as an approach with the traditional Textual CBR approaches. Section 2 shows how DeepKAF is able to handle complex NLP

tasks like the ones mentioned in the former chapters. **Sections 3 and 4** explain how Siamese neural networks and autoencoders architectures have added huge value to **DeepKAF**. Given that the genericity of **DeepKAF** is one of the main advantages, **Section 7** is highlighting how **DeepKAF** is generic and can be used with various CBR use-cases.

5.6.1 DeepKAF VS Traditional Textual CBR

Text is a unique way of expressing knowledge. Text can be denoted as a collection of words in a well-known language, which transmits meaning (i.e. ideas) when interpreted in agglomeration (Richter and Weber, 2013). Those collections are not organized in easy to interpret performances, for example, attribute-value pairs. Textual CBR is used when sources of knowledge for CBR knowledge containers are in textual format.

In traditional TCBR systems, the textual formulations can be used for queries, problems, and solutions.The lack of explicit formal structures means that they cannot be automatically understood by the computer system. In (Orecchioni et al., 2007), the authors were building a CBR system to analyze air investigation incident reports. Different bag of words and **Wordnet** techniques were applied to decode the domain knowledge. The authors mentioned the challenges they faced with extracting the domain knowledge and used ML techniques to support understanding the incidents reports. It was challenging to obtain similar reports because investigators may typically want to obtain reports based on multiple factors such as weather conditions, aircraft type, geographic location, or cause of the incident. The number of incidents and sentences makes this a daunting task, to be extracted manually from sentences (Orecchioni et al., 2007). In (Brüninghaus and Ashley, 2001), the main motivation behind the research was that most TCBR approaches are limited to the degree that they are based on efficient, but weak information retrieval (IR) methods. These do not allow for reasoning about the similarities between cases, which is mandatory for many CBR tasks beyond text retrieval, including adaptation or argumentation. The authors introduced an IR approach

that is called **AutoSlog** (Riloff, 2000). They worked on manually building the classification tree to extract and build the features that they were going to use. Given the number of textual data that CBR systems have to handle, building such a tree in a manual process would be an excessive process. The results presented in the aforementioned research papers were promising and showed a significant improvement to the domain knowledge decoding. However, the two researches, and many others in the area of TCBR, were not confronted with mixed-languages, domain-specific abbreviation, or grammatically incorrect text. However, these three common characteristics are essential for nearly any potential industrial CBR system. **DeepKAF** has been built to deal specifically with those kinds of problems with text processing. Another relevant work has been seen in text reasoning graph, a graph-based representation with expressive power to represent the chain of reasoning underlying the analysis as well as facilitate the adaptation of a past analysis to a new problem. The authors have used manually constructed lexico-syntactic patterns developed by Khoo (S. G. Khoo, 1996) to extract the relations between texts. Khoo and the co-authors built manual lexico-syntactic patterns in order to be able to generate similarities. The major disadvantage of all the aforementioned techniques is that they involve so much manual work that would be very costly to provide in case there is more complicated textual data or mixed-language text. By contrast, **DeepKAF** and the unsupervised learning approach contribute to overcoming the main drawbacks of what is presented in the previous researches. In (Öztürk, Prasath, and Moen, 2010), the authors presented a brilliant approach based on Holographic Reduced Representations(HRR), which have been extensively used in image and signal processing (Plate, 1995). HRR uses a vector operation called circular convolution, which is a multiplicative operation that allows two or more different types of information about a feature. The difference between HRR and other approaches presented in the literature is that it is forming the syntactic relations between words and works in an unsupervised learning approach. Syntactic relations are important, and works perfectly fine when the text is from one language and grammatically correct. Word embeddings, in contrast, can capture relations between

words based on their sequence and that specific words appear after other words (semantic relations), regardless of the language or grammar. Word embedding is actually seen as one of the popular applications of unsupervised learning. They do not need annotated corpora. Embedding takes advantage of a lower-dimensional space while retaining semantic relations. Using word embeddings was the key to build relationships between words and sentences in text without the need to understand the text content. With other deep learning models, the domain knowledge decoding process was semi-automated with a high accuracy in the retrieval results and minimum effort from domain experts or knowledge engineers. Using word embeddings combined with autoencoders helped in building more precise and correct relationships between only the important words and sentences, which affected the end results of **DeepKAF**.

5.6.2 DeepKAF and Complex NLP Tasks

NLP copes with the construction of computational algorithms for the automatic analysis and representation of human language. The history of natural language processing (NLP) generally started in the 1950s, although work can be found from earlier periods. The history of natural language processing (NLP) generally began in the 1950s, though work from earlier periods can be found. In 1950 Alan Turing published an article entitled "Computing Machinery and Intelligence" (TURING, 1950) proposing what is now called the Turing test as an intelligence criterion. In this section, the NLP methods that **DeepKAF** is using are going to be explained and why those methods in specific were chosen and how they are giving **DeepKAF** an edge in the area of Textual CBR.

For a long time, most approaches used to study NLP problems applied shallow machine learning models and time-consuming, hand-crafted features. This leads to problems such as the curse of dimensionality because linguistic knowledge with sparse representations (high-dimensional characteristics) was present. Neural-based models, however, have achieved

superior results on various language-related tasks compared to traditional machine learning models such as SVM or logistic regression, with the recent popularity and success of word embeddings (low-dimensional, distributed representations).

Why is NLP difficult? The processing of natural language is considered a difficult problem in computer science. It is the existence of the human tongue that complicates NLP. It is not easy for computers to understand the rules which govern the passage of information using natural languages. Some of those rules may be high-level and abstract; for instance, when someone uses a personal remark to impart knowledge. Some of these laws, on the other hand, might be low-level; for example, using the character **"s"** to denote the plural of objects. Although humans can quickly learn a language, what makes NLP difficult for machines to implement is the complexity and imprecise characteristics of the natural languages. The primary methods used to complete natural language processing tasks are syntactic analysis and semantic analysis.

Syntax refers to word structure in a phrase in such a manner that it makes grammatical sense.

Here are some syntax techniques that can be used:

1. Lemmatization: For easy analysis, it involves reducing the different inflected forms of a word into one form.

2. Morphological segmentation: It includes the division of words into single units, called morphemes.

3. Word segmentation: It includes the division of a broad piece of continuous text into separate units.

4. Part-of-speech tagging: It includes defining the spoken element for every word, and different meanings for the same word based on its position in the sentence.

5. Parsing: For the given sentence, it requires performing grammatical analysis.

6. Stemming: It involves cutting the inflected words to their root form.

Semantics refers to the interpretation of a text. Semantic analysis is one of natural language processing's complicated aspects that has not yet been entirely overcome. Here are some techniques in semantic analysis:

1. Named entity recognition (NER): It includes deciding which sections of a text can be categorized and divided into preset classes. Examples of such groups include individual names and place names.

2. Word sense disambiguation: It involves giving a context-based word meaning.

3. Natural language generation: This includes using databases to extract and translate abstract thoughts into human language.

Both text semantics and syntax are important in order to build accurate similarity measures. Hence, it is essential to use the right techniques in any framework that intends to deal with such NLP difficulties. The strong points about **DeepKAF** are that **DeepKAF** is designed with the aforementioned NLP challenges in mind, combining neural networks for text processing as described before. With the complexity of NLP problems and the approaches that are available, the pre-processing layer in **DeepKAF** is using the combination of autoencoders and word embeddings (Amin et al., 2019) models to be able to capture the most important parts from the text and build word embeddings with the optimized text representation.

In the process of finding the right approach that is able to find a representation for text without the need of understanding the content, there were several frameworks like **TwitterNLP** (Owoputi et al., 2013) and **StanfordNLP** (Manning et al., 2014). Both frameworks have been applied successfully with all NLP problems. Both frameworks provided a

tokenizer, a part-of-speech tagger, hierarchical word clusters, and a dependency parser for tweets, along with annotated corpora and web-based annotation tools. However, they were not able to give good results with grammatically incorrect and mixed-language text as explained in Amin et al., 2018a. Therefore, finding another approach that is able to overcome these issues that traditional NLP frameworks have was necessary. The word embeddings approach was chosen based on the state-of-the-art discussions in the area of neural networks and NLP problems (Goldberg, 2017). Goldberg in his survey (Goldberg, 2017) discussed several neural network approaches that are able to solve different NLP problems. Word embeddings models in general outperformed the other approaches in representing relationships between different words in the text and later between sentences (e.g., sentence2vec) and documents (e.g., doc2vec) as described in (Le and Mikolov, 2014b).

Word embedding is a class of techniques in which individual words are interpreted as real-valued vectors in a predefined vector space. When input to a neural network includes semantic categorical features (e.g., features that take one of k's distinct terms, such as words from a closed vocabulary), it is normal to equate each conceivable feature meaning (e.g., each word in the vocabulary) with a d-dimensional vector for any d. Instead, these vectors are called model parameters, and trained along with the other parameters (Goldberg, 2017). That explains why word embeddings can outperform the traditional NLP frameworks. Traditional NLP frameworks count mainly on parsing sentences to extract the text features, but word embeddings count on the dense vector representation of the text regardless of how the text is really structured.

Given the above explanations, **DeepKAF** is using word embeddings to be able to tackle more text representations challenges. Autoencoders that are part from the pre-processing layer are going to be explained in the next sections.

5.6.3 DeepKAF and Siamese Networks

A systematic approach to building similarity measures between textual cases is important for effective CBR applications. In order to ease the task of building similarity measures, **DeepKAF** had been experimentally compared with several neural network architectures (Amin et al., 2018c), using the straightforward cosine distance and averaging word vectors trained using word2vec and its descendants (Amin et al., 2018a). Each approach had its own pros and cons, as described in **Chapter 6** during the evaluation process. The Siamese neural network architecture was the extraordinary solution that boosted the overall performance of the applications built on top of **DeepKAF**. SNN algorithm was first introduced in (Bromley et al., 1993b) to verify signatures written on a touch-sensitive pad. The twin fully identical sub-networks approach has been used intensively in many high-performance real-time object tracking tasks like CFnet (Valmadre et al., 2017), StructSiam (Yunhua Zhang et al., 2018a), SiamFC-tri (Dong and Shen, 2018), DSiam (Guo et al., 2017), SA-Siam (He et al., 2018), SiamRPN (B. Li et al., 2018), DaSiamRPN (Zhu et al., 2018), Cascaded SiamRPN (Fan and Ling, 2018), SiamMask (Wang et al., 2018), SiamRPN (Bo Li et al., 2018), Deeper and Wider SiamRPN (Z. Zhang, Peng, and Wang, 2019). As shown, SNNs are very popular in finding what makes two objects similar. The SNN can find similarity or a relationship between two comparable objects, and most of the historical work is in image similarities. Siamese architecture is perfectly suited to situations where there are just a few samples per class, as in the **DeepTMS** use case described in **Chapter 6**.

Based on the experiments that have been carried out in this research, SNN with two LSTM sub-networks, excelled in the area of finding text similarities. In the CBR literature, to the best of my knowledge, Siamese networks have not been applied in TCBR applications. Interesting research has been carried out using SNNs to build similarity measures in CBR as part of a project to generate the similarity knowledge on the SelfBACK dataset (Martin et al., 2017). In this work convolutional Siamese network implementations have been used to

build similarities between different cases in a dataset that contains time series data. Results were baselined against a typical CNN, and the Siamese architecture outperformed the CNN implementation. The difference between the implementation mentioned in (Martin et al., 2017) and the work presented in this paper is in how Siamese Networks are being used and trained. The SelfBACK dataset does contain numerical time series data and not textual data. Comparing to **DeepKAF**, in (Martin et al., 2017) authors trained the Siamese architecture over the entire case not on specific attributes. This is different from the **DeepKAF** approach, which trains and dedicates a model for every attribute to generate the local similarities and then calculate the global similarity, as described in **Chapter 5**. Using a Siamese Network per attribute helped in having a model that preserves relevant information about each attribute and not be distracted by other information that might decrease the accuracy. Long-short term memory is very successful in solving NLP problems, as shown in (X. Li and Wu, 2014; Yunhua Zhang et al., 2018b). LSTM architectures use the internal memory to remember or use information across long input sequences. They can evaluate, classify and retain the related information about each part of the sentence and in the whole paragraph. The local context of any input sentence turned out to be useful for additional details on a word in a sentence and to eventually enhance the interpretation of the sentences. In our tests, the local context enhanced the estimation of the consistency of the sentence by reducing the mean squared error (MSE) and increasing the correlation value.

5.6.4 DeepKAF and Autoencoders

One of the very challenging tasks that the **DeepKAF** implementation has faced was the pre-processing of the input text. Text pre-processing is simply the task of putting the text into the shape that is analyzable for the intended purpose. Text pre-processing is difficult because what could be applied to an input text can be the worst to be done for a different task.

During experimenting with **DeepKAF** implementations, the main pre-processing challenges can be summarized in the following:

1. Noisy input text with unnecessary parts, for example, signatures and greetings
2. Highly dimensional input text (too long) like what is described in the CaseLaw experiment presented in **Chapter 6**

Before using autoencoders as a core part in the **DeepKAF** pre-processing layer, an LSTM model was used to help in identifying the unnecessary parts of the input text and then use the traditional principal component analysis (PCA) to reduce the text dimensionality. This approach worked perfectly fine and very promising results were achieved. However, to train the LSTM to be able to identify the unnecessary parts in the text was a road-blocker. It was an excessive task to prepare a dataset that has a sufficient number of cases to train the LSTM model. Therefore, the idea came up of following an unsupervised learning approach to do that task of finding the best representation of the input text and do the dimensionality reduction at the same time. Autoencoder is a certain form of an unsupervised feed forward neural network that encodes input $x$into hidden layer h and then decodes it back from its hidden representation. Autoencoders are popular in the tasks of dimensionality reduction and denoising data regardless of the data type. Autoencoders have applications in image processing (Mabu et al., 2018); S. Chen et al., 2017, acoustic analysis (Abeßer et al., 2017) and NLP (Miao, Yu, and Blunsom, 2015; X. Zhang et al., 2019; Fu et al., 2018).

DeepKAF is targeting mainly TCBR applications, and in TCBR, sequence processing can be very challenging, not only because the length of the input sequence may vary. This is difficult since machine learning algorithms, and neural networks in particular, are built to operate with fixed-length inputs. An additional problem with textual data is that the timing of measurements can make it challenging to extract features that are appropriate for use as inputs to supervised learning models, frequently involving specific expertise in the field or in the field of signal processing. Finally, certain predictive modeling issues involving

sequences involve a guess that they themselves are indeed sequences. Hence, it is essential to have an approach that is able to put the input text in the shape that other deep learning models within **DeepKAF** can process. There were two main candidates for text denoising and dimensionality reduction, namely PCA and autoencoders

How autoencoders are different from PCA: In essence, PCA is restricted to a linear representation, while autoencoders can have nonlinear encoder/decoders. Hence, autoencoders can be exactly the same as PCA if the linear encoder and decoder are being used with square error loss function and normalized inputs Plaut, 2018. In (Almotiri, K. Elleithy, and A. Elleithy, 2017), authors mentioned that the AE approach achieved 98% accuracy compared to 97% accuracy using PCA. Given that the training time for AE was 9+ hours, comparing to less than an hour to train a PCA model, the authors chose to follow the PCA approach. However, based on the work in (Plaut, 2018), with a growing number of input features, PCA would result in slower performance relative to autoencoders.

Based on the literature and understanding of the challenges that **DeepKAF** implementations have to deal with to improve the state of the art, autoencoders were chosen as the approach to denoise the input text and decrease dimensionality. **Chapter 6** will show how the combination of skip-thought autoencoders and Siamese MaLSTM improved the retrieved results. Autoencoders helped in minimizing the effort required to prepare datasets to train the models that are going to eliminate the unnecessary part of the text.

5.7 Summary

This chapter presented the architectural view of **DeepKAF** and the models applied. It has been explained how each CBR process is executed via a deep learning model. The underlying decisions being made have been justified based on the process of developing the respective components. As we are dealing with a highly dynamic AI sub-field, the related state-of-

the-art literature in the area of using deep learning in CBR has been demonstrated with clarification how this work is different from **DeepKAF**. The architecture helps in abstracting the idea behind **DeepKAF** because the technical details and deep learning models can change based on the domain knowledge. Therefore, abstracting **DeepKAF** as described in this chapter is important for the reproducibility of this research (and for going beyond). In the following chapters, more technical details are provided, together with the experimental evaluation that shows how **DeepKAF** was able to improve the efficiency/quality of a CBR system.

This chapter also explained the competitive advantage of **DeepKAF** over other CBR approaches in general and TCBR in specific. The discussion in the chapter started with comparing **DeepKAF** with traditional textual CBR approaches and how **DeepKAF** can provide an edge to any CBR system. A review from the literature and popular work in the area of TCBR has been presented to show the difference between **DeepKAF** and those approaches. The chapter continued with covering the another important part of **DeepKAF**, namely complex NLP tasks. Section 5.5 explained how **DeepKAF** is using DL models without hiding the explanation advantage of any CBR system. Section 5.6.2 explained why NLP tasks are challenging and what makes an NLP task a complex one. The section also showed how **DeepKAF** is designed with such complex NLP tasks in mind and how to solve them. Sections 5.6.3 and 5.6.4 clarified why specific decisions were made and why specific models were chosen. In those sections, a glimpse of history was provided and comparison with other approaches was introduced.

