Learning and Engineering Similarity Functions for Business Recommenders

Hans Friedrich Witschel and Andreas Martin
FHNW University of Applied Sciences and Arts Northwestern Switzerland
School of Business, Riggenbachstrasse 16, CH-4600 Olten, Switzerland
{hansfriedrich.witschel@fhnw.ch, andreas.martin@fhnw.ch}

Abstract

We study the optimisation of similarity measures in tasks where the computation of similarities is not directly visible to end users, namely clustering and case-based recommenders. In both, similarity plays a crucial role, but there are also other algorithmic components that contribute to the end result.

Our suggested approach introduces a new form of interaction into these scenarios that make the use of similarities transparent to end users and thus allows to gather direct feedback about similarity from them. This happens without distracting them from their goal – rather allowing them to obtain better and more trustworthy results by excluding dissimilar items.

We then propose to use the feedback in a way that incorporates machine learning for updating weights and decisions of knowledge engineers about possible additional features, based on insights derived from a summary of user feedbacks. The reviewed literature and our own previous empirical investigations suggest that this is the most feasible way – involving both machine and human, each in a task that they are particularly good at.

Introduction

In many sub-disciplines of artificial intelligence, the notion of *similarity* plays an important role. This is true for instance in clustering (Strehl, Ghosh, and Mooney, 2000) where items are grouped by similarity and in case-based reasoning (CBR) where knowledge is re-used by transferring insights from similar previous cases (Cunningham, 2009). More precisely, CBR usually structures cases into a problem and a solution part and, given a current problem, retrieves past cases with a similar problem description in order to apply (parts of) their previous solution to the current problem.

In this paper, we study possibilities for improving similarity measures for both clustering and case-based reasoning.

Regarding CBR, we will focus more specifically on an important application area, namely *business recommenders*. These are systems that can take over the role of business consultants (or support them in doing their job) by analysing a situation and supporting business decisions by giving appropriate recommendations.

Business recommenders are different from other recommenders in certain respects (see Witschel and Martin, 2018; Felfernig and Burke, 2008) obviously, in business scenarios, the utility of recommendations is defined by business requirements, rather than by a person's preferences or taste. In addition, business recommenders are typically invoked much more rarely than recommenders that consumers use to find products (books, music, movies etc.) of their taste. This implies that a business recommender cannot gather large amounts of information (a profile) about a user – rather, users need to describe their context and requirements in the form of a query when accessing the recommender. Finally, the collection of requirements or context variables can be rather complex, going beyond simple key-value pairs.

All in all, these differences rule out the most popular approaches to building recommender systems, namely collaborative filtering – which relies on large numbers of user ratings and assumes that items do not have an internal structure – and content-based filtering, which constructs user profiles from a repeated interaction between user and system. Instead, case-based recommenders have been proposed (Bridge et al., 2005), which proceed by constructing a description of the business problem at hand, retrieve cases with a similar problem description and combine elements of their solution. While the retrieval part of this approach is not only applicable in business scenarios, but also for e.g. consumers in e-commerce – who may want to express certain requirements that they would like a product to meet (Smyth, 2007) – the combination step is usually obsolete in e-commerce, but not in business recommenders.

In summary, the case-based recommender is a special case of a classical CBR system, where a combination of solution elements from retrieved cases is performed automatically – as opposed to many CBR applications where case combination and adaptation is done manually.

Clustering and case-based (business) recommenders have thus one thing in common: although similarity is a very important ingredient for both of them, the result of the similarity computation is not directly visible to the user. In clustering, the user sees a grouping of similar elements; in a case-based recommendation, the user sees the recommended so-

 In A. Martin, K. Hinkelmann, A. Gerber, D. Lenat, F. van Harmelen, P. Clark (Eds.), Proceedings of the AAAI 2019 Spring Symposium on Combining Machine Learning with Knowledge Engineering (AAAI-MAKE 2019). Stanford University, Palo Alto, California, USA, March 25-27, 2019.

lution elements that have been taken from cases with similar problems. In both methods, there is something in between the similarity computation and what the user sees – in clustering, it is the clustering algorithm, in case-based recommenders, it is the step that selects and combines solution elements from previous similar cases.

Why is this problematic? In many areas where functions are used to score or rank items, such as, e.g. information retrieval, users can see and rate the output of applying the function directly. This allows to *learn* good functions by training learning algorithms with the (implicit) feedback of users (Li, 2011). This has also been studied for classical CBR scenarios (Stahl, 2001; Lamontagne and Guyard, 2014). However, as argued above, when applying similarity functions in clustering or case-based (business) recommenders, this is not possible since the similarity computation is (partly or fully) hidden from the user.

In this position paper, we argue for new ways of interaction between users and both clustering algorithms and case-based (business) recommenders that allow gathering directly similarity-related feedback from users (and learn from it) while maintaining their focus on the utility of the end result. We will thus be able to exploit human knowledge about the similarity in two ways: firstly to adapt certain parts of the similarity function (mainly weights) automatically instead of engineering them manually. Secondly, we suggest exploiting the feedback also to support knowledge engineers in extending the similarity function, e.g. by including more attributes.

