Answering with Cases: A CBR Approach to Deep Learning

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

[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,
s.kapetanakis@brighton.ac.uk
[5] Gluru Research, Gluru, London
stelios@gluru.co
[6] Department of Computing, University of Middlesex, London, UK
m.petridis@mdx.ac.uk

Abstract. Every year tenths of thousands of customer support engineers around the world deal with, and proactively solve, complex help-desk tickets. Daily, almost every customer support expert will turn his/her attention to a prioritization strategy, to achieve the best possible result. To assist with this, in this paper we describe a novel case-based reasoning application to address the tasks of: high solution accuracy and shorter prediction resolution time. We describe how appropriate cases can be generated to assist engineers and how our solution can scale over time to produce domain-specific reusable cases for similar problems. Our work is evaluated using data from 5000 cases from the automotive industry.

Keywords: Case-based Reasoning, Deep Learning, Natural Language Processing

1 Introduction

Effective Customer Support can be a challenge. Both for a company and for a trained system engineer it depends on endless hours of case scanning, a large variety of complex factors and in cases obscure case definitions.To complete a series of tickets successfully a help-desk engineer needs an appropriate prioritization strategy for every working day. The engineer must select a suitable prioritization route, based on the problem description, complexity and historical evidence upon its possible solution. The aim of this work is to help support engineers to achieve the best possible outcome for a given ticket. We propose case-based

reasoning (CBR) as the problem solver by increasing the solution accuracy and provide shorter prediction resolution time.

This work combines deep learning and big data with CBR to automate the acquisition of a domain specific knowledge. The growth of intensive data-driven decision-making has caused broad recognition [1], and the promise that Artificial Intelligence (AI) technologies can augment it even further. Within the Case-based Reasoning community there have been several examples of applying data-driven methods to fast changing work environments with several benefits from it. Recently, the customer experience industry has adopted a data-centric vision in an equivalent way, as companies embrace the power of data to optimise their business workflows and the quality of their services[1].

In this work we focus on large-scale ticket management support, helping help-desk managers to optimize their prioritization strategy and achieve superior performance. A key concept in that of timely ticket resolution, measured in resolved tickets per minute, which usually leads to high resolution vs. lower accuracy. Research on successful customer support ticket resolutions has identified several features that influence resolutions results. For example, the work of Maddern et al. [14] looks at the effect of grammatically incorrect sentences, abbreviations, mix between different languages and semantic challenges.Besides the knowledge containers domain vocabulary: how similarity measures are formulated and are able to identify the adaptation knowledge [7].

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 (CBR, Big Data, DL) [3]. The age of Big Data poses novel ways of thinking to address technical challenges. with Deep Learning neural networks extracting meaningful abstract representations from raw data. 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 [4][5][6], making it attractive for extracting meaningful representations and patterns from Big Data.

The research approach in this paper aims to assess the effect of combining CBR with Deep Learning on overcoming the challenges that come with highly-complex, highly-noisy domains. Our proposed work has been mainly designed and implemented to support Help-Desk engineers in prioritizing and solving new, raw-content tickets as they come from customers. We present a hybrid Textual Case-based reasoning (hTCBR) approach using Deep Neural Networks and Big Data Technologies. hTCBR poses two main advantages: a) it does not rely on manually constructed similarity measures as with traditional CBR and b) it does not require domain expertise to decode the domain knowledge.

This paper is structured as follows: First we describe the application domain and the main limitations of the processes in place. Section 3 explains our ap-

proach, our faced challenges and the followed solution architecture. Section 4 presents the carried-out evaluation with domain experts to ensure the efficiency of our proposed approach. In Section 5 we discuss the related work followed by the summary, conclusion and future work in Section 6.

2 Application Domain

Most companies have a dedicated internal Help-Desk team 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 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 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. 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 from a 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

3 Hybrid Textual CBR Approach on Ticket Management System

3.1 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[8]. 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 we decided our CBR approach 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 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.

3.2 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.