Chapter Six

DeepKAF Experiments and Validation

6.1 About this Chapter

Chapter 5 described the architecture and the processes within **DeepKAF**. In order to apply **DeepKAF** effectively and show the real benefits, three requirements have to be fulfilled: First having domain-specific textual data, second having a stream of textual data that requires real-time analysis, and third having domain experts that can validate the retrieved cases. This chapter provides the technical view of the **DeepKAF** implementation and the experiments that have been carried out to evaluate the efficiency. For the evaluation of **DeepKAF**, two experiments have been carried out to investigate the cross-domain applicability of the framework. Two different use cases and two different data sets have been used: one from a private automotive industry (**DeepTMS**) (Amin et al., 2018a) and the second one is from the law domain (*Caselaw Access Project* 2018), respectively. Both have been implemented using a dedicated cloud server hosted on Google Cloud Platform (*Google Cloud Platform* 2018) that allowed me to cope with the amount of data that were not able to be used on a desktop PC. For both experiments, **Keras** and **NLTK** python frameworks have been used to build and train the models, along with other frameworks that will be explained with each experiment.

6.2 Experiment 1 - Deep Ticket Management System (DeepTMS)

DeepTMS is the software name that is built on top of **DeepKAF**. This experiment has been focused on implementing a ticket management system by using a hybrid approach, CBR with DNN and big data technologies. The aim was to improve the response time to any "incoming' tickets by identifying and presenting the most relevant solutions from similar problems using a historical knowledge base. Any retrieved solutions were presented and recommended in real-time support cases to a help-desk engineer.

Work on **DeepKAF** has been already experimenting extensively on trials to improve the accuracy of the framework as well as minimize the efforts to acquire new knowledge (Amin et al., 2018c; Amin et al., 2020); (Amin et al., 2018a; Amin et al., 2018b).

All historical research along with the findings of each experiment will be discussed to demonstrate various solutions and why some models have been chosen over others.

For this work, the implemented application and any used data were a joint application between the German Research Center for Artificial Intelligence (**DFKI**) and a multinational automotive company located in Germany with branches all over the world.

6.2.1 DeepTMS - Application Domain

Most companies have a dedicated internal help-desk team for customer support, since service quality is usually measured via customer satisfaction. Inside the company, most of the help-desk tickets come through emails to a dedicated help-desk team. Once received help-desk agents prioritize the tickets and assign them to specialist engineers inside the team to work on it. The company had several historical datasets describing a plethora of issues that have happened in the past along with proposed solutions to those.

A historical case could be represented in the form of problem description, solution, and

keywords. When new tickets arrive, a help-desk engineer should search within the company's knowledge base to confirm whether any solution(s) exist(s) or not. As reported by domain experts, their processes in place were suffering from the following issues:

1. A help-desk agent prioritizes or routes the ticket in the wrong way. Such an action can lead to longer times to a successful ticket resolution.

2. Lack of enough experience or deep knowledge of the help-desk engineer

3. It is not easy to find proposed solutions from a historical knowledge base and engineers find it detrimentally time-consuming and not always leading to a solution

The main aim of **DeepTMS** was to improve the response time to any "new", incoming tickets by identifying and presenting the most relevant solutions from similar problems using a historical knowledge base. These solutions then were presented and recommended in production to a help-desk engineer.

Cases representation: Historical cases were represented in the form of problem description, problem solution, relevant department, case classification, and keywords. A data audit on the historical cases reported: cases written in both German and English languages, grammar with errors and substantial amount of used domain-specific abbreviations. This pertained to the most common help-desk management systems challenges that usually are:

1. Large volumes of demand/ incident / support tickets coming in different formats and from a variety of channels

2. Very specialized and usually restrictive Service Level Agreements (SLA) that enforce help-desk departments to solve tickets in specific amount of time based on the company priorities

3. Error-prone prioritization or routing of tickets which can lead to imminent breach of SLA(s)

4. Substantial limitations in finding a suitable solution from a historical knowledge base. The vast majority of engineers have found it detrimentally time-consuming and not always been capable of finding a solution.

The following section will present our approach to this domain and the contribution in terms of SNNs

6.2.2 Related Work - CBR State-of-the-art in Ticket Management Systems

One of the most popular help-desk CBR system is **HOMER** (Göker et al., 1998). **HOMER** (T. R. Roth-Berghofer, 2004b) is a help-desk support system designer for the same purpose of **DeepTMS**. HOMER used an object-oriented approach to represent cases and used a question-answering approach to retrieve cases. HOMER showed very good results when it first presented in 1998. In 2004, it got even further improvements and gave better results than the first version. However, any existing fast-pace work environments demand solutions that are able to deal with big amounts of data in real-time with minimum human interference. Comparing to **DeepTMS**, we focused more on how to automate the extraction of similarities and deal with unstructured or mixed-languages text, but this approach also cannot be automated to be integrated in the real business environments.

6.2.3 The Dataset

A mixed-language customer support tickets' dataset that was collected from the automotive company in Germany (Amin et al., 2018a). The dataset contained German and English textual descriptions of problems along with their associated solutions and several other attributes, like the sender group, the department in question, and others.

6.2.4 The Challenges

After analyzing the application domain and the data we received, we identified the following challenges:

1. Building cases was a tedious and extremely time-consuming task for domain experts. Experts were not able to add much effort, and hence we resorted to as much automation during the build-up of the CBR system as possible.

2. Any existing knowledge base and new tickets were received in a bilingual format (English, German, or both), which added more complexity in the text analysis and pre-processing to build cases or retrieve similar cases.

3. Tickets were primarily written by non-native English or German speakers, and they could have contained several grammar mistakes or vague domain abbreviations.

speakers,

Due to the last two challenges, it was not possible to use any traditional NLP frameworks for text understanding like TwitterNLP and Stanford NLP, since their application did not lead to promising results. Therefore, we decided to use DNNs and word embeddings to improve the text pre-processing and similarity measures.

6.2.5 The Methodology

Text is used to express knowledge. Text is a collection of words in any well-known language that can convey a meaning (i.e., ideas) when interpreted in aggregation(Richter and Weber, 2013). To build a textual CBR system, we discussed the system process and how normally the help-desk agents prioritize and route tickets. From this process, four attributes were identified as key ones to make a decision. These were: 1. Email Subject 2. Email Content 3. Email Sender Group (the company was organized internally in different groups and each group had its own applications and systems). 4. The initial priority of the ticket assigned

by the team who reported it. Based on the above attributes, a help-desk agent would decide how to proceed with this ticket. Based on those discussions with experts, the decided CBR approach was as follows:

1. Case Generation: Since there were not too many attributes, cases were generated with flat attribute-value representation features

2. Case Retrieval: Due to the complexity of the NLP, case similarities needed a rich context-aware similarity measure. As such, a trained neural network for identifying and recommending solutions from the historical case base was selected.

3. Case Adaptation: Adaptation rules are not included during this implementation, but should be added in the next phases.

6.2.6 DeepTMS: The Solution Architecture

DeepTMS solution architecture consists of three main modules (see Figure 6.1):

1. Input Process (Data Generation) Module: This module is responsible for generating and simulating the emails (tickets) stream.

2. Map/Reduce-Hadoop-Cluster (Data Processing and Retrieval): This module is responsible for receiving the tickets and doing the ticket content pre-processing/processing, then retrieve the similar tickets from the Case Base (Case Generation, Retrieval and Retain).

3. Graphical User Interface (Data Visualization): This module is responsible for visualizing the results to the system end-users.

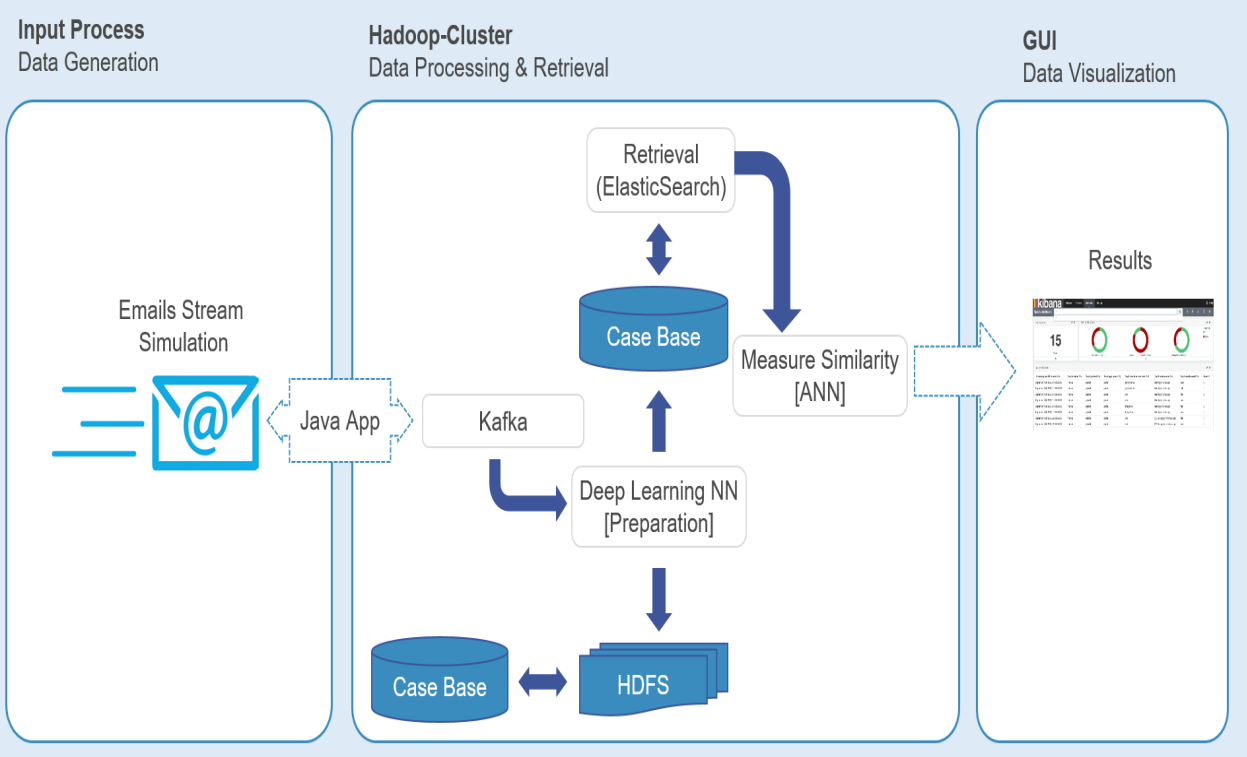

Figure 6.1 DeepTMS Solution Architecture

6.2.7 The Hybrid CBR Approach based on DeepKAF

The first decision we had to make in the development of **DeepTMS** was how we are going to handle the challenges mentioned before, and which approach we should apply. The selected approach combines a deep neural network with CBR to capture and decode domain knowledge. Our approach uses deep learning algorithms in the context of NLP. More specifically it applies them throughout the task of prioritizing emails based on their content and it measures text similarity based on their semantics. We, therefore, presented several neural network types to represent a sequence of sentences as a convenient input for our different models. First, we divided the emails into sub-groups based on the business sectors they were coming from. The first stage was the ticket pre-processing, which divided into five main processes (see Figure 6.2)

1. P1: Input Process (Data Generation): was responsible for generating and simulating

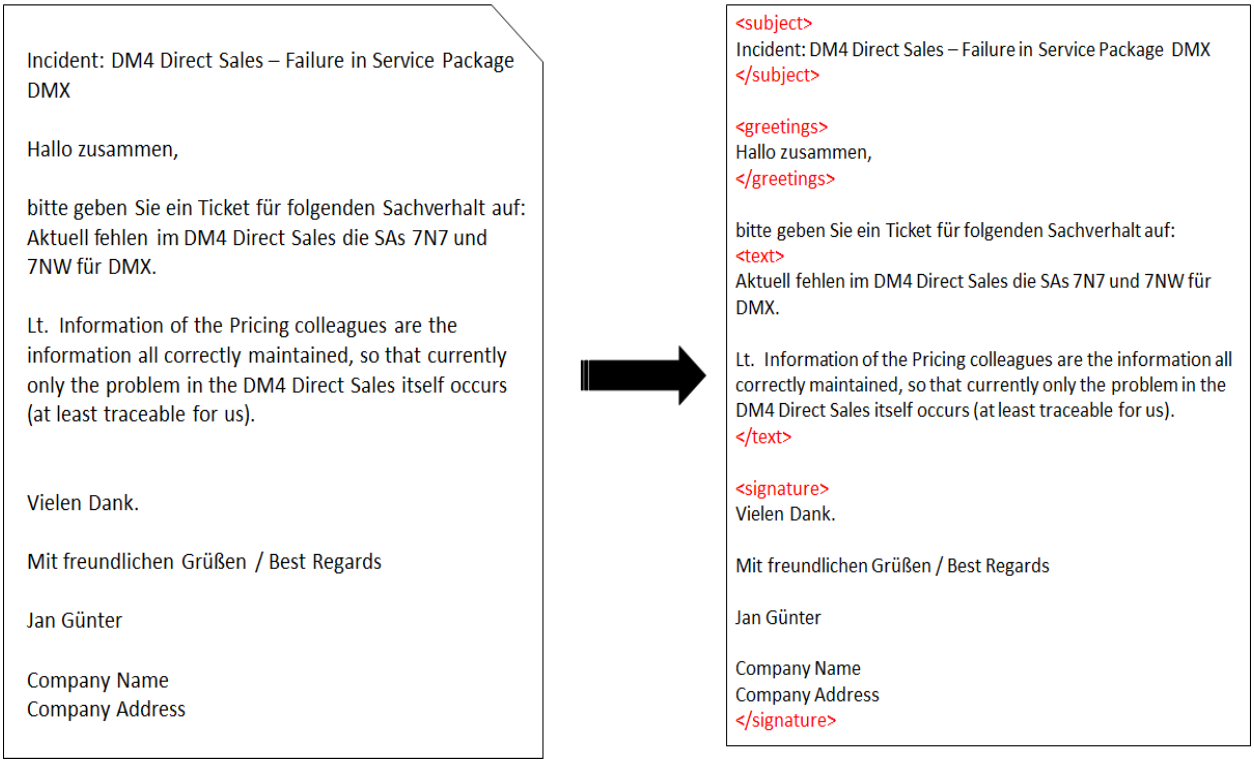

Figure 6.2 Ticket Pre-processing

the emails (tickets) stream

2. P2: Prioritization process: was prioritizing incoming tickets based on historical cases and their recorded priorities

3. P3: Greetings filter: which identified and eliminated any unnecessary text (e.g., greetings, signatures, etc.) from any email

4. P4: Stemming and stop words elimination: in either German or English language

5. P5: Text vectorization

Vocabulary Containers

Vocabulary is one of the knowledge containers and represents the information collected from the domain to express knowledge (Richter and Weber, 2013). By filling in this container,

I identify terms that are useful for the main system tasks. The acquisition of the domain vocabulary has direct effect on the system performance, and that's why it is usually done with intensive help from domain experts. As mentioned in Section 3.2 utilizing several experts to manually assist with decoding domain knowledge was rather expensive, therefore an alternative was sought. In order to improve the acquired vocabulary, I followed the typical three methods described in (Richter and Weber, 2013). Deep neural networks have been used to remove irrelevant words and extracted the main features that represent certain text using the Word2Vec models (Mikolov, K. Chen, and Corrado, 2013). In the next section, I describe how exactly Word2Vec worked to build neural word embeddings.

Neural Word Embedding

Most of the deep learning models are not able to process strings or plain text. They require numbers as inputs to perform any sort of job, classification, regression, etc... Many current NLP systems and techniques treat words as atomic units, therefore, in order to apply a deep learning model to NLP, we need to convert words to vectors first. Word embedding is the process of converting text into a numerical representation for further processing. The different types of word embeddings can fall into two main categories:

1. **Frequency-based embedding (FBE)**:

   FBE algorithms focus mainly on the number of occurrences for each word, which requires a lot of time to process and exhaustive memory allocation to store the co-occurrence matrix. A severe disadvantage of this approach is that quite important words may be skipped since they may not appear frequently in the text corpus.

2. **Prediction-based embedding (PBE):**

   PBE algorithms are based on neural networks. These methods are prediction based in the sense that they assign probabilities to seen words. PBE algorithms seem the present state-of-the-art for tasks like word analogies and word similarities.

PBE methodologies were known to be limited in their word representations until Mitkolov et al. introduced Word2Vec to the NLP community (Mikolov, K. Chen, and Corrado, 2013). Word2vec consists of two neural network language models: A continuous bag of words and skip-gram. In both models, a window of predefined length is moved along the corpus, and in each step the network is trained with the words inside the window. Whereas the CBOW model is trained to predict the word in the center of the window based on the surrounding words, the skip-gram model is trained to predict the context based on the central word. Once the neural network has been trained, the learned linear transformation in the hidden layer is regarded as the word representation. In this work we have used skip-gram model since it demonstrates better performance in semantic task identification (Altszyler, Sigman, and Fernández Slezak, 2016).

Text Pre-Processing

In the text pre-processing stage, raw text corpus preparation tasks are taking place in anticipation of text mining or NLP. We trained our Word2Vec model over the ticket corpus overall to build cases used in similarity measures. As any text pre-processing tasks, we have two main components: 1. Tokenization, 2. Normalization. Tokenization is a step which splits longer strings of text into smaller pieces, or tokens. Normalization generally refers to a series of related tasks meant to put all text on a level playing field: converting all text to the same case (upper or lower), removing punctuation, converting numbers to their word equivalents, and so on. Normalization puts all words on equal footing, and allows processing to proceed uniformly. Normalizing text can mean performing a number of tasks, but for our approach, we will apply normalization in four steps: 1. Stemming, 2. Lemmatization 3. Eliminating any stopping words (German or English) 4. Noise Removal (e.g. greetings and signatures). In essence, we can consider the Word2Vec model or any other model that could be built as a substitution to the traditional taxonomies.

