Reasoning with Cases and Deep Learning

Kareem Amin[1,3], George Lancaster[6], Stelios Kapetanakis[4], Klaus-Dieter Althoff[1,2], Andreas Dengel[1,3], and Miltos Petridis[5]

[1] German Research Center for Artificial Intelligence, Smart Data and Knowledge Services, Trippstadter Strae 122, 67663 Kaiserslautern, Germany, kareem.amin,klaus-dieter.althoff,andreas.dengel@dfki.uni-kl.de
[2] Institute of Computer Science, Intelligent Information Systems Lab, University of Hildesheim, Hildesheim, Germany,
[3] Kaiserslautern University, P.O. Box 3049, 67663 Kaiserslautern, Germany
[4] School of Computing Engineering and Mathematics, University of Brighton, UK, s.kapetanakis@brighton.ac.uk
[5] Department of Computing, University of Middlesex, London, UK, m.petridis@mdx.ac.uk
[6] School of Computer Science, Electrical and Electronic Engineering, and Engineering Maths, University of Bristol, UK qv18258@bristol.ac.uk

Abstract. Businesses can benefit greatly from analysing their document assets. These can vary greatly from plain text messages across customer support tickets to complex message exchanges and workflow logs within countless business transactions. Decoding text-based domain knowledge can be a challenging task due to the need for a comprehensive representation and evaluation of the business process ontology, activities, rules and paths. To provide an adequate process coverage, significant time and monetary resources should be invested as well as a high maintenance portfolio, especially for large processes and environments that change dynamically. This work investigates a novel natural language processing path which combines Case-based Reasoning and Deep Neural Networks. Our aim is to minimize the effort from domain experts while extracting domain knowledge from rich text, containing domain abbreviations, grammatically incorrect text and mixed language. Our proposed approach seems promising and a possible future direction in the industry.

Keywords: Textual Case-based Reasoning, Deep Neural Networks, Natural Language Processing, Similarity Measures

1 Introduction

Unstructured text is one of the major challenges in automatic knowledge extraction due to text multi-provenance, complex unrelated sources and domain fuzziness. User language can complicate this process further by introducing grammar

errors, writing in mix (multi)languages and by implying concepts that lead to domain ambiguity.Examples of such lexical content ambiguity can be due to difference in intended meaning for the same word, word synonymy (different terms have similar meaning, or text context that affects the meaning of specific words) and others making traditional NLP techniques inadequate to process it.Textual CBR (TCBR) is a CBR [2] sub-branch which is related to Textual Data. Research in the area shows cases where researchers need to be able to build both cases and similarity measures empirically out of their data corpus in order to be able to retrieve the most similar cases. The aforementioned challenges are not new to the TCBR domain, however the current challenges in text richness, data volume and fuzziness urge for disruptive approaches in order to be able to tackle them efficiently.

This work introduces a comparison between fastText word embeddings approach and Word2vec that was presented in a previous work on top of **DeepKAF** framework [20]. The main motivation for this work is to investigate how Deep Learning models can help in decoding complex textual domain knowledge and build efficient similarity measures while requiring the minimum possible input from domain experts. Deep Learning models are effective when dealing with learning from large amounts of structured or unstructured data. Within the CBR paradigm, Deep Learning models can benefit from the available amounts of data and extract meaningful abstract representations. While Deep Learning can learn from large volumes of labeled data, it can also learn from large amounts of unlabeled/unsupervised data [3][4][5], making it attractive for extracting meaningful representations and patterns from Big Data.

The growth of intensive data-driven decision-making has led to extensive use of Artificial Intelligence (AI) technologies [1] since they seem capable to improve it further. Within the Case-based Reasoning community there have been several examples of applying data-driven methods to dynamically improve business decision making. Following its business needs the customer support industry has also adopted a data-centric approach, as companies turn to a priory collected data to optimise their business workflows and the quality of their services[1]. This work extends the **Deep K**nowledge **A**cquisition **F**ramework (DeepKAF) which has been designed to build CBR systems that are applied in real-world industrial domains. CBR is superior to Black-box techniques in several industrial domains where the explainabiltiy of decisions is necessary. However, building successful CBR systems requires substantial effort from the domain experts which causes significant overheads most of the time. DeepKAF is addressing these issues associated with building CBR systems in industrial domains with promising results as it can be seen from previous work (cite DeepKAF paper, and potentially the AI conference Paper).