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 Deep Neural Networks and Word Embeddings to improve the text pre-processing and similarity measures.

Sections 3.3 and 3.4 describe the proposed solution architecture along with the tools and methodologies we have applied to overcome the aforementioned challenges.

3.3 DeepTMS: The Solution Architecture

DeepTMS 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 pre-processing/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.

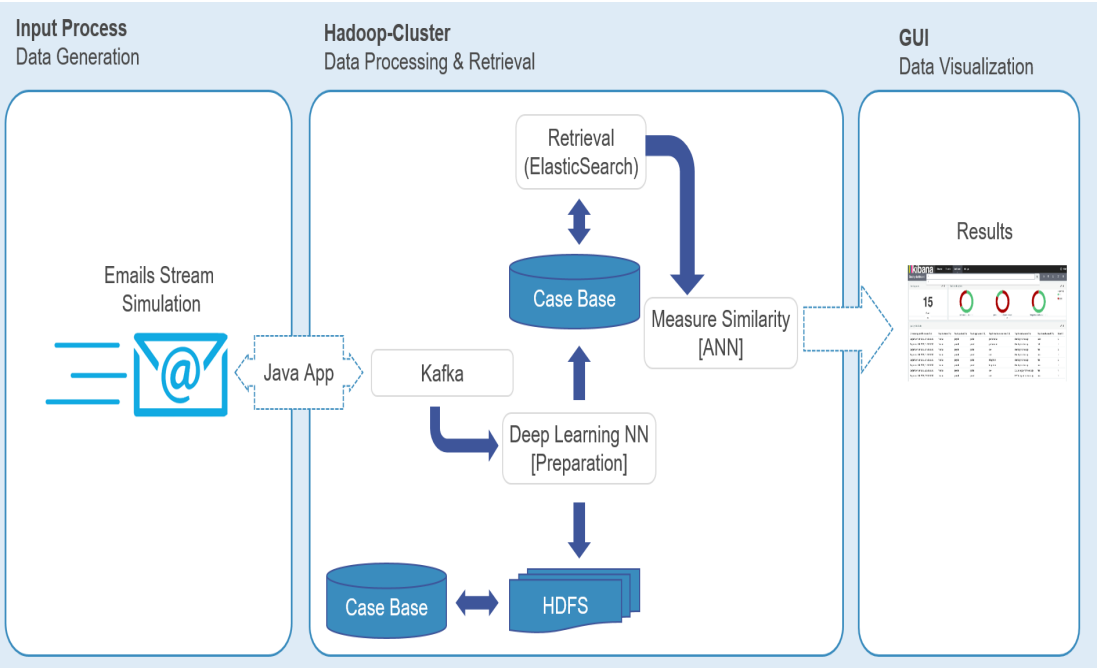

Fig. 1. DeepTMS Solution Architecture

3.4 The Hybrid CBR Approach

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 Natural Language Processing (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, present 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 2)

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: Greedings 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