Similarity Measures

Similarity measures are highly domain-dependent and used to describe how cases are related to each other. In CBR, comparison of cases can be performed along multiple important dimensions (Ashley, 1991; Brüninghaus and Ashley, 1998). Cases that only match partially, can be adapted to a problem situation, using domain knowledge contained in the system (Aleven, 1998). Thus, methods, like in particular information retrieval, which are based only on statistical inferences over word vectors, are not appropriate or sufficient. Instead, mechanisms for mapping textual cases onto a structured representation are required. A basic assumption for applying the principle for similarity measures is that both arguments of the measure follow the same construction process. This allows us comparing the corresponding sub-objects systematically. For the system, I defined the two types of similarity measures: Local similarity measures and global similarity measures. Local similarity measures describe the similarity between two attributes, and the global similarity measures describe the similarity between two complete cases. In the next section, we elaborate how we applied the local similarity measures followed by the global similarity measures.

Local similarity measures: Based on the collected data and the discussions with experts, we defined the local similarity measures. We have mainly four attributes which are distinctive, except for the email subject and content. For the Priority (integer) and Sending Groups (distinctive strings) we used distance functions. For the email subject and content, we counted upon the Word2Vec model to give us the similarity degrees between different texts, after applying all the aforementioned prepossessing tasks.

Global similarity measures: The global similarity measure defines the relations between attributes and gives an overall weight to the retrieved case. The weight of each attribute demonstrates its importance within the case. The weighted Euclidean distance

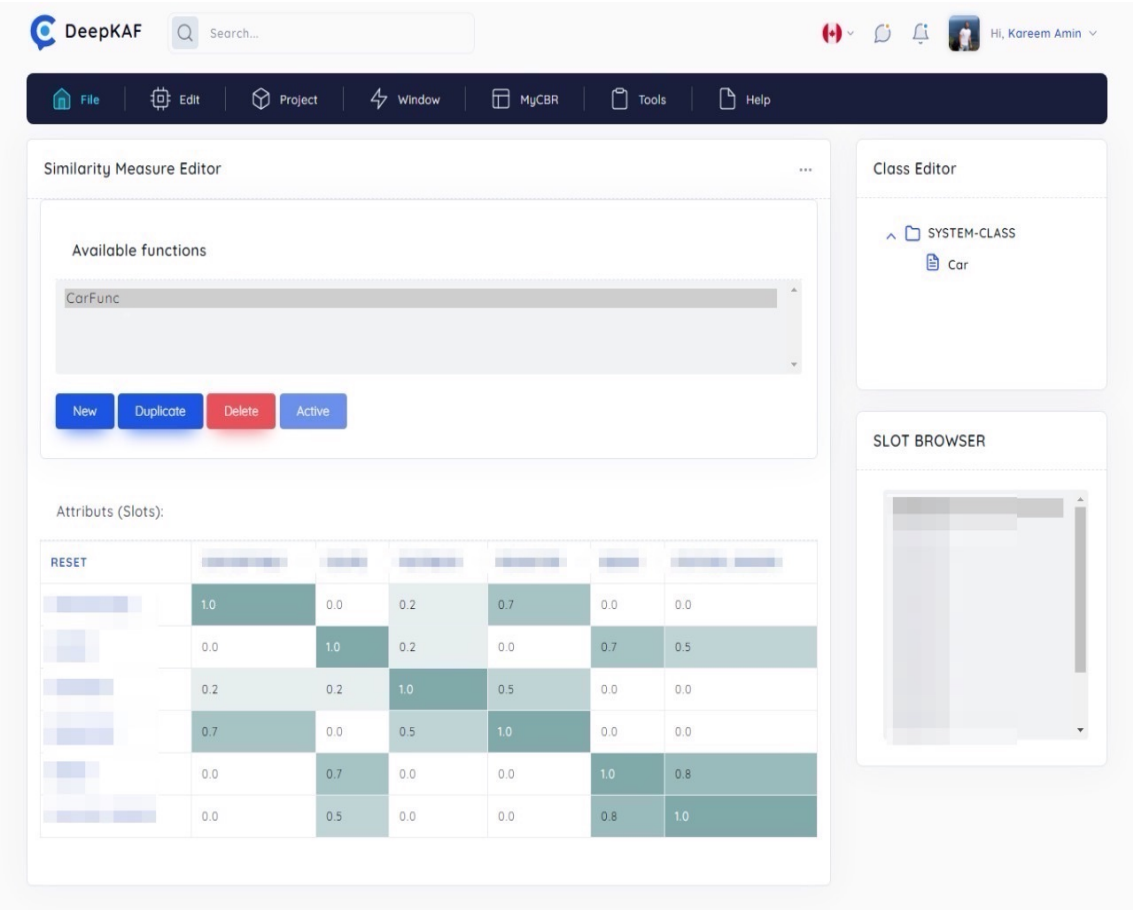

Figure 6.3 DeepKAF similarities per attributes

was chosen for the calculation of the global similarity, as applied in (Bach, K. Althoff, et al., 2011) . The weight of each attribute has been defined in collaboration with the domain experts. We decided to use a weight range between 1 and 5. The most important values are weighted with 5.0 and 4.0 determined by the experts on which attribute value they would use to evaluate the case. They have decided to give the following weights to the attributes: Priority = 2.0, email content = 4.0 or 5.0, email subject = 2.0 or 3.0, sending group = 3.0 or 4.0). After giving the weights to the attributes, we then sum up the given weights then normalize the result to get a value between 0.0 and 1.0 and come up with the overall global case similarity.

6.2.8 Experiment 1 - Evaluation Results

For experiment 1, three approaches with different combination of deep learning models have been used in the aim of improving the retrieval results. The four paths are going to be described as the following:

1. LSTM and Word2Vec
2. LSTM and fastText
3. Siamese Networks and Word2Vec
4. Autoencoders and Siamese Networks and Word2vec

LSTM and Word2Vec - Evaluation Results

In the beginning, our approach was to use SVM and vectorization to prioritize emails (Amin et al., 2018a). Early results from this approach were promising, but not in sub-group cases. When we performed a more intensive test with a large volume of emails, it failed to prioritize with high accuracy. Therefore, we decided to build several states of the art neural network models: CNNs, RNNs, and LSTMs (Hochreiter and Schmidhuber, 1997) to test and compare their results. Deep neural network applications seemed to perform substantially better on all sub-groups.

This evaluation is divided into two parts:

1. The case priority given by the neural network
2. The retrieved cases and suggested solutions to the new case

During our system testing and evaluation phase, we decided to use different neural network models to explore, validate and compare accuracy results for each and every model. We applied three neural network models: CNNs, RNNs, and LSTMs (Hochreiter and Schmidhuber, 1997). Word2Vec was applied to vectorize text input and build word representations

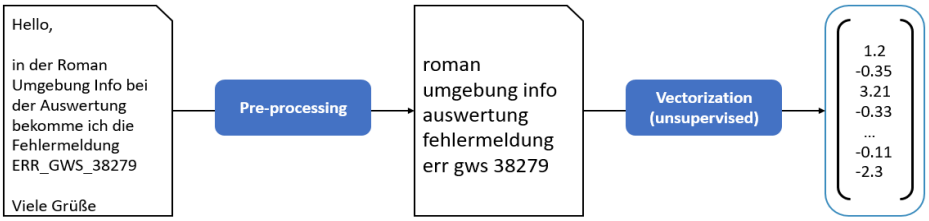

Figure 6.4 Text Vectorization

in the vector space (See Figure 6.4). Sequences of such vectors were processed using various neural net architectures.

Word2Vec was built using 300,000 historical tickets in an unsupervised training mode. All networks were built with one hidden layer, and utilized the Word2Vec model we have already built. To train the three different neural net models, we have also used 300,000 old tickets with known priorities in a supervised learning process. An additional 10,000 tickets were used to evaluate the models in prioritizing the test tickets automatically. Table 6.1 summarizes the prioritizing stage results.

Table 6.1 Prioritization Results

Neural Network Model	**Accuracy**	**Precision**	**Recall**	**F1**
Convolutional Neural Network **(CNN)**	82.67%	82.52%	82.64%	82.58%
Recurrent Neural Network **(RNN)**	89.28%	89.19%	89.27%	89.23%
Long Short-Term Memory Network **(LSTM)**	92.35%	92.13%	92.23%	92.16%

The second evaluation part is retrieving similar cases based on the similarity measures we defined before, and using Word2Vec model to give the degree of similarity between two texts. Since the LSTM model showed the best results in prioritizing the tickets, we continued to build our solution with LSTM models.

In the **Results Discussion** section, we are presenting details about the difference be-

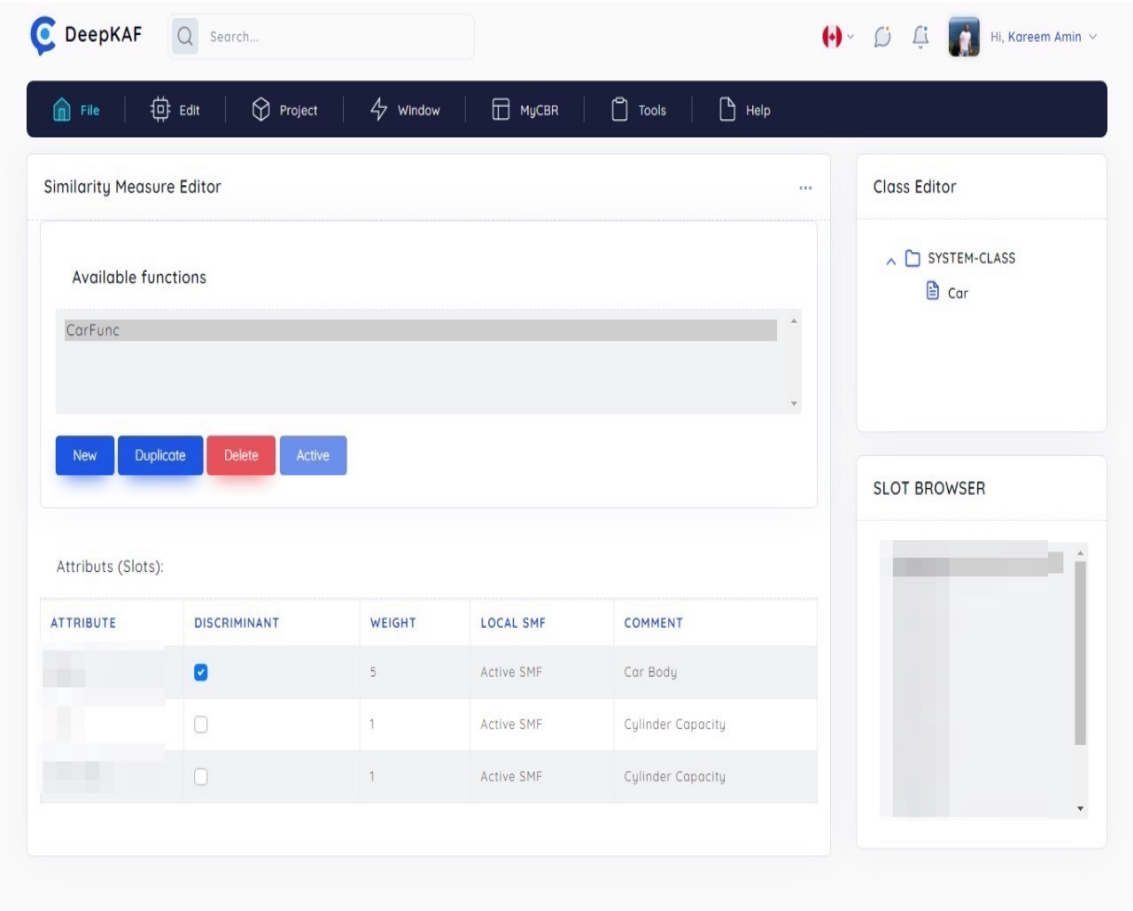

Figure 6.5 DeepKAF - Query

decide where the most relevant solution was positioned among the retrieved ten. We defined also four levels that the most relevant solution could belong to. These were: 1. **one:three** 2. **four:seven** 3. **eight:ten** 4. **Not Listed**. For the evaluation we used the same 10000 test tickets that were used in the prioritization stage. Table 6.2 shows the results for this stage.

Table 6.2 LSTM and Word2vec Retrieval Results

Level	**Number of Cases**	**Percentage**
One : Three	7764	77.64%
Four : Seven	1468	14.68%
Eight : Ten	692	6.92%

Table 6.2 LSTM and Word2vec Retrieval Results

Level	**Number of Cases**	**Percentage**
Not Listed	76	0.76%

Results Discussion and Lessons Learned From the first evaluation results and during the implementation, there were lessons learnt that led to the enhancements that are going to be demonstrated in the next evaluations. During the implementation of **DeepTMS**, neural networks have been used in tickets pre-processing to eliminate the redundant text and pass the most relevant text to deep neural networks for prioritization purposes. For both tasks, LSTMs outperformed all the other neural network models we used. It is recommended to use LTSM for text related tasks, but it is also important to mention that it takes longer time both for its training phase, and for text processing afterwards. CNNs are more appropriate for image-related tasks. However, we investigated them since the literature suggests them as appropriate to areas where changes take place in the network architecture and can give promising results in text processing as well (Y. Kim, 2014). CNNs are faster in training and processing phases than RNNs and LSTMs. Since an LSTM is a special RNN case, it seemed to perform well on text tasks, better than standard CNNs and worse than LSTMs. In terms of training and processing performance, they take longer than CNNs and less time compared to LSTMs.

For building the Word Embedding using Word2Vec and use them within the neural networks models, the performance is pretty good, and it can get improved with more text we use in building the model, since it expands the word corpus and improves the ability to find relationships between words. We started building the Word2Vec model with 50000 tickets, and the results were worse compared to training with six times more tickets.

LSTM and fastText - Evaluation Results

In this evaluation (Amin et al., 2018b), **fastText** (Joulin et al., 2016) word embeddings model has been used instead of **Wrod2vec**. As mentioned in section 6.2, LSTM model is used to do the pre-processing tasks and prioritize the incoming tickets, and it scored a total F1 score: **92.16%** in prioritizing tickets correctly, which is exactly the same as the previous evaluation. The new experiment carried out in this research is to use the **fastText** model in the retrieval process and compare it with the **Word2vec** model that was used before.

fastText was built using 300,000 historical tickets in an unsupervised training mode. To train the LSTM model, we needed a labeled dataset containing pairs of sentences and an attribute to identify the similarity degree between those two sentences. For that goal, a manually generated dataset was prepared based on historical data which contained 5000 pairs of sentences and have assigned a degree of similarity that varied between 1:5 for each pair. Table 6.3 shows a comparison in retrieval results between the Word2vec model that was implemented and fastText model. **DeepTMS** suggested ten solutions to a new ticket, and then experts were called to decide where the most relevant solution was positioned among the retrieved ten. We defined also four levels that the most relevant solution could belong to. These were: 1. one:three 2. four:seven 3. eight:ten 4. Not Listed. For the evaluation we used 10000 Test Tickets that were used in the Prioritization stage.

Table 6.3 fastText vs Word2vec Retrieval Results

	fastText		**Word2vec**	
Level	**Number of Cases**	**Percentage**	**Number of Cases**	**Percentage**
One : Three	5144	51.44%	7764	77.64%
Four : Seven	1431	14.31%	1468	14.68%
Eight : Ten	1682	16.82%	692	6.92%
Not Listed	1743	17.43%	76	0.76%

Results Discussion and Lessons Learned From Table 1, we can summarize the findings as the following:

1. From a practical perspective, the choice of hyper-parameters for generating fastText embeddings becomes key since the training is at character n-gram level. That's one of the reasons why fastText might be giving worse results than Word2vec

2. fastText training takes longer to generate embeddings compared to Word2vec

Based on the above findings, **DeepKAF** implementation continued with Word2vec as the word embeddings model to be used.

Siamese Network and Word2Vec - Evaluation Results

Towards the goal of improving the overall **DeepTMS** accuracy and minimize the effort needed from the domain experts. In this evaluation (Amin et al., 2020) the LSTM model that has been applied to do the retrieval task is going to be replaced by a Siamese Network Architecture, but there is still another LSTM model that is still being used to do the pre-processing task which outperformed all the other models that were used. before, **Why Siamese Networks?** Siamese networks typically contain two or more identical sub-networks which share parameters and weights of the same configuration. Siamese neural networks are capable of identifying similarities or connections between different objects. Typically, two identical sub-networks are used to process similar inputs, and another module will take its outputs to a final output(Bromley et al., 1993a). A Siamese neural network architecture was chosen as for the experimental setup since it provides the following advantages:

1. It is robust compared to CNNs and RNNs in processing complex text

2. It can provide better text embeddings

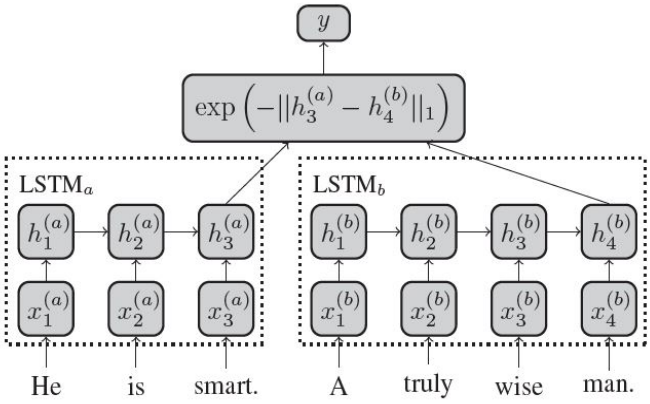

Figure 6.6 MaLSTM Network Architecture

3. By sharing weights across sub-networks, they reduce the number of parameters to train for. This practice reduces the need for training data required and is less possible to overfit

Siamese Manhattan LSTM Model (MaLSTM): Manhattan LSTM (MaLSTM) model was introduced in (Mueller and Thyagarajan, 2016) and showed remarkable results in finding the Semantic Relatedness among sentences and outperformed any existing baselines for semantic relatedness. This performance is achieved by using specific hidden units (i.e. dimensions of the sentence representation) to encode specific sentence characteristics, having as a result a trained **MaLSTM** that infers semantic relevance among sentences by simply aggregating their differences across a set of characteristics (see Figure 6.6).