Related Work

Clustering and CBR have quite different types of similarity measures.

In clustering, the data objects to be clustered are usually described by vectors which often comprise a certain type of variable – demanding similarity measures for vectors of binary (Choi, Cha, and Tappert, 2010; Lesot, Rifqi, and Benhadda, 2009), categorical (Boriah, Chandola, and Kumar, 2008) or numerical variables (Lesot, Rifqi, and Benhadda, 2009). Challenges begin to appear when objects are described by a mixture of variable types (Cheung and Jia, 2013). Especially for categorical attributes, since the overall similarity of objects is based on a combination of local, per-attribute similarities, the question arises how much each local similarity should contribute to the global similarity. Similarly, when both categorical and numerical variables are present, it must be decided how contributions of both should be combined and weighted (Cheung and Jia, 2013).

In CBR, the situation is often different because cases can have a complex structure – i.e. they cannot be represented simply by key-value pairs, but must be characterised by *relations* to other objects. These relations can be n:m, i.e. a case might be related to several objects of a certain kind, whose number can deviate from case to case. For instance, one may wish to characterise the case of a company by the IT systems that the company runs – obviously, this is not a simple attribute of a company, but an n:m relation between companies and IT systems. A common way to represent such case structures is to use graphs or ontologies (Martin, 2016; Martin et al., 2017; Martin and Hinkelmann, 2018). Corresponding similarity measures have been developed (Witschel et al., 2015; Ontañón and Plaza, 2012; Hefke et al., 2006). Just like similarity measures used in clustering, both simple attribute-based and relational case representations lead to the question of adequate *weights* when combining local attribute-level or relation-level similarities into a global similarity.

Identifying the attributes that should characterise data objects or case content, the selection of the similarity functions and the definition of the view-point specific importance of the characterisation items expressed by weights is a task where human knowledge and experience are needed (Stahl, 2002; Martin, 2016; Martin and Hinkelmann, 2018).

Most approaches to *automatically learning* aspects of similarity in CBR focus on learning the weights that are used to combine local similarities into a global similarity (Stahl, 2001; Lamontagne and Guyard, 2014). In order to do so, user feedback about the utility of retrieved or ranked items is gathered and then used to update weights using either a form of gradient descent (Lamontagne and Guyard, 2014; Stahl, 2001), Bayesian inference (Abdel-Aziz, Strickert, and Hüllermeier, 2014) or genetic algorithms (Jarmulak, Craw, and Rowe, 2000).

In (Lamontagne and Guyard, 2014), feedback is distinguished into *relevance feedback* – i.e. binary feedback about the utility of retrieved cases – and ranking feedback – i.e. a specification of a desired ranking by the user. Stahl (2001), on the other hand, argues that users will not be able to specify either absolute utility or full rankings, but rather only "to compare the utility of two given cases". Such "case order feedback" is then employed for learning.

In clustering, learning similarity metrics directly from pairs of objects labelled as either similar or dissimilar (Ying and Li, 2012) or from relative comparisons (in the form "A is closer to B than to C") (Schultz and Joachims, 2004) has been quite extensively studied. However, such studies all focus on objects represented by purely numerical attributes.

Despite all these efforts, to the best of our knowledge, there is no suitable approach to learn similarity metrics for either clustering or CBR that a) works in tasks where similarities are used, but not directly visible to users, b) works for object representations that are relational and/or using mixed attribute types and c) works in a way that exploits both the strengths of machine learning – in e.g. adapting weights based on feedback – and of humans, in e.g. identifying the right attributes to describe and compare data objects or cases.

A Novel Similarity Engineering Process

This section presents a novel method for developing similarity functions including weights, based on earlier findings from practice and related work.

Central insights and assumptions

Considering the characteristics of case-based (business) recommenders and clustering algorithms, as described in the Introduction and related work from the previous section, a suitable solution for learning similarity metrics will be based on the following insights:

- Insight 1: It is a cognitively intensive task for humans to build an initial case characterisation in case-based recommenders, because, firstly, cases can have a complex structure and secondly, the characterisation needs to be generalised (Martin, 2016).
- Insight 2: Humans are not good at estimating weights, e.g. for weighted-sum global similarity functions. Having them do so forces them to make subjective decisions that can hardly be justified by any concrete experience or explicit knowledge Stahl (2002).
- Insight 3: It is a challenging task for humans to derive from the individual mental similarity models a unified similarity model, which can be used for a configuration of a case-based recommender. This configuration of a consolidated mental similarity model is made by determining global and local similarity functions and assigning weights, which requires profound expert knowledge (Martin, 2016).
- Insight 4: Both case-based recommenders and clustering algorithms represent situations in which the result of similarity computation is not directly visible to humans. The utility of the results that the user does see (recommendations and clusters) depends also on other algorithmic components. This makes it impossible to use the feedback of humans regarding the utility of these results directly for the tuning of the similarity measure (see argumentation in the Introduction).
- Insight 5: However, humans are assumed to be capable of providing feedback regarding either relative comparisons Stahl (2002); Schultz and Joachims (2004) or – in a binary form – regarding the utility or relevance of retrieved items (Lamontagne and Guyard, 2014).
- Insight 6: Algorithms for learning of similarity measures usually focus on weight adaptation. It is hard to design them to identify and suggest missing attributes, i.e. attributes that should be additionally incorporated into a similarity measure. This is typically still a human task.