This paper is structured as follows: Section 2 describes the neural network text embeddings that are used throughout this work. Section 3 presents our application domain. Section 4 presents the followed methodlogy. Section 5 presents the carried-out evaluation with domain experts to ensure the efficiency of our

proposed approach. Section 6 discusses the related work followed by the summary, conclusion and future work in Section 7.

2 Neural Word Embeddings

Most of the Deep Learning models aren't able to process strings or plain text since they require numerical inputs for tasks such as classification, regression, etc... Several NLP systems and techniques treat words as atomic units, therefore, in order to apply a Deep Learning model on NLP, words to be converted to vectors. "Embedding" words or generating word embeddings 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 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 Mikolov et al. introduced Word2Vec to the NLP community [7]. 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.

Facebook Research open sourced their Word Embeddings model fastText [19]. fastText is essentially an extention of Word2vec model, which treats each word as collection of character ngrams. Word2Vec differs from fastText in terms of how it deals with words: Word2Vec "learns" vectors just for complete words (as found in the training corpus) while fastText learns vectors for the n-grams found within each word, as well as each complete word.

The differences in representing words and sentences make fastText able to: 1. Generate better word embeddings for rare words (Word2Vec is highly dependant on the number of occurrences per each word to decide whether it will be included in the embeddings or not) 2. fastText is able to construct a vector for a word

based on its character ngrams even if the word doesn't appear in its training corpus.

In previous work [20], Word2vec model has been implemented using an LSTM network. In this work, we will implement the fastText model and compare the results.

3 Application Domain

Most companies have several dedicated internal Help-Desk teams for customer support since service quality is usually measured via customer satisfaction. For this work the implemented application and any used data is a joint application between the German Research Center for Artificial Intelligence (DFKI) and a Multinational Automotive Company (the company) located in Munich, Germany with branches all over the world. Inside the company, most of its help-desk tickets come through emails to dedicated help-desk teams. 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 they 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) exists or not.The text presented in this case is written in mixed-language mode (German + English), loosely connected grammar and lots of domain specific abbreviations. As reported by domain experts, their processes in place were suffering from a number of issues:

1. A help-desk agent prioritizes or routes a ticket wrongly. Such an action usually leads to longer resolution times
2. Lack of enough experience or specialized knowledge from a help-desk engineer
3. Substantial difficulty to find a suitable solution from a historical knowledge base. The vast majority of engineers have found it detrimentally time consuming and not always capable of finding a solution

4 Methodology

Previous work on DeepKAF [20] has focused on two main neural networks: 1. An LSTM Network used to prioritize the input text 2. An LSTM network used to do further classification on the tickets and then measure the similarity between two texts using the **Word2Vec** word embeddings model.

DeepKAF has been proven effective in the task of finding the most similar cases compared to the traditional key nearest neighbour algorithm which is typically used during a CBR retrieval process. This work uses **fastText** word embeddings model instead of Word2Vec and compares the performance between the two.

For the needs of this work the Case-based cycle is defined as:

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 Natural Language Processing (NLP) case similarities required 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.

4.1 Applied Model Structure

Model optimization is one of the greatest challenges in the implementation of machine learning solutions. Hyperparameters play a vital role in building **fastText** models. During building the **fastText** model and based on previous work on **DeepKAF** and other work that implemented fastText, the hyperparameters are as follows:

1. ngram = 2 (fastText specific parameter)
2. Embedding Dimension = 300
3. Layer Size = 200
4. Iteration = 1

These hyperparameters are defined after several trials upon which we have been comparing the outcomes.

4.2 DeepKAF: The Solution Architecture

DeepKAF solution architecture consists of three main modules (See Figure 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 & Retrieval): This module is responsible for receiving the tickets and doing the ticket content preprocessing/processing, then retrieve the similar tickets from the Case Base (Case Generation, Retrieval & Retain).
3. Graphical User Interface (Data Visualization): This module is responsible for visualizing the results to the system end-users.