MaLSTM model has two networks $LSTM_a$ and $LSTM_b$ correspondent to two sentences respectively in any chosen sentence-pair. In this work we have selected Siamese architectures with "tied" weights such as that $LSTM_a = LSTM_b$. The general untied version of this model seems more useful for applications in asymmetric domains such as information retrieval (where search queries are stylistically distinct from stored documents). **MaLSTM** has been tested among different tasks like Sentence Representation, semantic sentences similarity measures and sentences paraphrase detection.

In **DeepTMS** there are mainly two deep neural networks applied: 1. An LSTM model to do the pre-processing and prioritization 2. A **MaLSTM** Siamese model to measure similarities between two distinct sentences. For (1) previous work has presented a LSTM model with efficient results in doing pre-processing and prioritization tasks correctly. An LSTM model with Word2Vec was used to measure similarities that although they showed good results in measuring distance between sentences, they did not give high accuracy when tested with more complex sentences, To improve this, Siamese networks were adopted and applied. Hence, for (2) a **MaLSTM** model is applied and its results are compared with the old LSTM model implemented to measure similarities. The experiment carried out in this evaluation is to use the**MaLSTM** model in the retrieval process and compare it with the results presented in previous work.**Word2Vec** is still being applied to vectorize text and build dense word representations in the vector space.

The Applied Model Structure: Model optimization is one of the greatest challenges in the implementation of machine learning solutions. Hyper-parameters are variables that determine both the network structure as well as how the network is trained. During building the **MaLSTM** model and based on previous work on **DeepTMS**, the hyper-parameters have been adjusted to be as the following:

1. **Neuron Activation Function** = Relu
2. **Learning Rate** = 0.1
3. **Regularization** = L2 (.001)
4. **LSTM Layer Size** = 200
5. **Batch Size** = 32
6. **Epochs** = 47

7. **For the output layer we used Loss Function** = MCXENT and **Activation Function** = SOFTMAX

These hyper-parameters are defined after several trials upon which we have been comparing the outcomes. The above combination seemed to yield the most accurate results.

Models Training and Sentences Pair Creation: **Word2Vec** was built using **300,000** historical tickets in an unsupervised training mode. To train the **MaLSTM** model, a labelled dataset was required containing pairs of sentences and an attribute to identify similarity degrees between those two sentences. For that goal, a manually generated dataset was prepared based on historical data and containing **5000** pairs of sentences and assigned a degree of similarity varies between 1:5 to each pair.

For the evaluation, **10000** Test Tickets were used as a stream of new tickets that required a solution. **Table 1** presents a comparison in retrieval results between the old LSTM model and **MaLSTM**. The evaluation process was done with company domain experts and technicians. **DeepTMS** suggested ten solutions to a new ticket, and then experts were called to decide where the most relevant solution was positioned among the retrieved ones. We defined also four levels that the most relevant solution could belong to. These were: **1. one:three 2. four:seven 3. eight:ten 4. not listed**.

Table 6.4 LSTM vs MaLSTM Retrieval Results

	LSTM		**MaLSTM**	
Level	**Number of Cases**	**Percentage**	**Number of Cases**	**Percentage**
One : Three	7764	77.64%	8354	83.54%
Four : Seven	1468	14.68%	1510	15.1%
Eight : Ten	692	6.92%	30	0.3%
Not Listed	76	0.76%	106	1.06%

Results Discussion and Lessons Learned From **Table 6.4**, the findings are as the following:

1. **MaLSTM** model gives better results in retrieving the most suitable cases among the top three and seven retrieved ones. However, **MaLSTM** was not able to provide any correct solution in the top ten results in more cases than the old LSTM. From these results, more investigation should be carried out on whether there are any further improvements that could be implemented with **MaLSTM** to get better results.

2. **MaLSTM** took longer training time than the LSTM model and required more processing power in order to get trained.

3. **MaLSTM** took longer to retrieve cases and measure similarities than the LSTM model.

4. **Word2Vec** is outperforming in terms of building dense word embeddings model that is applied on multi-lingual text. **Word2Vec** performance always gets improved with more text is being during the training phase, since it expands the word corpus and the ability to find relationships. **Word2Vec** was initially built with **50,000** tickets, and the results were worse compared to training with full **300,000** training set.

Autoencoders and Siamese Networks and Word2vec - Evaluation Results

In this evaluation (Amin et al., 2019), autoencoders have been used as a self-supervised learning approach that can find the best representation for the input text without the need for the pre-processing task that was done earlier by the LSTM model. Throughout the implementation of **DeepTMS**, the accuracy has improved by applying and comparing several models to determine the best candidate to decode complex domain knowledge. Word embeddings models combined with several deep neural networks models have shown promising results

Amin et al., 2018c;Amin et al., 2018a;Amin et al., 2020. However, some of the previously applied deep neural networks models require substantial supervised training over considerably large amounts of data to start being regarded as reliable, especially the models used to predict relevance between two distinct sentences. To minimize the degree of supervised learning in this work, LSTMs are replaced with autoencoders self-trained models that can summarize and extract features from text. The use of autoencoders can have a further advantage, this of capturing linguistic phenomena (such as word-level dependency) that Word Embeddings models are not able to capture. An observation that derived throughout our experimentation was that changing the type of input information has a profound effect on the kind of similarities the vectors will eventually capture. In such case the application of autoencoders can help to improve the quality of the input data which leads to better model accuracy. The experiment carried out in this research is to combine the **MaLSTM** with **autoencoders (skip-thought)** model in the text pre-processing and retrieval processes and compare it with the results presented in the previous work which used the **MaLSTM** alone.

Skip-thought autoencoder vectors is an approach that was introduced by Kiros et al. (2015) (Kiros et al., 2015). The authors describe a methodology for unsupervised learning of a both generic and distributed sentence encoder. That is, an encoder maps words to a sentence vector and a decoder is used to generate the surrounding sentences. In this setting, an encoder is used to map, for example, an English sentence into a vector. The decoder then conditions on this vector to generate a translation for the source English sentence. Sentences that share semantic and syntactic properties are mapped to similar vector representations, see **Figure 6.7** for a typical skip-thought model architecture. Consequently, the authors introduce a simple and rigid vocabulary expansion method to encode words that were not seen as part of training, allowing the skip-thought model the ability to process any unknown words. **skip-thought** models are inspired by the skip-gram structure used in Word2Vec (Mikolov, K. Chen, and Corrado, 2013) however, the authors propose an objective function

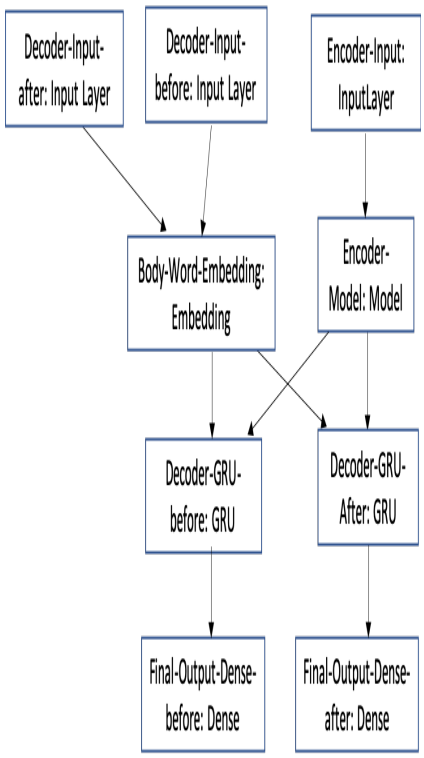

Figure 6.7 Skip-thought Model Architecture

that abstracts this idea to sentence level instead of individual words.

Encoder. Let $w_1^i, ..., w_N^i$ be the words in a sentence, s_i where N is the number of words in the sentence. At each time step, the encoder produces a hidden state h_t^i which can be interpreted as their presentation of the sequence $w_1^i, ..., w_t^i$. The hidden state h_N^i thus represents the full sentence.

Decoder. The decoder is a neural language model which conditions on the encoder output h_i. The computation is similar to that of the encoder except we introduce matrices C_z**,** C_r and C that are used to bias the update gate, reset gate and hidden state computation by the sentence vector. One decoder is used for the next sentence, s_{i+1} while a second decoder is used for the previous sentence s_{i-1}. Separate parameters are used for each decoder, except for the vocabulary matrix **V**, which is the weight matrix connecting the decoder's hidden state for computing a distribution over-words.

Training During the Training Phase, three models were used, each of them being trained in a different way.

1. **Skip-thought autoencoder**: Through a self-supervised training mode, the autoencoder was trained with over 300,000 tickets to find an intermediate low-dimensional representation for the content of each available support ticket.

2. **Word2Vec**: A Word2Vec model was trained by using an unsupervised training mode in order to build the required neural word embeddings from the word corpus (Amin et al., 2018a).

3. **Siamese MaLSTM**: A Siamese Network was then trained in a supervised training mode in order to learn the semantic similarities among different sentences (Amin et al., 2020).

Model optimization is one of the greatest challenges in any implementation of deep learning solutions. Hyper-parameters are variables that determine both the network structure as well as how the network is trained. After a substantial amount of different configuration settings and based on previous work on **DeepTMS**, the hyper-parameters with the optimum output were identified and are shown in **Table 6.5**:

Table 6.5 Models hyper-parameters

Skip-thought Autoencoder	**Word2Vec**	**MaLSTM**
Embedding Dimension = 150 **Latent Dimension** = 128 **Batch Size** = 64 **Epochs**= 76 **Optimizer** = Nadam **Learning Rate** = 0.0001 **Decoder Activation Function** = SOFTMAX	**Word Vectors Dimensioality** = 150 **Window** = 10 **Min Word Frequency** = 10 **Epochs** = 34 **Learning Rate** = 0.0001 **Activation Function** = SOFTMAX	**Neuron Activation Function** = Relu **Learning Rate** = 0.0001 **Regularization** = L2 (.001) **LSTM Layer Size** = 200 **Batch Size** = 64 **Epochs** = 47 **Output Layer Loss Function** = MCXENT **Activation Function** = SOFTMAX

Experiment Results

The system is evaluated by the ability to provide better solutions in the top 10 and decreasing the probability of not recommending any solutions. To evaluate the model, 10,000 test tickets were simulated as a stream of new tickets (as an industrial prerequisite) that required a solution in real-time. Table 6.6 presents a comparison in retrieval results between using **MaLSTM** alone (the old approach) and the new approach **skip-thought MaLSTM**. The evaluation process was done both with company business domain experts, help-desk engineers and domain technicians.**DeepTMS** suggested ten solutions to each ticket, and then experts

were called to decide where the most relevant solution was positioned among the retrieved ones. To ease the task of solution ranking in terms of accuracy, we defined four levels of relevance that a solution could belong to. These were: 1.Very relevant **one:three** 2. Relevant **four:seven** 3.Low-confidence **eight:ten** 4. **Not Listed**.

Table 6.6 MaLSTM VS Skip-thought + MaLSTM Retrieval Results

	MaLSTM		**Skip-thought + MaLSTM**	
Level	**Cases**	**Percentage**	**Cases**	**Percentage**
One:Three	8354	83.54%	8832	88.32%
Four:Seven	1510	15.1%	942	9.42%
Eight:Ten	30	0.3%	215	2.15%
Not Listed	106	1.06%	11	0.11%

Results Discussion and Lessons Learned The results show that **skip-thought** autoencoder helped in improving the overall accuracy of **DeepTMS** in finding the best solutions for the tickets in the top ten and decreasing the number of **Not Listed** solutions. The domain experts confirmed that if the desktop agent is able to find a solution within the first ten recommended solutions, that should be acceptable. However, the major advantage of using autoencoder models over LSTM models to denoise the data is the reduction in training effort. The LSTM model required a training set built on the full text, with both the necessary parts and the unnecessary ones in order to find the best representation for the email text. On the other hand, the AE training did not require such an effort since it was directly trained over all the text we had in a self-supervised learning process. AE helped speeding up the overall model training process of and improved the overall retrieval time as well.

6.3 DeepTMS Experiment - Conclusion

DeepTMS introduced a hybrid approach for Textual CBR system based on **DeepKAF** framework. **DeepTMS** uses DNNs to assist in automatic feature extractions from text and define similarity across text. DeepTMS is able to automate the building of text similarity without exhaustive expert involvement and work in real-time on new tickets to suggest the most relevant solutions. However, such an approach hides part of the explainability capability of CBR approaches. The main goal from that project was to show how a hybrid approach using deep learning and CBR can ease in dealing with complex tasks. Such an approach seems appropriate to deal with high volume data that need to be processed fast and in real-time.As explained in the previous parts, **DeepTMS** implementation was an ongoing project to achieve the best accuracy while minimizing the Domain Experts' effort and the time required to prepare the training datasets. Basic word embeddings and LSTM were applied at the first evaluation, and results were very promising to continue in the same direction. The second evaluation was a comparison between various word embedding models (Word2vec and fastText) and the results showed that Word2vec outperformed fastText in finding out how two sentences are similar. The third evaluation focused on improving the similarity measures, using a more advanced architecture like Siamese networks. Based on the Siamese networks architecture literature, **MaLSTM** has been found among the best models that can achieve high accuracy in terms of sentences similarities compared to other architectures. The overall system performance has greatly improved following the introduction of MaLSTM. The effort needed to create and train a Siamese network, however, was tedious as there was a need to prepare a dataset with sentence pairs to determine whether they are identical or not, which led to DeepTMS' fourth evaluation. In the fourth evaluation, autoencoders were added in the pre-processing stage to find an optimized text representation in a self-learning approach. Combining a Siamese network with an autoencoder and Word2vec showed the best results comparing to any other tested approaches.

To conclude, the presented work in this evaluation showed that an unsupervised skip-thought autoencoder has been combined with **MaLSTM** on large scale unstructured text and has seemed to improve the overall framework performance in terms of less training time, faster retrieval and more accurate results. Siamese networks and autoencoders with word embeddings are able to help TCBR systems by building stronger relationships across cases and measure similarities with minimal input from domain experts.

6.4 Experiment 2 - Finding Similarities - CaseLaw Access System

The CaseLaw Access system implemented in experiment 2 is based on the best combination of architectures that proved high accuracy in Experiment 1. The main goal of experiment 2 is to show the efficiency of the **DeepKAF** architecture and its genericity over different use cases and challenges. In the CaseLaw Dataset there is a very long description of each case which will be a test for autoencoders concept in data dimensionality reduction and case representation. **Figure 6.8** shows a sample case description from the CaseLaw Dataset.

6.4.1 The Dataset

Parts from the United States (US) CaseLaw dataset *Caselaw Access Project* 2018 were used. The dataset is following 360 years of US case-law. Its **C**aselaw **A**ccess **P**roject (CAP) API and bulk data services include 40 million pages of U.S. court decisions and almost 6.5 million unique cases. The CAP team has created metadata for each volume, including unique barcodes, reporter names, titles, jurisdictions, publication dates and other volume-level information. Key metadata fields, like case name, citation, court and decision date, were amended for accuracy, while the text of each case was left as a raw Optical Character Recognition (OCR) output. The dataset was regarded as a great application domain for