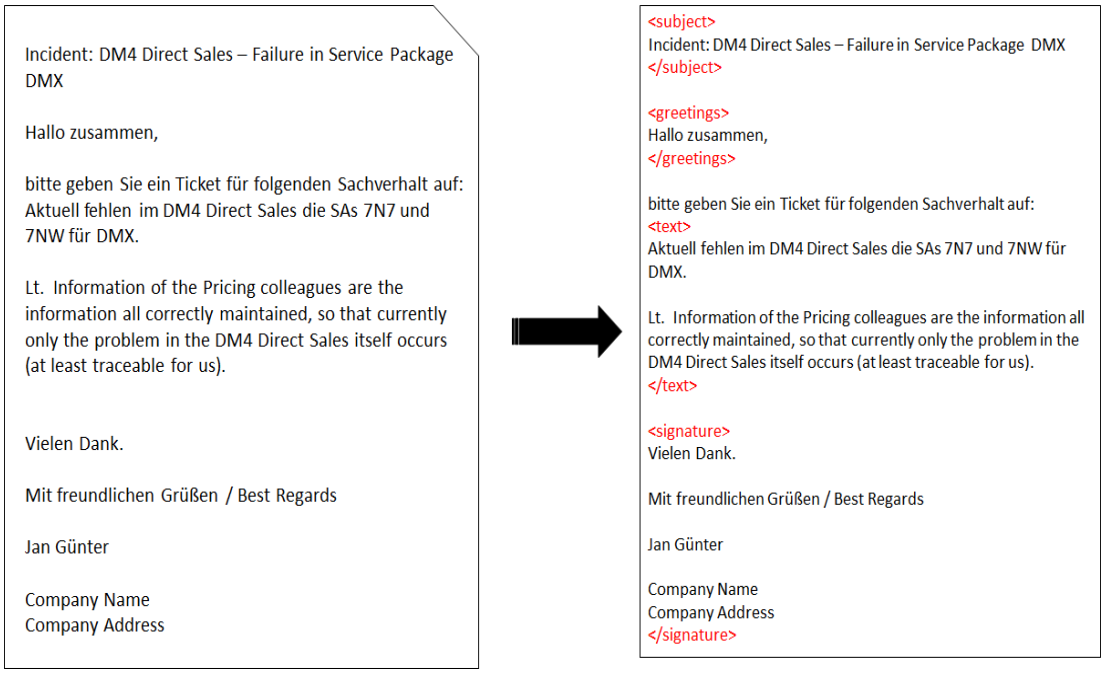

Fig. 2. Ticket Pre-processing

In the beginning, our approach was to use Support Vector Machines (SVM) and Vectorization to prioritize emails. Early results from this approach were promising but not in sub-group cases. When we performed a more intensive test with large volume of emails, it failed to prioritize with high accuracy. Therefore, we decided to build several state of the art neural network models: Convolutional Neural Networks (CNNs), Recurrent Neural Network (RNNs), and Long Short-Term Memory (LSTMs) [16] to test and compare their results. Deep neural network applications seemed to perform substantially better on all sub-groups (Detailed results will be shown in the next section).

3.4.1 Vocabulary Containers Vocabulary is one of the knowledge containers and represents the information collected from the domain to express knowledge [17]. 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 [7]. We have used neural networks to remove irrelevant words and extracted the main features that represent certain text using the Word2Vec models [12]. In the next section we describe how exactly Word2Vec worked to build neural word embeddings.

3.4.2 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... Many current NLP systems and techniques treat words as atomic units, therefore, in order to apply a Deep Learning model on 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 Mitolov et al. introduced Word2Vec to the NLP community [12]. 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 this work we have used Skip-gram model since it demonstrates better performance in semantic task identification [13].

3.4.3 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.

3.4.4 Similarity Measures Similarity measures are highly domain dependant and used to describe how cases are related to each other. In CBR, comparison of cases can be performed along multiple important dimensions [9][11]. Cases that only match partially, can be adapted to a problem situation, using domain knowledge contained in the system [10]. 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. We decided to use the weighted euclidean distance for the calculation of the global similarity as applied in [15] . 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.

4 Experimental Evaluation

Our system 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 [16]. Word2Vec was applied to vectorize text input and build word representations in the vector space (See Figure 3). Sequences of such vectors were processed using various neural net architectures.

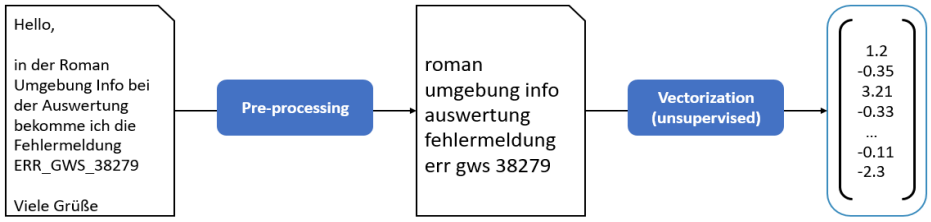

Fig. 3. Text Vectorization

Word2Vec was built using 300,000 historical tickets in an unsupervised training mode. All networks were built with one hidden layer, and utilised 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 1 summarizes the prioritizing stage results.