4.3 The Hybrid CBR Approach with LSTM & fastText Word Embeddings

This work combines a Deep Neural Network with CBR to capture and decode domain knowledge with the help of applied Deep Learning algorithms in the context of Natural Language Processing (NLP). In our domain emails are being prioritised based on their content and text similarity is measured based on their semantics. We present several Neural Network types to represent a sequence of

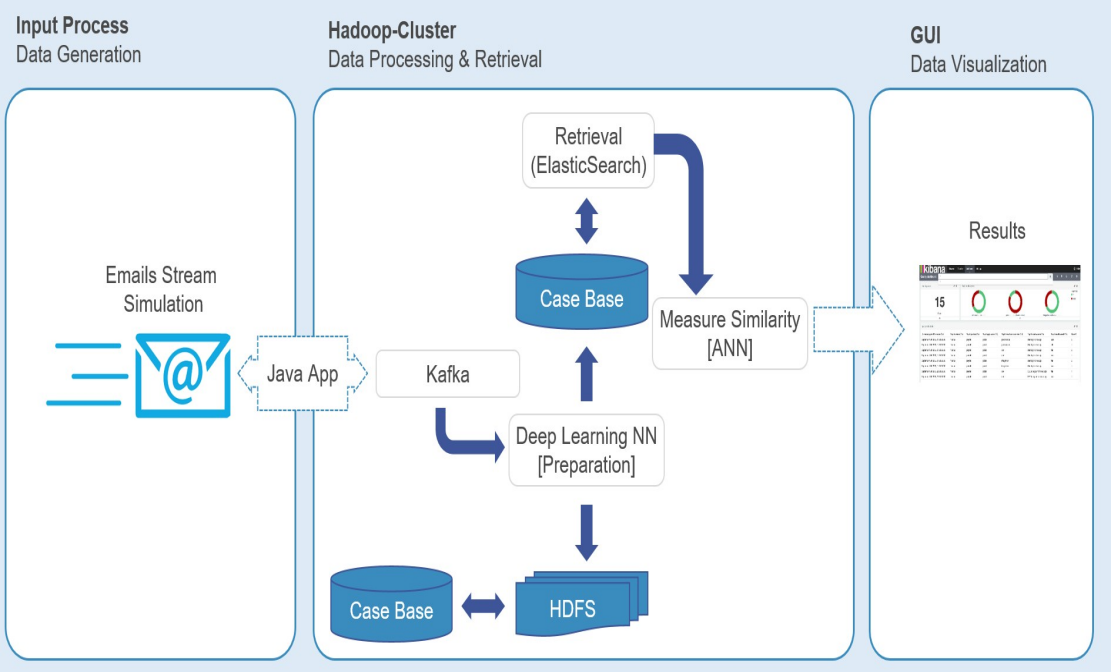

Fig. 1. DeepKAF Solution Architecture

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 was divided into five main processes:

1. P1: Input Process (Data Generation): was responsible for generating and simulating 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 (ex. greetings, signatures etc.) from any email
4. P4: Stemming and stop words elimination: in either German or English language
5. P5: Text vectorization

During the implementation of the first release of DeepKAF, several approaches have been applied. Initially, Support Vector Machines (SVMs) and Vectorization were used to prioritize emails, however, when the case complexity has risen, different Deep Neural Networks Models were applied on an experimental basis: eg. CNNs, RNNs and LSTMs. LSTMs [9] seemed to outperform all the other models. This work uses LSTM to do pre-processing and trim out unnecessary text parts. LSTMs do also a primary filtering of the incoming text

(Process: P2 & P3), and the fastText is used during the retrieval process to measure similarity among different cases.

4.3.1 Vocabulary Containers Vocabulary is one of the knowledge containers and represents the information collected from the domain to express knowledge [10]. By filling in this container we 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, we followed the typical three methods described in [6]. We have used neural networks to remove irrelevant words and extracted the main features that represent certain text using the fastText model [19].

4.3.2 Neural Word Embedding Most of the Deep Learning models aren't able to process strings or plain text. They require numbers as inputs to perform any sort of job, classification, regression, etc... Most existing NLP systems and techniques treat words as atomic units in order to apply a Deep Learning model in NLP, we need to convert words to vectors first. Word embedding is the process of converting text into a numerical representation for further processing as shown in section 2.

4.3.3 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 & signatures). In essence we can consider the Word2Vec model or any other model that could be built as a substitution to the traditional taxonomies.

4.3.4 Similarity Measures Similarity measures are highly domain dependant and used to describe how cases are related to each other. For our system we defined 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 present how we applied Local Similarity Measures followed by Global ones.

Local Similarity Measures: Based on the collected data and discussions with experts, we defined the local similarity measures. We have mainly four distinctive attributes excluding the email subject and content. For the Priority (integer) and Sending Groups (distinctive strings) we used distance functions. For the email subject and content, a word embeddings model was used to help finding the the degrees of similarity among distinctive texts, after applying all the aforementioned prepossessing tasks.