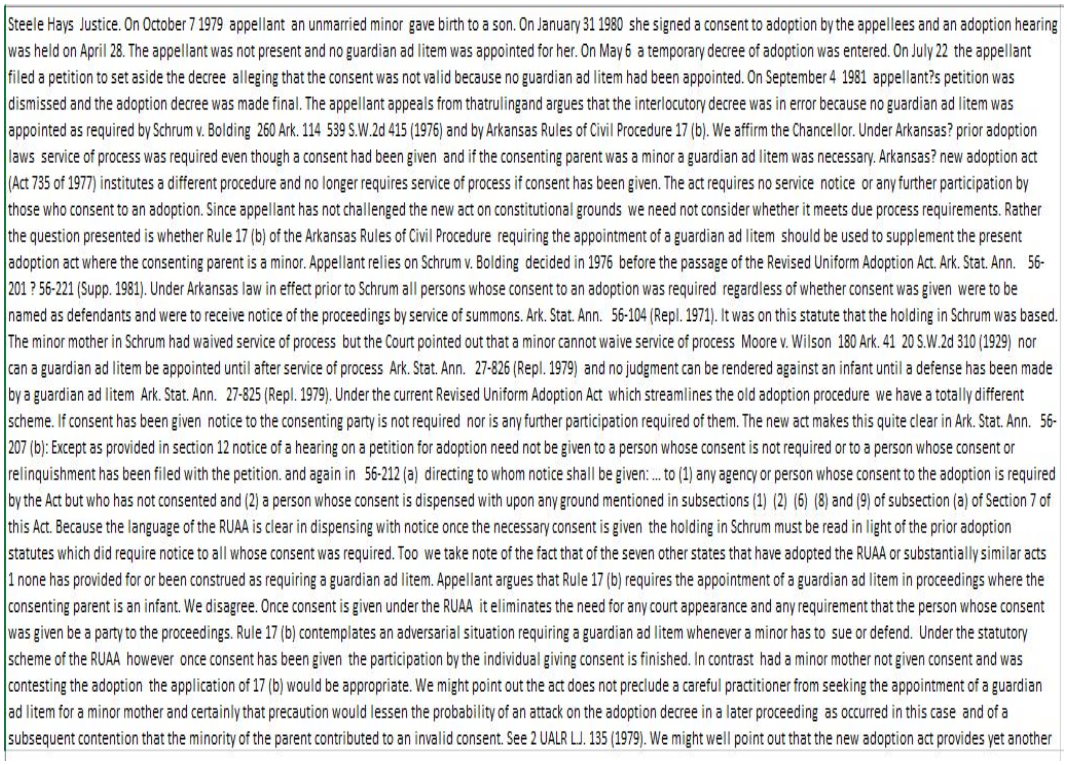
Steele Hays Justice. On October 7 1979 appellant an unmarried minor gave birth to a son. On January 31 1980 she signed a consent to adoption by the appellees and an adoption hearing was held on April 28. The appellant was not present and no guardian ad litem was appointed for her. On May 6 a temporary decree of adoption was entered. On July 22 the appellant filed a petition to set aside the decree alleging that the consent was not valid because no guardian ad litem had been appointed. On September 4 1981 appellant?s petition was dismissed and the adoption decree was made final. The appellant appeals from thatrulingand argues that the interlocutory decree was in error because no guardian ad litem was appointed as required by Schrum v. Bolding 260 Ark. 114 539 S.W.2d 415 (1976) and by Arkansas Rules of Civil Procedure 17 (b). We affirm the Chancellor. Under Arkansas? prior adoption laws service of process was required even though a consent had been given and if the consenting parent was a minor a guardian ad litem was necessary. Arkansas? new adoption act (Act 735 of 1977) institutes a different procedure and no longer requires service of process if consent has been given. The act requires no service notice or any further participation by those who consent to an adoption. Since appellant has not challenged the new act on constitutional grounds we need not consider whether it meets due process requirements. Rather the question presented is whether Rule 17 (b) of the Arkansas Rules of Civil Procedure requiring the appointment of a guardian ad litem should be used to supplement the present adoption act where the consenting parent is a minor. Appellant relies on Schrum v. Bolding decided in 1976 before the passage of the Revised Uniform Adoption Act. Ark. Stat. Ann. 56-201 ? 56-221 (Supp. 1981). Under Arkansas law in effect prior to Schrum all persons whose consent to an adoption was required regardless of whether consent was given were to be named as defendants and were to receive notice of the proceedings by service of summons. Ark. Stat. Ann. 56-104 (Repl. 1971). It was on this statute that the holding in Schrum was based. The minor mother in Schrum had waived service of process but the Court pointed out that a minor cannot waive service of process Moore v. Wilson 180 Ark. 41 20 S.W.2d 310 (1929) nor can a guardian ad litem be appointed until after service of process Ark. Stat. Ann. 27-826 (Repl. 1979) and no judgment can be rendered against an infant until a defense has been made by a guardian ad litem Ark. Stat. Ann. 27-825 (Repl. 1979). Under the current Revised Uniform Adoption Act which streamlines the old adoption procedure we have a totally different scheme. If consent has been given notice to the consenting party is not required nor is any further participation required of them. The new act makes this quite clear in Ark. Stat. Ann. 56-207 (b): Except as provided in section 12 notice of a hearing on a petition for adoption need not be given to a person whose consent is not required or to a person whose consent or relinquishment has been filed with the petition. and again in 56-212 (a) directing to whom notice shall be given: ... to (1) any agency or person whose consent to the adoption is required by the Act but who has not consented and (2) a person whose consent is dispensed with upon any ground mentioned in subsections (1) (2) (6) (8) and (9) of subsection (a) of Section 7 of this Act. Because the language of the RUAA is clear in dispensing with notice once the necessary consent is given the holding in Schrum must be read in light of the prior adoption statutes which did require notice to all whose consent was required. Too we take note of the fact that of the seven other states that have adopted the RUAA or substantially similar acts 1 none has provided for or been construed as requiring a guardian ad litem. Appellant argues that Rule 17 (b) requires the appointment of a guardian ad litem in proceedings where the consenting parent is an infant. We disagree. Once consent is given under the RUAA it eliminates the need for any court appearance and any requirement that the person whose consent was given be a party to the proceedings. Rule 17 (b) contemplates an adversarial situation requiring a guardian ad litem whenever a minor has to sue or defend. Under the statutory scheme of the RUAA however once consent has been given the participation by the individual giving consent is finished. In contrast had a minor mother not given consent and was contesting the adoption the application of 17 (b) would be appropriate. We might point out the act does not preclude a careful practitioner from seeking the appointment of a guardian ad litem for a minor mother and certainly that precaution would lessen the probability of an attack on the adoption decree in a later proceeding as occurred in this case and of a subsequent contention that the minority of the parent contributed to an invalid consent. See 2 UALR LJ. 135 (1979). We might well point out that the new adoption act provides yet another

Figure 6.8 Sample Case Body content

CBR, both for our evaluation and for possible future research.

6.4.2 Training

Due to the very high volume of data in this dataset (over 40 million pages) and the limitations of our infrastructure, we used 30,000 law cases split into a training (20,000 cases) and testing (10,000 cases) dataset. The training dataset contained a total of 36,831,400 Words and 2,643,000 Sentences. We used a combined architecture of **skip-thought and Doc2vec** (an instance from Word2vec) and **MaLSTM** in order to automate the process of building similarities. Each case was represented with the following attributes:- a. case body b. reporter full name c. court name d. majority. For the purpose of model evaluation, we simulated a dataset of 5000 of low-dimensional cases generated by the **skip-thought** autoencoder (2500 similar cases and 2500 not similar cases). This dataset was then used to train the **MaLSTM** to classify whether certain investigated cases were similar or not.

Hyper-parameters for the models were adjusted to fit to the new data. All hyper-parameters were found based on consequent experimentation runs. The training was stopped when the model seemed to converge to acceptable results. The used hyper-parameters are shown in **Table 6.7**:

Table 6.7 Models hyper-parameters

Skip-thought Autoencoder	**Doc2Vec**	**MaLSTM**
Embedding Dimension = 400 **Latent Dimension** = 400 **Batch Size** = 64 **Epochs**= 44 **Optimizer** = Nadam **Learning Rate** = 0.0001 **Decoder Activation Function** = SOFTMAX	**Word Vectors Dimensionality** = 300 **Window** = 10 **Min Word Frequency** = 50 **Epochs** = 53 **Learning Rate** = 0.0001 **Activation Function** = SOFTMAX	**Neuron Activation Function** = Relu **Learning Rate** = 0.0001 **Regularization** = L2 (.001) **LSTM Layer Size** = 400 **Batch Size** = 64 **Epochs** = 22 **Output Layer Loss Function** = MCXENT **Activation Function** = SOFTMAX

6.4.3 Experiment Results

The accuracy and F1 Score of the overall approach in finding if two cases are similar or not is shown in **Table 6.8**

Table 6.8 CaseLaw Experiment Results

	Skip-thought and MaLSTM
Accuracy	87.23%
F1	87.44%

Results Discussion and Lessons Learnt From the results, the combination of deep learning models that have been used showed high accuracy in the retrieval process. With a self-supervised learning approach, autoencoders helped reduce the text dimensionality, which helped a lot with training the Siamese Network on an optimized text rather than the long.

Experiment 2, proved that **DeepKAF** can handle different dataset characteristics and the models involved within the CBR paradigm are able to semi-automate the building of similarity measures.

6.5 Conclusion

As the latest version of **DeepKAF**, the **skip-thought** autoencoder architecture has replaced the **DeepKAF** LSTM model which was responsible for the text pre-processing and showed better performance in text denoising and low-dimensional representation with less training effort compared to the LSTM model. The approach of building a word embeddings model (Word2Vec) using the output of the autoencoder and use this model within a Siamese network

(**MaLSTM**) gives the best results in automating the process of finding textual similarity measures with minimum effort from both domain experts and knowledge engineers and preparing training datasets.

We have demonstrated different deep learning models that are being used and experimented that they give the best results (Word2vec and skip-thought and MaLSTM). Based on the above results, these models can be specialized further by increasing the number of layers and their nodes per layer to be able to learn more complex codings. However, this approach has to be used with caution since, for instance, the autoencoder may tend to simply copy its inputs to the output, without learning any internal text representation.

To conclude, although **DeepKAF** is predominantly developed and experimented with textual data, **DeepKAF** can be applied from a theoretical point of view to more complex, heterogeneous data sources, such as mixed textual and image data. With **DeepKAF**, CBR systems will hit new grounds where data is unstructured and heterogeneous when it comes to industrial implementations.

6.6 Summary

Chapter 6 discussed the validity of **DeepKAF** approach. **DeepKAF** has been implemented and tested on two distinctive use cases to prove its genericity. The two use cases were selected specially to test two main metrics: 1. Scalability, 2. Genericity. The ticket management system (DeepTMS) is a real domain application where CBR as an approach fits the most. However, the implementation faced many in terms of domain knowledge decoding and building similarity measures. **DeepKAF** as a hybrid-CBR approach showed excellent results in the ability to understand the domain jargon without extensive interference from the domain experts. The second use case was on the law cases, where cases were described in a very long text with millions of cases. **DeepKAF** has been able to process the long text and build similarity measures from millions of cases. At the end, **DeepKAF** was able to

tell which cases are similar to each other and hence should get almost the same verdict.

Chapter Seven

Conclusion & Future Work

7.1 Conclusion

CBR has a competitive advantage over the other ML techniques, namely being "explainable by nature". That gives CBR confidence in any crucial domains where explainability is a must like healthcare, aerospace, and mortgage approvals.

This thesis proposes the use of deep learning approaches within the CBR paradigm to overcome the CBR challenges in building similarity measures and retrieval systems in order to answer the main research questions posed in **Chapter 1** (Section 1.6). By combining CBR, big data, and deep learning approaches, the benefits of the three techniques have been brought together to solve complex problems in a real industrial domain. Based on the related work shown in this thesis, despite the substantial benefits that a CBR implementation can bring to an organization, CBR has been always suffering from being unable to be utilized easily in an industrial market since its both its implementation and maintenance are long and exhausting processes. On the other hand, deep learning approaches have proven their efficiency in solving several problems but lacking the explainability part which is the main benefit of using CBR. Putting CBR and deep learning together, CBR can hide the deep learning explainability problem, while deep learning can hide the need to exert too much effort to decode a domain knowledge.

This chapter presents the conducted research work in pursuit of the answers to the questions stated at the introduction, the contributions of this thesis as well as the arisen questions for possible future research work.

The literature survey has shown the state-of-the-art in the related disciplines of hybrid CBR approaches and highlighted the challenges that any CBR system can face. However, a good practice and unified approach towards combining CBR and deep learning in the textual CBR domain does not explicitly exist. Therefore, a concrete qualitative and then quantitative research approaches were formulated (research methods) in order to address the given problem.

Although, this research can be been applied with Textual CBR but can be used with different CBR applications in the industrial domain where the available data amounts are too large and complicated to be manually decoded by domain experts (e.g., videos, images or sound files). This can be done by eliminating the obstacles that come with processing textual data written with wrong grammar or mixed languages, improving the insights of the similarity calculations using unsupervised learning approaches, as well as providing an accurate retrieval of cases based on available past solutions to similar cases. In order to achieve this, different experiments with different approaches have been applied in order to come up with the good practice of how deep learning can effectively work with CBR.

Experiment results and models accuracy have shown how **DeepKAF** can significantly help towards the improving the presence of CBR in the industrial domain, namely by tackling real industrial problems and deriving a standard methodology of how to approach them using the **DeepKAF** approach for successful automatic decoding of domain knowledge and building similarity measures.

As set out in the introduction, the principal research question was whether CBR can deal with the huge amounts of unstructured data in the industrial domain and use deep learning approaches to assist effectively in automatically decoding the domain knowledge with minimum efforts from the experts?.

Since decoding domain knowledge processes are intensively engaging the domain experts, the unavailability of the domain experts and the costs of involving them into any activities were taken into account. As described throughout the introduction, the main aims of this research were:

1. To understand what is a suitable representation of unstructured textual cases
2. To understand how similarity measures among mixed-language text can be built
3. To define how the explanation of a suggested solution can be presented to users along with any related insights
4. To investigate appropriate technical architecture and tools to support a CBR system to effectively deal with millions of cases and being able to find similar cases and provide solutions in real-time

Towards the investigation of the above aims, a Hadoop eco-system along with Elastic-Search, Kafka, Keras and a workflow orchestrator Apache Oozie were chosen to build the framework that has been used in the research overall. The selected tools presented characteristics of modern frameworks that can handle huge amounts of data in real-time.

As seen from the exhaustive literature research, the data is getting more complex in terms of sources and sizes. Domain experts are not available all the time to provide help in how to understand the data.

7.2 Future Work

The work presented in this thesis was a leap towards the future of CBR. Besides the thesis contributions stated above, the future research paths that it motivates can also be regarded as a real contribution. The main future improvements are summarized below, including a brief discussion upon them.

1. A more in-depth investigation of the knowledge interoperability that the developed architecture offers. More combinations of deep learning approaches should be applied and tested against different datasets with different approaches.

2. A deep learning model can be built per case attribute to measure the local similarities, then use the outputs as input for another deep learning model to measure global similarities.

3. Investigate how the knowledge acquisition and decoding can be exploited efficiently and be of benefit to systems of a larger scale, or systems that may have been recently deployed and might not contain a robust past knowledge representation because of any challenges mentioned in this thesis.

4. Taking a step ahead but based on the architecture proposed, further work can look into the challenge of hiding some explainability by using different deep learning models for different tasks and combining contextual information to enhance the explainability process.

5. Finally, the provenance of any used cases and their associated solutions should be investigated further, identifying up to which extent they could assist and augment the explanation process.

As for this thesis, the main focus was automatic decoding of domain knowledge and building similarity measures, the future research should focus on expanding the usage of using deep learning models for other purposes, like improving the retrieval results, generate adaptation rules, generate new cases based on the collected knowledge.

To put it in **Alan Turing's** words: "We can only see a short distance ahead, but we can see plenty there that needs to be done."

REFERENCES

Aamodt, Agnar and Enric Plaza (1994). "Case-based reasoning: Foundational issues, methodological variations, and system approaches". In:

Abeßer, Jakob et al. (2017). "Acoustic Scene Classification by Combining Autoencoder-Based Dimensionality Reduction and Convolutional Neural Networks". In:

Acorn, Timothy L. and Sherry H. Walden (1992). "Smart: Support: Management Automated Reasoning Technology for Compaq Customer Service". In: *Proceedings of the The Fourth Conference on Innovative Applications of Artificial Intelligence (IAAI-92), July 12-16, 1992, San Jose, CA, USA*. Ed. by A. Carlisle Scott and Philip Klahr. AAAI, pp. 3–18. URL: http://www.aaai.org/Library/IAAI/1992/iaai92-003.php.

Adedoyin, A. et al. (2016). "Evaluating Case-Based Reasoning Knowledge Discovery in Fraud Detection". In: *ICCBR Workshops*.

Agorgianitis, Ioannis, Stelios Kapetanakis, et al. (2017). "Business Process Workflow Monitoring Using Distributed CBR with GPU Computing". English. In: Thirtieth International FLAIRS Conference , FLAIRS-30 ; Conference date: 22-05-2017 Through 24-05-2017, pp. 495–498.

Agorgianitis, Ioannis, Miltos Petridis, et al. (2016). "Evaluating Distributed Methods for CBR Systems for Monitoring Business Process Workflows". In: *ICCBR Workshops*.

Aha, David W., Len Breslow, and Héctor Muñoz-Avila (2001). "Conversational Case-Based Reasoning". In: *Appl. Intell.* 14.1, pp. 9–32. DOI: 10.1023/A:1008346807097. URL: https://doi.org/10.1023/A:1008346807097.

Aha, David W., David McSherry, and Qiang Yang (2005). "Advances in conversational case-based reasoning". In: *Knowledge Eng. Review* 20.3, pp. 247–254. DOI: 10.1017/S0269888906000531. URL: https://doi.org/10.1017/S0269888906000531.

Aleven, V. (1998). "Teaching Case-Based Argumentation through a Model and Examples". In:

Alippi, Cesare and Manuel Roveri (2008). "Just-in-Time Adaptive Classifiers—Part I: Detecting Nonstationary Changes". In: *Neural Networks, IEEE Transactions on* 19, pp. 1145–1153. DOI: 10.1109/TNN.2008.2000082.

Alippi, Cesare and Manuel Roveri (2009). "Just-in-Time Adaptive Classifiers—Part II: Designing the Classifier". In: *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* 19, pp. 2053–64. DOI: 10.1109/TNN.2008.2003998.

Almotiri, Jasem, Khaled Elleithy, and Abdelrahman Elleithy (2017). "Comparison of autoencoder and Principal Component Analysis followed by neural network for e-learning using handwritten recognition". In: pp. 1–5. DOI: 10.1109/LISAT.2017.8001963.

Alshammari, Gharbi et al. (2017). "A Hybrid CBR Approach for the Long Tail Problem in Recommender Systems". In: *Case-Based Reasoning Research and Development*. Ed. by David W. Aha and Jean Lieber. Cham: Springer International Publishing, pp. 35–45. ISBN: 978-3-319-61030-6.

Altszyler, Edgar, Mariano Sigman, and Diego Fernández Slezak (2016). "Comparative study of LSA vs Word2vec embeddings in small corpora: a case study in dreams database". In: *CoRR* abs/1610.01520. arXiv: 1610.01520. URL: http://arxiv.org/abs/1610.01520.

Amin, Kareem et al. (2018a). "Answering with Cases: A CBR Approach to Deep Learning: 26th International Conference, ICCBR 2018, Stockholm, Sweden, July 9-12, 2018, Proceedings". In: pp. 15–27. ISBN: 978-3-030-01080-5. DOI: 10.1007/978-3-030-01081-2_2.

Amin, Kareem et al. (2018b). "Case-based Reasoning in Natural Language Processing : Word 2 vec VS fastText". In:

Amin, Kareem et al. (2018c). "Dynamic Process Workflow Routing Using Deep Learning". In: *Artificial Intelligence XXXV*. Ed. by Max Bramer and Miltos Petridis. Cham: Springer International Publishing, pp. 132–142. ISBN: 978-3-030-04191-5.

— (2019). "Cases without Borders: Automating Knowledge Acquisition Approach using Deep Autoencoders and Siamese Networks in Case-Based Reasoning". In: *31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019, Portland, OR, USA, November 4-6, 2019*. IEEE, pp. 133–140. DOI: 10.1109/ICTAI.2019.00027. URL: https://doi.org/10.1109/ICTAI.2019.00027.