Table 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 between the three applied models. The evaluation was done with company 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 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 2 shows the results for this stage.

Table 2: Retrieval Results

Level	**Number of Cases**	**Percentage**
One : Three	7764	77.64%
Four : Seven	1468	14.68%
Eight : Ten	692	6.92%
Not Listed	76	0.76%

4.1 Results Discussion & Lessons Learned

During the implementation of DeepTMS, we used neural networks 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 [18]. CNNs are faster in training and processing phases than RNNs and LSTMs. Since an LSTM is a special RNN case they seemed to perform well on text tasks, better t 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 always gets 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 6 times more tickets.

5 Related Work

The related work to this research, is defined on the following three axes: 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 [19][20][21], 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 [22] [23] 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 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 DeepTMS, 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 [24]. The work of automating knowledge extraction using Neural Networks can be compared to the work presented in [25] 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 [26] to extract the relations between texts.

6 Summary

This paper presents DeepTMS, a hybrid CBR system that uses Deep Neural Networks 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. Our main goal through this research 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.

7 Future Work

DeepTMS is a starting point towards similarity measures extraction. In the near future we plan to use the fastText text classifier presented by Facebook [27]. Recurrent Siamese Network models are also considered to improve the efficiency of text similarity measures [28]. DeepTMS is not dealing with too many attributes and the text in the emails is not too long. The results from our presenting implementation are promising, however, it should be tested in more complex environments to show how a hybrid CBR approach can scale. Additional layers to the deep neural networks might be required to be added or more complex models should be applied for better results. The built models can be used to find similarity for any new additional attributes.

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. XUE-WEN CHEN, XIAOTONG LIN, Big Data Deep Learning: Challenges and Perspectives, IEEE Access (Volume: 2), Page(s): 514 - 525,May 2014
4. 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.
5. 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.
6. 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
7. 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)
8. Richter, M. M., Weber, R., Springer-Verlag GmbH. (2016). Case-Based Reasoning: A Textbook.

9. Ashley, K., Modeling Legal Argument, Reasoning with Cases and Hypotheticals. MIT-Press, 1990
10. Aleven, V., Teaching Case-Based Argumentation through a Model and Examples. Ph.D. Dissertation, University of Pittsburgh, Intelligent Systems Program, 1997
11. Stefanie Brninghaus and Kevin D. Ashley (1998) How Machine Learning Can be Beneficial for Textual Case-Based Reasoning. In: Proceedings of the AAAI-98/ICML-98 Workshop on Learning for Text Categorization (AAAI Technical Report WS-98-05). Pages 71-74.Madison, WI.
12. 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
13. Altszyler, Edgar and Sigman, Mariano and Fernndez Slezak, Diego, Comparative study of LSA vs Word2vec embeddings in small corpora: a case study in dreams database, 2016
14. Maddern, M., Maull, R., Smart, A. (2007). Customer satisfaction and service quality in UK financial services. International Journal of Production and Operations Management, 27, 998-1019
15. 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
16. Hochreiter, Sepp & Schmidhuber, Jrgen. Long Short-term Memory. Neural computation. 1997.
17. 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
18. Yoon Kim, Convolutional Neural Networks for Sentence Classification, Conference on Empirical Methods in Natural Language Processing, 2014
19. 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
20. 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
21. 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
22. 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
23. 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
24. Ozturk, Pinar & Prasath, R.Rajendra & Moen, Hans. (2010). Distributed Representations to Detect Higher Order Term Correlations in Textual Content.

25. 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
26. 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)
27. Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov, Bag of Tricks for Efficient Text Classification, dblp computer science bibliography, 2017
28. Jonas Mueller, Aditya Thyagarajan Siamese Recurrent Architectures for Learning Sentence Similarity, Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16), 2016