Global Similarity Measures: Global Similarity Measure define the relations among attributes and give an overall weight to the retrieved case. The weight of each attribute demonstrates its importance within the case. We decided to use the weighted euclidean distance for the calculation of the global similarity as applied in [8] . 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 to come up with the overall global case similarity.

5 Experimental Evaluation

DeepKAF [20] was evaluated using the following criteria (Each neural network uses **fastText** to perform its tasks) :

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

As mentioned in **section 3.3**, 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. 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 applied to vectorize text input and build word representations in the vector space (See Figure 2). Sequences of such vectors were processed using various neural net architectures.

5.1 Dataset Characteristics

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

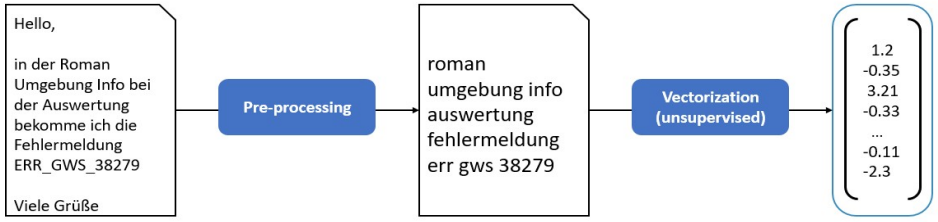

Fig. 2. Text Vectorization

sentences. For that goal, we have prepared a manually generated dataset 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 1 shows a comparison in retrieval results between the Word2vec model that implemented in [20] and fastText model. DeepKAF 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 1: Retrieval Results

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

5.2 Results Discussion & Lessons Learned

From **Table 1**, we can summarize the findings as the following:

1. From a practical perspective, the choice of hyperparameters for generating fastText embeddings becomes key since the training is at character ngram 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

6 Related Work

The related work to this research, is defined on the following three areas: 1. Text processing issues with incorrect sentences and mixed languages 2. Help-desk CBR systems 3. Automation of text relation extraction.

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 the type of CBR systems where the cases are represented as text. 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 [11][12][13], 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 to ease the task of finding similarities between cases and automate the knowledge from textual cases.

HOMER [14] [15] is a help desk support system designer for the same purpose of DeepKAF. 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 and after its further improvement in 2004. 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 DeepKAF, we focused more on how to automate the extraction of similarities and deal with unstructured or mixed-languages text, but this approach also can't be automated to be integrated in the real business environments.

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 [16]. The work of automating knowledge extraction using Neural Networks can be compared to the work presented in [17] 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 authors have used manually constructed lexico-syntactic patterns developed by Khoo [18] to extract the relations between texts.

7 Summary and Future Work

This paper presents a new experimental version of DeepKAF, a hybrid CBR system that uses Deep Neural Networks to assist in automatic feature extractions from text and define similarity across text. Facebook **fasttext** has been applied to measure similarity across unstructured text tickets and compare the retrieved results with **Word2vec**. The applied Hybrid CBR approach has proved the efficiency of using Deep Neural Networks within CBR paradigm. Word embeddings along with Deep Neural Networks can help TCBR systems by building stronger relationships across textual cases and measure similarities with minimal input from domain experts. However, such an approach hides part of the explainability of CBR. Our main goal through this research was to show how a hybrid approach using deep learning and CBR can be competent in dealing with complex tasks. Such an approach seems appropriate to deal with high volumes of data that need to be processed fast and in real-time.

Siamese Networks are the next step for this research. Siamese networks are a special type of neural network architecture. Instead of a model learning to classify its inputs, a neural network learns to differentiate between two inputs and understand the similarity between them. Siamese neural architectures seem capable of demonstrating high accuracy in predicting similarity degrees across objects of high complexity. This approach should be experimented and tested to see how efficient it could be in helping minimizing the initial effort that should be exerted to prepare natural language data.