Amin, Kareem et al. (2020). "Advanced Similarity Measures Using Word Embeddings and Siamese Networks in CBR". In: *Intelligent Systems and Applications*. Ed. by Yaxin Bi, Rahul Bhatia, and Supriya Kapoor. Cham: Springer International Publishing, pp. 449–462. ISBN: 978-3-030-29513-4.

Arditi, David and Onur Behzat Tokdemir (1999). "Comparison of Case-Based Reasoning and Artificial Neural Networks". In: *Journal of Computing in Civil Engineering* 13.3, pp. 162–169. DOI: 10.1061/(ASCE)0887-3801(1999)13:3(162). eprint: https://ascelibrary.org/doi/pdf/10.1061/\%28ASCE\%290887-3801\%281999\%2913\%3A3\%28162\%29. URL: https://ascelibrary.org/doi/abs/10.1061/%5C%28ASCE%5C%290887-3801%5C%281999%5C%2913%5C%3A3%5C%28162%5C%29.

Arrieta Barredo, Alejandro et al. (2020). "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information Fusion* 58, pp. 82–115. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2019.12.012. URL: http://dx.doi.org/10.1016/j.inffus.2019.12.012.

Ashley, Kevin D. (1991). "Reasoning with Cases and Hypotheticals in HYPO". In: *International Journal of Man-Machine Studies* 34.6, pp. 753–796. DOI: 10.1016/0020-7373(91)90011-U. URL: https://doi.org/10.1016/0020-7373(91)90011-U.

Ashley, Kevin D. and Edwina L. Rissland (1987). "Compare and Contrast: A Test of Expertise". In: *AAAI*.

Bach, Kerstin and Klaus-Dieter Althoff (2012a). "Developing Case-Based Reasoning Applications Using myCBR 3". In: vol. 7466, pp. 17–31. DOI: 10.1007/978-3-642-32986-9_4.

— (2012b). "Developing Case-Based Reasoning Applications Using myCBR 3". In: vol. 7466, pp. 17–31. DOI: 10.1007/978-3-642-32986-9_4.

Bach, Kerstin, Klaus-Dieter Althoff, et al. (2011). "A Case-Based Reasoning Approach for Providing Machine Diagnosis from Service Reports". In: *Case-Based Reasoning Research and Development - 19th International Conference on Case-Based Reasoning, ICCBR 2011, London, UK, September 12-15, 2011. Proceedings*. Ed. by Ashwin Ram and Nirmalie Wiratunga. Vol. 6880. Lecture Notes in Computer Science. Springer, pp. 363–377. DOI: 10.1007/978-3-642-23291-6_27. URL: https://doi.org/10.1007/978-3-642-23291-6%5C_27.

Bach, Kerstin, Odd Erik Gundersen, et al. (2014). "Automatic Case Capturing for Problematic Drilling Situations". In: *Case-Based Reasoning Research and Development. ICCBR 2014*. Springer, pp. 48–62. ISBN: 978-3-319-11208-4. DOI: 10.1007/978-3-319-11209-1_5.

Bach, Kerstin, Meike Reichle, and Klaus-Dieter Althoff (2007). "A Domain Independent System Architecture for Sharing Experience". In: pp. 296–303.

Bach, Kerstin, Meike Reichle, Alexander Reichle-Schmehl, et al. (2008). "Implementing a Coordination Agent for Modularised Case Bases". In: pp. 1–12.

Bahls, Daniel and Thomas Roth-Berghofer (2007). "Explanation Support for the Case-Based Reasoning Tool myCBR". In: pp. 1844–1845.

El-Bahnasy, Khaled, Kareem Amin, and Mostafa Aref (2014). "A Novel Case base Indexing Model based on Power Set Tree". In: *International Journal of Computer Applications* 97, pp. 1–8. DOI: 10.5120/17016-7298.

Ballard, Dana H. (1987). "Modular Learning in Neural Networks". In: *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*. AAAI'87. Seattle, Washington: AAAI Press, pp. 279–284. ISBN: 0934613427.

Beaver, Ian and Joe Dumoulin (2013). "Applying MapReduce to Learning User Preferences in Near Real-Time". In: *Case-Based Reasoning Research and Development*. Ed. by Sarah Jane Delany and Santiago Ontañón. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 15–28. ISBN: 978-3-642-39056-2.

Bedué, Patrick et al. (2015). "Enriching Cooking Workflows with Multimedia Data from a High Security Cloud Storage". In: *ICCBR*.

Bello-Tomás, Juan José, Pedro A. González-Calero, and Belén Díaz-Agudo (2004). "JColibri: An Object-Oriented Framework for Building CBR Systems". In: *Advances in Case-Based Reasoning*. Ed. by Peter Funk and Pedro A. González Calero. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 32–46. ISBN: 978-3-540-28631-8.

Bengio, Y., A. Courville, and P. Vincent (2013). "Representation Learning: A Review and New Perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8, pp. 1798–1828. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2013.50.

Bengio, Yoshua (2013). "Deep Learning of Representations: Looking Forward". In: *CoRR* abs/1305.0445. arXiv: 1305.0445. URL: http://arxiv.org/abs/1305.0445.

Bengio, Yoshua, Aaron C. Courville, and Pascal Vincent (2012). "Unsupervised Feature Learning and Deep Learning: A Review and New Perspectives". In: *CoRR* abs/1206.5538. arXiv: 1206.5538. URL: http://arxiv.org/abs/1206.5538.

Bentley, Jon Louis (1975). "Multidimensional Binary Search Trees Used for Associative Searching". In: *Commun. ACM* 18.9, pp. 509–517. ISSN: 0001-0782. DOI: 10.1145/361002.361007. URL: https://doi.org/10.1145/361002.361007.

Bergmann, Ralph (2002a). *Experience Management: Foundations, Development Methodology, and Internet-Based Applications*. Vol. 2432. Lecture Notes in Computer Science. Springer. ISBN: 3-540-44191-3. DOI: 10.1007/3-540-45759-3. URL: https://doi.org/10.1007/3-540-45759-3.

— (2002b). *Experience Management: Foundations, Development Methodology, and Internet-Based Applications*. Vol. 2432. ISBN: 3-540-44191-3. DOI: 10.1007/3-540-45759-3.

Bergmann, Ralph, Andrea Freßmann, et al. (2006). "Case-Based Support for Collaborative Business". In: pp. 519–533. DOI: 10.1007/11805816_38.

Bergmann, Ralph and Yolanda Gil (2014). "Similarity assessment and efficient retrieval of semantic workflows". In: *Information Systems* 40, pp. 115–127. DOI: 10.1016/j.is.2012.07.005.

Bergmann, Ralph, Lisa Grumbach, et al. (2019). "ProCAKE: A Process-Oriented Case-Based Reasoning Framework". In:

Bergmann, Ralph, Janet Kolodner, and Enric Plaza (2005). "Representation in case-based reasoning". In: *Knowledge Eng. Review* 20, pp. 209–213. DOI: 10.1017/S0269888906000555.

Blum, Avrim and Carl Burch (2000). "On-line Learning and the Metrical Task System Problem". In: *Machine Learning* 39. DOI: 10.1145/267460.267475.

Bottou, Léon (1999). "On-line Learning and Stochastic Approximations". In: *On-Line Learning in Neural Networks*. Ed. by DavidEditor Saad. Publications of the Newton Institute. Cambridge University Press, pp. 9–42. DOI: 10.1017/CBO9780511569920.003.

Bromley, Jane et al. (1993a). "Signature Verification using a "Siamese" Time Delay Neural Network". In: *International Journal of Pattern Recognition and Artificial Intelligence* 7, p. 25. DOI: 10.1142/S0218001493000339.

— (1993b). "Signature Verification using a "Siamese" Time Delay Neural Network". In: *International Journal of Pattern Recognition and Artificial Intelligence* 7, p. 25. DOI: 10.1142/S0218001493000339.

Brüninghaus, Stefanie and Kevin D. Ashley (1998). "How Machine Learning Can be Beneficial for Textual Case-Based Reasoning". In: 34.6, pp. 71–74.

— (2001). "The Role of Information Extraction for Textual CBR". In: *Case-Based Reasoning Research and Development*. Ed. by David W. Aha and Ian Watson. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 74–89. ISBN: 978-3-540-44593-7.

Brynjolfsson, Erik and Kristina McElheran (2016). "Data in Action: Data-Driven Decision Making in U.S. Manufacturing". In: *IO: Empirical Studies of Firms Markets eJournal*.

C. Schank, Roger (1986). "Explanation Patterns: Understanding Mechanically and Creatively". In:

Carbonell, Jaime G. (1983). "Learning by Analogy: Formulating and Generalizing Plans from Past Experience". In: *Machine Learning: An Artificial Intelligence Approach*. Ed. by Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 137–161. ISBN: 978-3-662-12405-5. DOI: 10.1007/978-3-662-12405-5_5. URL: https://doi.org/10.1007/978-3-662-12405-5_5.

Carey-Simos, George (2019). *How much data is generated every minute?* URL: http://wersm.com/how-much-data-is-generated-every-minute-on-social-media/ (visited on 08/30/2019).

Carletti, Vincenzo, Pasquale Foggia, and Mario Vento (2013). "Performance Comparison of Five Exact Graph Matching Algorithms on Biological Databases". In: *New Trends in Image Analysis and Processing – ICIAP 2013*. Ed. by Alfredo Petrosino, Lucia Maddalena, and Pietro Pala. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 409–417. ISBN: 978-3-642-41190-8.

Caselaw Access Project (2018). URL: https://case.law/.

Cesa-Bianchi, Nicolò et al. (1996). "On-line Prediction and Conversion Strategies". In: *Machine Learning* 25, pp. 71–110. DOI: 10.1023/A:1018348209754.

Chaudhuri, Surajit, Umeshwar Dayal, and Vivek Narasayya (2011). "An Overview of Business Intelligence Technology". In: *Commun. ACM* 54.8, pp. 88–98. ISSN: 0001-0782. DOI: 10.1145/1978542.1978562. URL: https://doi.org/10.1145/1978542.1978562.

Cheetham, William (2005). "Tenth Anniversary of the Plastics Color Formulation Tool." In: *AI Magazine* 26, pp. 51–62.

Cheetham, William and Kai Goebel (2007). "Appliance Call Center: A Successful Mixed-Initiative Case Study". In: *AI Magazine* 28.2, pp. 89–100. DOI: 10.1609/aimag.v28i2.2042. URL: https://doi.org/10.1609/aimag.v28i2.2042.

Chen, Lingran and Wolfgang Robien (1994). "Application of the Maximal Common Substructure Algorithm to Automatic Interpretation of 13C-NMR Spectra". In: *Journal of Chemical Information and Computer Sciences* 34.4, pp. 934–941. DOI: 10.1021/ci00020a030. eprint: https://pubs.acs.org/doi/pdf/10.1021/ci00020a030. URL: https://pubs.acs.org/doi/abs/10.1021/ci00020a030.

Chen, Min, Shiwen Mao, and Yunhao Liu (2014). "Big data: A survey". In: *Mobile Networks and Applications* 19. DOI: 10.1007/s11036-013-0489-0.

Chen, Shuangshuang et al. (2017). "Image Classification Based on Convolutional Denoising Sparse Autoencoder". In: *Mathematical Problems in Engineering* 2017, pp. 1–16. DOI: 10.1155/2017/5218247.

Chen, Wei-Chou et al. (2002). "A parallelized indexing method for large-scale case-based reasoning". In: *Expert Syst. Appl.* 23, pp. 95–102. DOI: 10.1016/S0957-4174(02)00029-5.

Chen, X. and X. Lin (2014). "Big Data Deep Learning: Challenges and Perspectives". In: *IEEE Access* 2, pp. 514–525. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2014.2325029.

Chen, Yanpei, Sara Alspaugh, and Randy Katz (2012). "Interactive Analytical Processing in Big Data Systems: A Cross-Industry Study of MapReduce Workloads". In: *Proc. VLDB Endow.* 5.12, pp. 1802–1813. ISSN: 2150-8097. DOI: 10.14778/2367502.2367519. URL: https://doi.org/10.14778/2367502.2367519.

Chien, Jen-Tzung and Hsin-Lung Hsieh (2013). "Nonstationary Source Separation Using Sequential and Variational Bayesian Learning". In: *Neural Networks and Learning Systems, IEEE Transactions on* 24, pp. 681–694. DOI: 10.1109/TNNLS.2013.2242090.

Chun-Guang, Chang et al. (2004). "Research on case adaptation techniques in case-based reasoning". In: *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*. Vol. 4, 2128–2133 vol.4.

Claesen, Marc and Bart De Moor (2015). "Hyperparameter Search in Machine Learning". In: *CoRR* abs/1502.02127. arXiv: 1502.02127. URL: http://arxiv.org/abs/1502.02127.

Coello, J. and R. D. C. Santos (1999). "Integrating CBR and Heuristic Search for Learning and Reusing Solutions in Real-Time Task Scheduling". In: *ICCBR*. Springer.

Cohnitz, Daniel (2002). "Explanations are like salted peanuts. Why you can't cut the route toward further reduction". In:

Collobert, Ronan and Jason Weston (2008). "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML 08. Helsinki, Finland: Association for Computing Machinery, pp. 160–167. ISBN: 9781605582054. DOI: 10.1145/1390156.1390177. URL: https://doi.org/10.1145/1390156.1390177.

Cooper, Brian et al. (2010). "Benchmarking cloud serving systems with YCSB". In: pp. 143–154. DOI: 10.1145/1807128.1807152.

Cordier, Amélie et al. (2014). "Taaable: A Case-Based System for Personalized Cooking". In: vol. 494. DOI: 10.1007/978-3-642-38736-4_7.

Craw, Susan (2010). "Case-Based Reasoning". In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, pp. 147–154. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_97. URL: https://doi.org/10.1007/978-0-387-30164-8_97.

Daengdej, J. et al. (1996). "Dynamically creating indices for two million cases: A real world problem". In: *Advances in Case-Based Reasoning*. Ed. by Ian Smith and Boi Faltings. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 105–119. ISBN: 978-3-540-49568-0.

Dai, Hanjun et al. (2017). "Sequence2Vec: A novel embedding approach for modeling transcription factor binding affinity landscape". In: *Bioinformatics (Oxford, England)* 33. DOI: 10.1093/bioinformatics/btx480.

De Loor, Pierre, Romain Bénard, and Pierre Chevaillier (2011). "Real-time retrieval for case-based reasoning in interactive multiagent-based simulations". In: *Expert Systems With Applications - ESWA* 38. DOI: 10.1016/j.eswa.2010.10.048.

Dean, Jeffrey et al. (2012). "Large Scale Distributed Deep Networks". In: *Advances in neural information processing systems*.

DeMiguel, Juan, Laura Plaza, and Belén Díaz-Agudo (2008). "ColibriCook: A CBR System for Ontology-Based Recipe Retrieval and Adaptation." In: pp. 199–208.

Desarkar, Anindita and Ajanta Das (2017). "Big-Data Analytics, Machine Learning Algorithms and Scalable/Parallel/Distributed Algorithms". In: pp. 159–197. ISBN: 978-3-319-49735-8. DOI: 10.1007/978-3-319-49736-5_8.

Diaz-Agudo, Belén et al. (2007). "Building CBR systems with jcolibri". In: *Science of Computer Programming* 69, pp. 68–75. DOI: 10.1016/j.scico.2007.02.004.

Dieterle, Sebastian and Ralph Bergmann (2014). "A Hybrid CBR-ANN Approach to the Appraisal of Internet Domain Names". In: *Case-Based Reasoning Research and Develop-*

ment. Ed. by Luc Lamontagne and Enric Plaza. Cham: Springer International Publishing, pp. 95–109. ISBN: 978-3-319-11209-1.

Dong, Xingping and Jianbing Shen (2018). "Triplet Loss in Siamese Network for Object Tracking". In: *The European Conference on Computer Vision (ECCV)*.

Elwell, Ryan and Robi Polikar (2009). "Incremental learning in nonstationary environments with controlled forgetting". In: pp. 771–778. DOI: 10.1109/IJCNN.2009.5178779.

— (2011). "Incremental Learning of Concept Drift in Nonstationary Environments". In: *Neural Networks, IEEE Transactions on* 22, pp. 1517–1531. DOI: 10.1109/TNN.2011.2160459.

Fan, Heng and Haibin Ling (2018). "Siamese Cascaded Region Proposal Networks for Real-Time Visual Tracking". In: *CoRR* abs/1812.06148. arXiv: 1812.06148. URL: http://arxiv.org/abs/1812.06148.

Finnie, Gavin and Zhaohao Sun (2002). "Similarity and metrics in case-based reasoning". In: *Information Technology papers* 17. DOI: 10.1002/int.10021.abs.

Floyd, Michael, Alan Davoust, and Babak Esfandiari (2008). "Considerations for Real-Time Spatially-Aware Case-Based Reasoning: A Case Study in Robotic Soccer Imitation". In: *ECCBR 2008*. Vol. 5239, pp. 195–209. DOI: 10.1007/978-3-540-85502-6_13.

Floyd, Michael and Babak Esfandiari (2011). "Learning State-Based Behaviour using Temporally Related Cases". In: 829.

Forcher, Björn et al. (2014). "Intuitive justifications of medical semantic search results". In: *Engineering Applications of Artificial Intelligence* 30, pp. 1–17. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2014.01.013. URL: http://www.sciencedirect.com/science/article/pii/S0952197614000207.

Freund, Yoav and Robert Schapire (2000). "Game Theory, On-line Prediction and Boosting". In: *COLT*. DOI: 10.1145/238061.238163.

Fu, Zhenxin et al. (2018). "Style Transfer in Text: Exploration and Evaluation". In: *AAAI*.

Gedikli, Fatih, Dietmar Jannach, and Mouzhi Ge (2014). "How should I explain? A comparison of different explanation types for recommender systems". In: *International Journal of Human-Computer Studies* 72.4, pp. 367–382. ISSN: 1071-5819. DOI: https://doi.org/10.1016/j.ijhcs.2013.12.007. URL: http://www.sciencedirect.com/science/article/pii/S1071581913002024.

Géron, Aurélien (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media, Inc. ISBN: 9781491962282.

Goel, Ashok Kumar (1989). "Integration of Case-Based Reasoning and Model-Based Reasoning for Adaptive Design Problem-Solving". AAI9014426. PhD thesis.

Goel, Ashok and Belen Diaz-Agudo (2017). *What's Hot in Case-Based Reasoning*.

Göker, Mehmet et al. (1998). "The development of HOMER a case-based CAD/CAM help-desk support tool". In: *Advances in Case-Based Reasoning*. Ed. by Barry Smyth and Pádraig Cunningham. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 346–357. ISBN: 978-3-540-49797-4.

Goldberg, Yoav (2017). "Neural Network Methods for Natural Language Processing". In: *Synthesis Lectures on Human Language Technologies* 10.1, pp. 1–309. DOI: 10.2200/S00762ED1V01Y201703HLT037. URL: https://doi.org/10.2200/S00762ED1V01Y201703HLT037.

Google Cloud Platform (2018). URL: https://cloud.google.com/.

Gundersen, Odd Erik et al. (2013). "A Real-Time Decision Support System for High Cost Oil-Well Drilling Operations". In: *AI Magazine* 34. DOI: 10.1609/aimag.v34i1.2434.

Guo, Q. et al. (2017). "Learning Dynamic Siamese Network for Visual Object Tracking". In: *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1781–1789.

Hammer, Barbara and Thomas Villmann (2007). "How to process uncertainty in machine learning?" In: pp. 79–90.

Hammond, Kristian J. (1986). "CHEF: A Model of Case-Based Planning". In: *Proceedings of the 5th National Conference on Artificial Intelligence. Philadelphia, PA, USA, August 11-15, 1986. Volume 1: Science*. Ed. by Tom Kehler. Morgan Kaufmann, pp. 267–271. URL: http://www.aaai.org/Library/AAAI/1986/aaai86-044.php.

— (1989). *Case-Based Planning: Viewing Planning as a Memory Task*. USA: Academic Press Professional, Inc. ISBN: 0123220602.

— (1990). "Explaining and Repairing Plans That Fail". In: *Artif. Intell.* 45.1-2, pp. 173–228. DOI: 10.1016/0004-3702(90)90040-7. URL: https://doi.org/10.1016/0004-3702(90)90040-7.

Han, Mee Lan et al. (2016). "WHAP: Web-hacking profiling using Case-Based Reasoning". In: *2016 IEEE Conference on Communications and Network Security (CNS)*, pp. 344–345.

Han, Rui et al. (2015). "Benchmarking Big Data Systems: State-of-the-Art and Future Directions". In:

Hanney, Kathleen and Mark T. Keane (1997). "The adaptation knowledge bottleneck: How to ease it by learning from cases". In: *Case-Based Reasoning Research and Development*. Ed. by David B. Leake and Enric Plaza. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 359–370. ISBN: 978-3-540-69238-6.

He, Anfeng et al. (2018). "A Twofold Siamese Network for Real-Time Object Tracking". In: *CoRR* abs/1802.08817. arXiv: 1802.08817. URL: http://arxiv.org/abs/1802.08817.

Herlocker, Jon, Joseph Konstan, and John Riedl (2001). "Explaining Collaborative Filtering Recommendations". In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. DOI: 10.1145/358916.358995.

Hinkle, David and Christopher Toomey (1994). "Clavier: Applying Case-Based Reasoning to Composite Part Fabrication". In: *Proceedings of the Sixth Annual Conference on Innovative Applications of Artificial Intelligence, IAAI 1994, Seattle, Washington, USA, August 1-4, 1994*. Ed. by Elizabeth Byrnes and Jan Aikins. AAAI Press. URL: http://www.aaai.org/Library/IAAI/1994/iaai94-007.php.

Hinrichs, Thomas R. (1988). "Towards an architecture for open world problem solving". In: *Proceedings of the 1st case-based reasoning workshop*. Morgan Kaufmann. New York.

— (1989). "Strategies for Adaptation and Recovery in w a Design Problem Solver". In:

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Comput.* 9.8, pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

Hofstadter, D. R. (1985). *Analogies and roles in human and machine thinking.* New York: Basic Books: Metamagical Themas.

Hopfield, J.J. (2018). "Neural networks and physical systems with emergent collective computational abilities". In: pp. 7–19. DOI: 10.1201/9780429500459.

Huang, Shengsheng et al. (2010). "The HiBench Benchmark Suite: Characterization of the MapReduce-Based Data Analysis". In: pp. 41–51. DOI: 10.1109/ICDEW.2010.5452747.

Im, Kwang Hyuk and Sang Chan Park (2007). "Case-based reasoning and neural network based expert system for personalization". In: *Expert Systems with Applications* 32.1, pp. 77–85. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2005.11.020. URL: http://www.sciencedirect.com/science/article/pii/S095741740500309X.

Jære, Martha Dørum, Agnar Aamodt, and Pål Skalle (2002). "Representing Temporal Knowledge for Case-Based Prediction". In: *Advances in Case-Based Reasoning*. Ed. by Susan Craw and Alun Preece. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 174–188. ISBN: 978-3-540-46119-7.

Jai-Andaloussi, S. et al. (2013). "Medical content based image retrieval by using the Hadoop framework". In: *ICT 2013*, pp. 1–5. DOI: 10.1109/ICTEL.2013.6632112.

Jalali, Vahid and David Leake (2014). "Adaptation-Guided Case Base Maintenance". In: vol. 3.

— (2015). "CBR Meets Big Data: A Case Study of Large-Scale Adaptation Rule Generation". In: pp. 181–196. ISBN: 978-3-319-24585-0. DOI: 10.1007/978-3-319-24586-7_13.

Jalali, Vahid and David Leake (2017). "Scaling Up Ensemble of Adaptations for Classification by Approximate Nearest Neighbor Retrieval". In: *Case-Based Reasoning Research and Development*. Ed. by David W. Aha and Jean Lieber. Cham: Springer International Publishing, pp. 154–169.

Joulin, Armand et al. (2016). "Bag of Tricks for Efficient Text Classification". In: *CoRR* abs/1607.01759. arXiv: 1607.01759. URL: http://arxiv.org/abs/1607.01759.

Julián, Vicente and Juan Corchado Rodríguez (2012). "Temporal bounded reasoning in a dynamic case based planning agent for industrial environments". In: *Expert Systems with Applications* 39.

Kang, Yong-Bin, Shonali Krishnaswamy, and Arkady Zaslavsky (2011). "Retrieval in CBR Using a Combination of Similarity and Association Knowledge". In: *Advanced Data Mining and Applications*. Ed. by Jie Tang et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–14. ISBN: 978-3-642-25853-4.

Kapetanakis, Stelios, Miltiadis Petridis, Jixin Ma, and Liz Bacon (2009). "Workflow monitoring and diagnosis using case based reasoning on incomplete temporal log data". English. In: *Proceedings of the 8th International Conference on Case Based Reasoning*. Vol. Worksh.

— (2010). "Providing explanations for the intelligent monitoring of business workflows using case-based reasoning". English. In: *Proceedings of the Fifth International Workshop on Explanation-aware Computing, ExaCt 2010*. Vol. 650. CEUR Workshop Proceedings. Ceur Workshop Proceedings, pp. 25–36.

Kapetanakis, Stelios, Miltiadis Petridis, Jixin Ma, and Brian Knight (2010). "CBR-WIMS, an intelligent monitoring platform for business processes". English. In: *Proceedings of the 15th UK CBR workshop*. CMS Press, pp. 55–64.

Kapetanakis, Stelios and Miltos Petridis (2014). "Evaluating a Case-Based Reasoning Architecture for the Intelligent Monitoring of Business Workflows". In: *Studies in Computational Intelligence* 494, pp. 43–54. DOI: 10.1007/978-3-642-38736-4_4.

Kapetanakis, Stelios, Miltos Petridis, et al. (2010). "A Case Based Reasoning Approach for the Monitoring of Business Workflows". In: *Case-Based Reasoning. Research and Development*. Ed. by Isabelle Bichindaritz and Stefania Montani. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 390–405. ISBN: 978-3-642-14274-1.

Keane, Mark T. and Eoin M. Kenny (2019). "How Case Based Reasoning Explained Neural Networks: An XAI Survey of Post-Hoc Explanation-by-Example in ANN-CBR Twins". In: *CoRR* abs/1905.07186. arXiv: 1905.07186. URL: http://arxiv.org/abs/1905.07186.

Kersten, G.E. and D.B Meister (1996). "Qualitative representations of negotiations: Tutorial on a rule-based approach". In: Group Decis Negot 5. DOI: https://doi.org/10.1007/BF00419909.

Kim, Gang-Hoon, Silvana Trimi, and Ji-Hyong Chung (2014). "Big-Data Applications in the Government Sector". In: *Commun. ACM* 57.3, pp. 78–85. ISSN: 0001-0782. DOI: 10.1145/2500873. URL: https://doi.org/10.1145/2500873.

Kim, Yoon (2014). "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1746–1751. DOI: 10.3115/v1/D14-1181. URL: https://www.aclweb.org/anthology/D14-1181.

Kiros, Ryan et al. (2015). "Skip-Thought Vectors". In: *CoRR* abs/1506.06726. arXiv: 1506.06726. URL: http://arxiv.org/abs/1506.06726.

Labrinidis, Alexandros and H.V. Jagadish (2012a). "Challenges and opportunities with big data". In: *Proceedings of the VLDB Endowment* 5, pp. 2032–2033. DOI: 10.14778/2367502.2367572.

— (2012b). "Challenges and opportunities with big data". In: *Proceedings of the VLDB Endowment* 5, pp. 2032–2033. DOI: 10.14778/2367502.2367572.

Le, Quoc V. and Tomas Mikolov (2014a). "Distributed Representations of Sentences and Documents". In: *CoRR* abs/1405.4053. arXiv: 1405.4053. URL: http://arxiv.org/abs/1405.4053.

— (2014b). "Distributed Representations of Sentences and Documents". In: *CoRR* abs/1405.4053. arXiv: 1405.4053. URL: http://arxiv.org/abs/1405.4053.

Leake, D. B. (2002). "Artificial Intelligence (A brief non-technical introduction". In: Van Nostrand's Scientific Encyclopedia.

Lebowitz, Michael (1983). "Memory-Based Parsing". In: *Artif. Intell.* 21.4, pp. 363–404. DOI: 10.1016/S0004-3702(83)80019-8. URL: https://doi.org/10.1016/S0004-3702(83)80019-8.

Lecun, Yann (2001). "A Theoretical Framework for Back-Propagation". In:

Lees, Brian and Juan Corchado (1999). "Integrated Case-Based Neural Network Approach to Problem Solving". In: *XPS-99: Knowledge-Based Systems. Survey and Future Directions*. Ed. by Frank Puppe. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 157–166. ISBN: 978-3-540-49149-1.

Lewis, Richard A., Jonathan S. Mason, and Iain M. McLay (1997). "Similarity Measures for Rational Set Selection and Analysis of Combinatorial Libraries: The Diverse Property-Derived (DPD) Approach". In: *Journal of Chemical Information and Computer Sciences* 37.3. PMID: 9177003, pp. 599–614. DOI: 10.1021/ci960471y. eprint: https://doi.org/10.1021/ci960471y. URL: https://doi.org/10.1021/ci960471y.

Li, Bo et al. (2018). "SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks". In: *CoRR* abs/1812.11703. arXiv: 1812.11703. URL: http://arxiv.org/abs/1812.11703.

Li, B. et al. (2018). "High Performance Visual Tracking with Siamese Region Proposal Network". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8971–8980.

Li, H. et al. (2007). "Adaptation Rule Learning for Case-Based Reasoning". In: *Third International Conference on Semantics, Knowledge and Grid (SKG 2007)*, pp. 44–49.

Li, Oscar et al. (2017). "Deep Learning for Case-based Reasoning through Prototypes: A Neural Network that Explains its Predictions". In:

Li, Xiangang and Xihong Wu (2014). "Constructing Long Short-Term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition". In: *CoRR* abs/1410.4281. arXiv: 1410.4281. URL: http://arxiv.org/abs/1410.4281.

Likhachev, Maxim, Michael Kaess, and Ronald Arkin (2002). "Learning Behavioral Parameterization Using Spatio-Temporal Case-Based Reasoning". In: vol. 2, pp. 1282–1289. ISBN: 0-7803-7272-7. DOI: 10.1109/ROBOT.2002.1014719.

Littlestone, Nicholas, Manfred Warmuth, and Philip Long (1995). "On-line learning of linear functions". In: *Computational Complexity* 5, pp. 1–23. DOI: 10.1007/BF01277953.

Mabu, Shingo et al. (2018). "Unsupervised Image Classification Using Multi-Autoencoder and K-means++". In: *Journal of Robotics, Networking and Artificial Life* 5 (1), pp. 75–78. ISSN: 2352-6386. DOI: https://doi.org/10.2991/jrnal.2018.5.1.17. URL: https://doi.org/10.2991/jrnal.2018.5.1.17.

Manning, Christopher D. et al. (2014). "The Stanford CoreNLP Natural Language Processing Toolkit". In: *Association for Computational Linguistics (ACL) System Demonstrations*, pp. 55–60. URL: http://www.aclweb.org/anthology/P/P14/P14-5010.

Marcella, Rita and Iain Middleton (1996). "The role of the help desk in the strategic management of information systems". In: *OCLC Systems & Services: International digital library perspectives*.

Marling, C. et al. (2015). "Case-Based Reasoning as a Prelude to Big Data Analysis: A Case Study". In: *ICCBR*.

Martin, Kyle et al. (2017). "A Convolutional Siamese Network for Developing Similarity Knowledge in the SelfBACK Dataset". In: *ICCBR*.

McSherry, David (2002). "Mixed-Initiative Dialogue in Case-Based Reasoning". In: *6th European Conference ov Case Based Reasoning, ECCBR 2002, Aberdeen, Scotland, UK, September 4-7, 2002, Workshop Proceedings*, pp. 1–8.

Mehlhorn, Kurt (1984). *Data Structures and Algorithms 3: Multi-Dimensional Searching and Computational Geometry*. Berlin, Heidelberg: Springer-Verlag. ISBN: 0387136428.

Miao, Yishu, Lei Yu, and Phil Blunsom (2015). "Neural Variational Inference for Text Processing". In: *CoRR* abs/1511.06038. URL: http://arxiv.org/abs/1511.06038.

Mikolov, Tomas, Kai Chen, and Greg Corrado (2013). "Efficient Estimation of Word Representations in Vector Space". In:

Mille, Alain et al. (2013). "Trace-Based Reasoning - Modeling interaction traces for reasoning on experiences". In: *FLAIRS 2013 - Proceedings of the 26th International Florida Artificial Intelligence Research Society Conference*.

Mitchell, M. (1993). *Analogy-Making as Perception: A Computer Model*. Cambridge, Massachusetts: U.S.A: Bradford Books.

Mohd-Hassin, Mohd, Norita Md Norwawi, and Azizi Ab Aziz (2006). "Temporal Case-Based Reasoning for reservoir spillway gate operation recommendation". In: DOI: 10.1109/ICOCI.2006.5276517.

Mueller, Jonas and Aditya Thyagarajan (2016). "Siamese Recurrent Architectures for Learning Sentence Similarity". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16. Phoenix, Arizona: AAAI Press, pp. 2786–2792.

Navinchandra, D. (1988). "Case-based reasoning in CYCLOPS, a design problem solver". In: *Proceedings of the DARPA Case-Based Reasoning Workshop*, p. 286.

Netten, B. D. (1998). "Representation of failure context for diagnosis of technical applications". In: *Advances in Case-Based Reasoning*. Ed. by Barry xSmyth and Pádraig Cunningham. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 239–250. ISBN: 978-3-540-49797-4.

Nugent, Conor and Padraig Cunningham (2005). "A Case-Based Explanation System for Black-Box Systems". In: *Artif. Intell. Rev.* 24, pp. 163–178. DOI: 10.1007/s10462-005-4609-5.

Oliveira, Evaldo de (2007). "The Rosenblatt Bayesian Algorithm Learning in a Nonstationary Environment". In: *Neural Networks, IEEE Transactions on* 18, pp. 584–588. DOI: 10.1109/TNN.2006.889943.

Ontanon, Santiago et al. (2014). "Case-Based Prediction of Teen Driver Behavior and Skill". In: pp. 375–389. ISBN: 978-3-319-11208-4. DOI: 10.1007/978-3-319-11209-1_27.

Oord, A., S. Dieleman, and B. Schrauwen (2013). "Deep content-based music recommendation". In: *Advances in Neural Information Processing Systems*.

Orecchioni, Amandine et al. (2007). "Learning Incident Causes". In:

Owoputi, O. et al. (2013). "Improved part-of-speech tagging for online conversational text with word clusters". In: *Proceedings of NAACL-HLT* 2013, pp. 380–390.

Öztürk, Pinar, R. Rajendra Prasath, and Hans Moen (2010). "Distributed Representations to Detect Higher Order Term Correlations in Textual Content". In: *Rough Sets and Current Trends in Computing*. Ed. by Marcin Szczuka et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 740–750. ISBN: 978-3-642-13529-3.

Patterson, D., M. Galushka, and N. Rooney (2003). "Efficient Real Time Maintenance of Retrieval Knowledge in Case-Based Reasoning". In: *ICCBR*. Springer-Berlin.