References

1. Erik Brynjolfsson, Kristina McElheran, Data in Action: Data-Driven Decision Making in U.S. Manufacturing, Center for Economic Studies (CES), January 2016
2. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications 1(7) (March 1994)
3. Bengio Y (2013) Deep learning of representations: Looking forward. In: Proceedings of the 1st International Conference on Statistical Language and Speech Processing. SLSP13. Springer, Tarragona, Spain. pp 137.
4. Bengio Y, LeCun Y (2007) Scaling learning algorithms towards, AI. In: Bottou L, Chapelle O, DeCoste D, Weston J (eds). Large Scale Kernel Machines. MIT Press, Cambridge, MA Vol. 34. pp 321360.
5. Bengio Y, Courville A, Vincent P (2013) Representation learning: A review and new perspectives. Pattern Analysis and Machine Intelligence, IEEE Transactions on 35(8):17981828. doi:10.1109/TPAMI.2013.50
6. Richter, M.M.: Introduction. In: Lenz, M., Bartsch-Sporl, B., Burkhard, H.D., Wess, S. (eds.) Case-Based Reasoning Technology. LNCS (LNAI), vol. 1400, pp.116. Springer, Heidelberg (1998)
7. Tomas Mikolov and Kai Chen and Greg Corrado and Jeffrey Dean, Efficient Estimation of Word Representations in Vector Space, NIPS'13 Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, 2013
8. Bach K., Althoff KD., Newo R., Stahl A. (2011) A Case-Based Reasoning Approach for Providing Machine Diagnosis from Service Reports. In: Ram A., Wiratunga N. (eds) Case-Based Reasoning Research and Development. ICCBR 2011. Lecture Notes in Computer Science, vol 6880. Springer, Berlin, Heidelberg
9. Hochreiter, Sepp & Schmidhuber, Jrgen. Long Short-term Memory. Neural computation. 1997.
10. Richter MM, Lenz M, Bartsch-Sprl B, Burkhard H-D et al, Introduction. In: Case based reasoning technology: from foundations to applications. Lecture notes in artificial intelligence, vol 1400. Springer, Berlin, p 1, 1998
11. Stram R., Reuss P., Althoff KD. (2017) Weighted One Mode Projection of a Bipartite Graph as a Local Similarity Measure. In: Aha D., Lieber J. (eds) Case-Based Reasoning Research and Development. ICCBR 2017. Lecture Notes in Computer Science, vol 10339. Springer, Cham
12. Reuss P., Witzke C., Althoff KD. (2017) Dependency Modeling for Knowledge Maintenance in Distributed CBR Systems. In: Aha D., Lieber J. (eds) Case-Based Reasoning Research and Development. ICCBR 2017. Lecture Notes in Computer Science, vol 10339. Springer, Cham

13. Reuss P. et al. (2016) FEATURE-TAK - Framework for Extraction, Analysis, and Transformation of Unstructured Textual Aircraft Knowledge. In: Goel A., Daz-Agudo M., Roth-Berghofer T. (eds) Case-Based Reasoning Research and Development. ICCBR 2016. Lecture Notes in Computer Science, vol 9969. Springer, Cham
14. Roth-Berghofer T.R, Learning from HOMER, a Case-Based Help Desk Support System. In: Melnik G., Holz H. (eds) Advances in Learning Software Organizations. LSO 2004. Lecture Notes in Computer Science, vol 3096. Springer, Berlin, Heidelberg
15. Gker M. et al., The development of HOMER a case-based CAD/CAM help-desk support tool. In: Smyth B., Cunningham P. (eds) Advances in Case-Based Reasoning. EWCBR 1998. Lecture Notes in Computer Science, vol 1488. Springer, Berlin, Heidelberg
16. Ozturk, Pinar & Prasath, R.Rajendra & Moen, Hans. (2010). Distributed Representations to Detect Higher Order Term Correlations in Textual Content.
17. Sizov G., ztrk P., tyrk J. (2014) Acquisition and Reuse of Reasoning Knowledge from Textual Cases for Automated Analysis. In: Lamontagne L., Plaza E. (eds) Case-Based Reasoning Research and Development. ICCBR 2014. Lecture Notes in Computer Science, vol 8765. Springer
18. Khoo, C.S.G.: Automatic identification of causal relations in text and their use for improving precision in information retrieval. Ph.D. thesis, The University of Arizona (1995)
19. Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov, Bag of Tricks for Efficient Text Classification, dblp computer science bibliography, 2017
20. Amin K., Kapetanakis S., Althoff KD., Dengel A., Petridis M. (2018) Answering with Cases: A CBR Approach to Deep Learning. In: Cox M., Funk P., Begum S. (eds) Case-Based Reasoning Research and Development. ICCBR 2018. Lecture Notes in Computer Science, vol 11156. Springer, Cham