Petridis, Miltos, Soran Saeed, and Brian Knight (2010). "An automatic case based reasoning system using similarity measures between 3D shapes to assist in the design of metal castings". In:

Phyllis, Koton (1988). "Using experience in learning and problem solving". In:

Pla, Albert et al. (2013). "EXiT*CBR.v2: Distributed case-based reasoning tool for medical prognosis". In: *Decision Support Systems* 54, pp. 1499–1510. DOI: 10.1016/j.dss.2012.12.033.

Plate, T. A. (1995). "Holographic reduced representations". In: *IEEE Transactions on Neural Networks* 6.3, pp. 623–641.

Plaut, Elad (2018). *From Principal Subspaces to Principal Components with Linear Autoencoders*. arXiv: 1804.10253 [stat.ML].

Plaza, Enric (2008). "Semantics and Experience in the Future Web". In: pp. 44–58. DOI: 10.1007/978-3-540-85502-6_3.

Plaza, Enric and Lorraine Mcginty (2005). "Distributed case-based reasoning". In: *The Knowledge Engineering Review* 20.3, pp. 261–265. DOI: 10.1017/S0269888906000683.

Plaza, Enric and Santiago Ontanon (2001). "Ensemble Case-Based Reasoning: Collaboration Policies for Multiagent Cooperative CBR". In: vol. 437-451, pp. 437–451. DOI: 10.1007/3-540-44593-5_31.

Prentzasm, Jim and Ioannis Hatzilygeroudis (2007). "Categorizing approaches combining rule-based and case-based reasoning". In: *Expert Syst. J. Knowl. Eng.* 24.2, pp. 97–122. DOI: 10.1111/j.1468-0394.2007.00423.x. URL: https://doi.org/10.1111/j.1468-0394.2007.00423.x.

R. Swartout, William (1983). "XPLAIN: a system for creating and explaining expert consulting programs". In: *Computer Compacts* 1, p. 211.

Raymond, J. W., E. J. Gardiner, and P. Willett (2002). "RASCAL: Calculation of Graph Similarity using Maximum Common Edge Subgraphs". In: *The Computer Journal* 45.6, pp. 631–644.

Reichle, Meike, Kerstin Bach, and Klaus-Dieter Althoff (2009). "The SEASALT Architecture and Its Realization within the docQuery Project". In: pp. 556–563. DOI: 10.1007/978-3-642-04617-9_70.

— (2011). "Knowledge Engineering within the Application Independent Architecture SEASALT". In: vol. 1, pp. 202–215. DOI: 10.1504/IJKEDM.2011.037643.

Rekik, Ines, Sabeur Elkosantini, and Habib Chabchoub (2017). "Integration of Case Based Reasoning in Multi-agent System for the Real-Time Container Stacking in Seaport Terminals". In: *Hybrid Artificial Intelligent Systems*. Springer. ISBN: 978-3-319-59649-5. DOI: 10.1007/978-3-319-59650-1_37.

Reuss, Pascal, Rotem Stram, et al. (2016). "FEATURE-TAK - Framework for Extraction, Analysis, and Transformation of Unstructured Textual Aircraft Knowledge". In: *Case-Based Reasoning Research and Development. ICCBR 2016. Lecture Notes in Computer Science*. Atlanta, Georgia: Springer.

Reuss, Pascal, Christian Witzke, and Klaus-Dieter Althoff (2017). "Dependency Modeling for Knowledge Maintenance in Distributed CBR Systems". In: *Case-Based Reasoning Research and Development*. Ed. by Jean Aha David W.and Lieber. Cham: Springer International Publishing, pp. 302–314. ISBN: 978-3-319-61030-6.

Richter, Michael and Rosina Weber (2013). *Case-based reasoning: a textbook*. ISBN: 978-3-642-40166-4.

Riloff, Ellen (2000). "Automatically constructing a dictionary for information extraction tasks". In:

Rissland, Edwina L. (1983). "Examples in Legal Reasoning: Legal Hypotheticals". In: *Proceedings of the 8th International Joint Conference on Artificial Intelligence. Karlsruhe, FRG, August 1983*. Ed. by Alan Bundy. William Kaufmann, pp. 90–93. URL: http://ijcai.org/Proceedings/83-1/Papers/020.pdf.

Roth-Berghofer, Thomas (2004). "Explanations and Case-Based Reasoning: Foundational Issues". In: vol. 3155, pp. 389–403. DOI: 10.1007/978-3-540-28631-8_29.

Roth-Berghofer, Thomas R. (2004a). "Explanations and Case-Based Reasoning: Foundational Issues". In: *Advances in Case-Based Reasoning*. Ed. by Peter Funk and Pedro A. González Calero. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 389–403. ISBN: 978-3-540-28631-8.

— (2004b). "Learning from HOMER, a Case-Based Help Desk Support System". In: *Advances in Learning Software Organizations*. Ed. by Grigori Melnik and Harald Holz. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 88–97. ISBN: 978-3-540-25983-1.

Rutkowski, Leszek (2004). "Adaptive Probabilistic Neural Networks for Pattern Classification in Time-Varying Environment". In: *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* 15, pp. 811–27. DOI: 10.1109/TNN.2004.828757.

S. G. Khoo, Christopher (1996). "Automatic identification of causal relations in text and their use for improving precision in information retrieval". In:

Sani, Sadiq et al. (2017). "kNN Sampling for Personalised Human Activity Recognition". In: pp. 330–344. ISBN: 978-3-319-61029-0. DOI: 10.1007/978-3-319-61030-6_23.

Schank, Roger C. (1983). *Dynamic memory - a theory of reminding and learning in computers and people*. Cambridge University Press. ISBN: 978-0-521-27029-8.

Schurz, Gerhard (2000). "Scientific Explanation: A Critical Survey". In: *Foundations of Science* 1. DOI: 10.1007/BF00145406.

Shalev-Shwartz, Shai (2012). "Online Learning and Online Convex Optimization". In: *Foundations and Trends in Machine Learning* 4, pp. 107–194. DOI: 10.1561/2200000018.

Shiu, Simon and Sankar K. Pal (2004). *Foundations of Soft Case-Based Reasoning*. Hoboken, NJ, USA: John Wiley & Sons, Inc. ISBN: 0471086355.

Simić, Svetlana et al. (2018). "A hybrid case-based reasoning approach to detecting the optimal solution in nurse scheduling problem". In: *Logic Journal of the IGPL* 28.2, pp. 226–238. ISSN: 1367-0751. DOI: 10.1093/jigpal/jzy047. eprint: https://academic.oup.com/jigpal/article-pdf/28/2/226/32932584/jzy047.pdf. URL: https://doi.org/10.1093/jigpal/jzy047.

Simpson, Robert Lee (1985). "A Computer Model of Case-Based Reasoning in Problem Solving: An Investigation in the Domain of Dispute Mediation (Analogy, Machine Learning, Conceptual Memory)". AAI8525622. PhD thesis. USA.

Sizov, Gleb, Pinar Öztürk, and Jozef Štyrák (2014). "Acquisition and Reuse of Reasoning Knowledge from Textual Cases for Automated Analysis". In: *Case-Based Reasoning Research and Development*. Ed. by Lucand Lamontagne and Enric Plaza. Springer International Publishing, pp. 465–479. ISBN: 978-3-319-11209-1.

Skalle, Pål and Agnar Aamodt (2005). "Knowledge-Based Decision Support in Oil Well Drilling". In: *Intelligent Information Processing II*. Ed. by Zhongzhi Shi and Qing He. Boston, MA: Springer US, pp. 443–455. ISBN: 978-0-387-23152-5.

Smyth, Barry and Mark Keane (1997). "Remembering To Forget: A Competence-Preserving Case Deletion Policy for Case-Based Reasoning Systems". In:

Smyth, Barry and Elizabeth McKenna (1999). "Building Compact Competent Case-Bases". In: *Case-Based Reasoning Research and Development*. Ed. by Klaus-Dieter Althoff, Ralph Bergmann, and L.Karl Branting. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 329–342. ISBN: 978-3-540-48508-7.

Sørmo, Frode, Jörg Cassens, and Agnar Aamodt (2005). "Explanation in Case-Based Reasoning–Perspectives and Goals". In: *Artif. Intell. Rev.* 24, pp. 109–143. DOI: 10.1007/s10462-005-4607-7.

Stram, Rotem, Pascal Reuss, and Klaus-Dieter Althoff (2017). "Weighted One Mode Projection of a Bipartite Graph as a Local Similarity Measure". In: *Case-Based Reasoning Research and Development, 25th International Conference on Case-Based Reasoning*. Trondheim, Norway: Springer.

Sugiyama, M. and Motoaki Kawanabe (2011). *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation*. DOI: 10.7551/mitpress/9780262017091.001.0001.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). "Sequence to Sequence Learning with Neural Networks". In: *CoRR* abs/1409.3215. arXiv: 1409.3215. URL: http://arxiv.org/abs/1409.3215.

Sycara, Katia P. (1998). "Multiagent Systems". In: *AI Magazine* 19.2, p. 79. DOI: 10.1609/aimag.v19i2.1370. URL: https://www.aaai.org/ojs/index.php/aimagazine/article/view/1370.

Troiano, L., A. Vaccaro, and Maria Carmela Vitelli (2016). "On-line smart grids optimization by case-based reasoning on big data". In: *2016 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems (EESMS)*, pp. 1–6.

Tsantekidis, A. et al. (2017). "Forecasting Stock Prices from the Limit Order Book Using Convolutional Neural Networks". In: *2017 IEEE 19th Conference on Business Informatics (CBI)*. Vol. 01, pp. 7–12.

TURING, A. M. (1950). "I.—COMPUTING MACHINERY AND INTELLIGENCE". In: *Mind* LIX.236, pp. 433–460. ISSN: 0026-4423. DOI: 10.1093/mind/LIX.236.433. eprint: https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf. URL: https://doi.org/10.1093/mind/LIX.236.433.

Valmadre, Jack et al. (2017). "End-To-End Representation Learning for Correlation Filter Based Tracking". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Verwimp, Lyan and Jerome R. Bellegarda (2019). "Reverse Transfer Learning: Can Word Embeddings Trained for Different NLP Tasks Improve Neural Language Models?" In: arXiv: 1909.04130.

Wang, Qiang et al. (2018). "Fast Online Object Tracking and Segmentation: A Unifying Approach". In: *CoRR* abs/1812.05050. arXiv: 1812.05050. URL: http://arxiv.org/abs/1812.05050.

Watson, I. (1999). "Case-based reasoning is a methodology not a technology". In: *Knowledge-Based Systems* 12.5, pp. 303–308. ISSN: 0950-7051. DOI: https://doi.org/10.1016/S0950-7051(99)00020-9. URL: http://www.sciencedirect.com/science/article/pii/S0950705199000209.

Weerts, Hilde J. P., Werner van Ipenburg, and Mykola Pechenizkiy (2019). "Case-Based Reasoning for Assisting Domain Experts in Processing Fraud Alerts of Black-Box Machine Learning Models". In: *CoRR* abs/1907.03334. arXiv: 1907.03334. URL: http://arxiv.org/abs/1907.03334.

Wender, Stefan and Ian Watson (2014). "Combining Case-Based Reasoning and Reinforcement Learning for Unit Navigation in Real-Time Strategy Game AI". In: *ICCBR 2014*. Springer, pp. 511–525. ISBN: 978-3-319-11208-4. DOI: 10.1007/978-3-319-11209-1_36.

Wess, Stefan, Klaus-Dieter Althoff, and Guido Derwand (1994). "Using k-d trees to improve the retrieval step in case-based reasoning". In: *Topics in Case-Based Reasoning*. Ed. by Stefan Wess, Klaus-Dieter Althoff, and Michael M. Richter. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 167–181. ISBN: 978-3-540-48655-8.

Willett, Peter (1999). "Matching of Chemical and Biological Structures Using Subgraph and Maximal Common Subgraph Isomorphism Algorithms". In: *Rational Drug Design*. Ed. by Donald G. Truhlar et al. New York, NY: Springer New York, pp. 11–38. ISBN: 978-1-4612-1480-9. DOI: 10.1007/978-1-4612-1480-9_3. URL: https://doi.org/10.1007/978-1-4612-1480-9_3.

Wooldridge, Michael (2009). *An Introduction to Multi-Agent Systems*. ISBN: 0470519460.

Xiong, Ning and Peter Funk (2009). "CBR Supports Decision Analysis with Uncertainty". In: *Case-Based Reasoning Research and Development*. Ed. by Lorraine McGinty and David C. Wilson. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 358–373. ISBN: 978-3-642-02998-1.

Xu, Yuhui and Xiaoyun Tian (2015/11). "Internet Big Data Information Analysis and Power Intelligent Automation Risk Prediction Based on Case Based Reasoning". In: *Proceedings of the 2015 3rd International Conference on Machinery, Materials and Information Technology Applications*. Atlantis Press, pp. 1159–1163. ISBN: 978-94-6252-120-9. DOI: https://doi.org/10.2991/icmmita-15.2015.213. URL: https://doi.org/10.2991/icmmita-15.2015.213.

Yamada, Ikuya et al. (2020). "Wikipedia2Vec: An Efficient Toolkit for Learning and Visualizing the Embeddings of Words and Entities from Wikipedia". In: *arXiv preprint 1812.06280v3*.

Zhang, Xinyuan et al. (2019). "Syntax-Infused Variational Autoencoder for Text Generation". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. DOI: 10.18653/v1/p19-1199. URL: http://dx.doi.org/10.18653/v1/p19-1199.

Zhang, Y., Su Zhang, and David Leake (2017). "Maintenance for Case Streams: A Streaming Approach to Competence-Based Deletion". In: *ICCBR*.

Zhang, Yunhua et al. (2018a). "Structured Siamese Network for Real-Time Visual Tracking". In: *Computer Vision – ECCV 2018*. Ed. by Vittorio Ferrari et al. Cham: Springer International Publishing, pp. 355–370. ISBN: 978-3-030-01240-3.

— (2018b). "Structured Siamese Network for Real-Time Visual Tracking". In: *Computer Vision – ECCV 2018*. Ed. by Vittorio Ferrari et al. Cham: Springer International Publishing, pp. 355–370. ISBN: 978-3-030-01240-3.

Zhang, Zhipeng, Houwen Peng, and Qiang Wang (2019). "Deeper and Wider Siamese Networks for Real-Time Visual Tracking". In: *CoRR* abs/1901.01660. arXiv: 1901.01660. URL: http://arxiv.org/abs/1901.01660.

Zhu, Zheng et al. (2018). "Distractor-aware Siamese Networks for Visual Object Tracking". In: *CoRR* abs/1808.06048. arXiv: 1808.06048. URL: http://arxiv.org/abs/1808.06048.

Kareem Amin

Summary

I am an AI researcher with entrepreneurial spirit since 2010. Throughout the past years, I helped many customers in MENA to kick-start their AI Journey and lead their way to success. With more than 11 publications in the AI area, I bring the academic and business knowledge together to define the best approaches for AI projects implementations, by selecting the right technology stack and the right AI approach to achieve my customer's aspirations.

Work Experience

May 2015 – October 2021
German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany
PhD Candidate, Senior Researcher

- I have **11 publications** in the area of hybrid Case-based Reasoning systems. I apply deep learning models (unexplainable) within the CBR (Explainable) to be able to solve problems while giving adequate level of explainability.

March 2021 – present
IBM Cloud & Cognitive Software, Munich, Germany
IBM Watson AIOps Elite – Tech Lead

- Lead conversations with customers to scope the project
- Write project proposals
- Do sizing and define the project plan
- Orchestrate the interaction between the customer and IBM
- Provide feedback on AIOps and suggestions to how to improve WatsonAIOps.

May 2019 – February 2021
IBM Cloud & Cognitive Software, Munich, Germany
IBM Data Science Elite Team – Senior Data / AI Engineer

- Build Data and AI Pipelines (from Data collection to model deployment)
- Lead conversations with customers to scope the project
- Do sizing and define the project plan
- Orchestrate the interaction between the customer and IBM
- Provide feedback on AIOps and suggestions to how to improve IBM Cloud Pak for Data.

February 2018 – February 2019
Accenture GmbH, Munich, Germany
Big Data Team Lead

- Build Big Data Solutions Architecture
- Contribute to pre-sales activities
- Implement Big Data Showcases

- Write proposals and estimate projects time plan

January 2017 – January 2018
Sulzer GmbH, Munich, Germany

Data / AI Engineer

- Help to build the Big Data Team at the company
- Contribute to several Automotive industrial projects
- Write proposals and estimate projects time plan

May 2014 – April 2015
Vision Valley, Egypt

SAP Business One Tech Lead – Built the first SAP Business One team in Egypt

- Orchestrated a team of 5 Senior SAP Consultants.
- Shaped the SAP B1 Team
- Contributed to initiate the SAP B1 new line of business.
- Validate Business Requirements.
- Estimate Projects Budget.
- Design the Business Blueprint.
- Contribute in all Technical and
- Optimize customers business processes.

September 2012 – May 2014
MegaSoft Consulting & Services, Egypt

IBM Maximo/Smart Cloud Control Desk Team Lead

- Coach a team of 7 Junior IBM Maximo/SCCD consultants.
- Contribute to validate business requirements from application services.
- Capture business requirements.
- Implement Maximo/SCCD as per ITIL best practices.
- Deployment activities in terms of user training, administration training and support team hand over for application services.
- Early support for customers.
- Motivate and provide training for new Maximo Consultants in both **Egypt** and **KSA**.

Key Achievements:
Built a team of Maximo Consultants and led them to implement successful projects over **Egypt** and **KSA** in very short period.

Education

09/2011 - 09/2014

Ain Shams University, Cairo, Egypt Master in Computer Science,
• Masters Area: Expert Systems, Enhance Knowledge Discovery and Information Retrieval Processes.
• Thesis Title: Enhancing Cases Retrieval and Adaptation in Case Based Reasoning.

09/2006 - 07/2010

Ain Shams University, Cairo, Egypt Bachelor in Computer Science
• Ranked 2nd in class