In summary, these insights suggest that a manually crafted similarity function can suffer mainly from two flaws: firstly, because of the difficulty to model similarity as a whole (see insights 1 and 3), similarity functions may not include some of the attributes that would be necessary to accurately define what makes two objects similar (resulting in insight 6). Secondly, because humans are not good at specifying weights (see insight 2), similarity functions may have suboptimal weights. Our new approach for engineering similarity functions attempts to remove both flaws.

The new similarity engineering process

Below, we describe a novel procedure for engineering similarity functions – as a joint venture between machine and human. The approach is inspired by an interaction mechanism described in one of our previous works (von Rohr, Witschel, and Martin, 2018). In that work, the goal was to estimate the effort of new projects based on previous experience. As shown in Figure 1, we built a system that performed the retrieval step of CBR and then learned a regression model from the n most similar cases to predict the effort for new projects. Here, the – otherwise usually invisible – result of the similarity computation in the retrieval step was made visible to the users, allowing them to discard projects considered dissimilar to the new one. This, in turn, led to a different recommendation.

Based on this idea, we propose the following procedure for learning similarity functions in case-based recommenders; an overview is given in Figure 2.

1. User involvement: Suppose that a case-based recommender, given a user query q, retrieves a set C of the n cases that are most similar to q. Assume further that the recommender combines the solutions of the cases in C into a new solution. What we propose is to show the cases $c \in C$ to the user and allow him/her to remove ones that are not considered similar to q, see Figure 1. This will lead to a different recommendation outcome – i.e. it gives the user more control over how recommendations are derived. As our experiments in (von Rohr, Witschel, and Martin, 2018) have shown, this is something that users feel confident to do and that even increases their trust in the final recommendations. As the figure also indicates, the user is also able to access a complete description of the case (in this case a project) via a link before deciding whether or not to exclude it. We suggest to always incorporate this possibility, i.e. making full case information available to users via a link.
2. Assuming that the similarity function used in step 1 was initially designed by humans and initialised with some human-estimated weights for combining attribute-level local similarities, we can now use the user feedback from step 1 to adapt these weights (remember that we assumed that humans are not good at estimating weights). For instance, as suggested in (von Rohr, Witschel, and Martin, 2018), the adaptation might be based on an evolutionary algorithm, using the average precision of a case ranking as a fitness function. This means that the algorithm learns weights in a way such that the final scores of cases selected/accepted by humans will be higher than the scores of rejected cases. If this is the case consistently, then the average precision will be optimal. Further details can be found in (von Rohr, Witschel, and Martin, 2018).
3. In a final step, after a certain number of users have used the recommender in the way described in step 1, we suggest to gather the data and display it to a knowledge engineer. More precisely, the system should show a summary of the interactions where users excluded cases. Among those, most attention should be given to those interactions where the excluded cases were almost identical to other, not excluded cases in the same ranking. Such situations will help the knowledge engineer to identify further attributes that might be relevant to describe cases. To

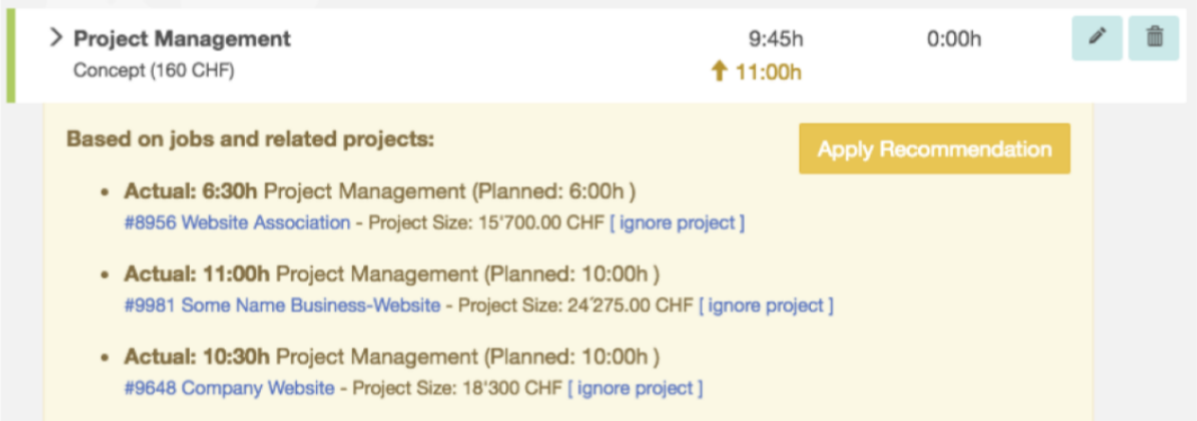

Figure 1: A screenshot of a case-based recommender with explicit similarity feedback from (von Rohr, Witschel, and Martin, 2018)

understand the rationale behind this, let us assume that e.g. projects are described by three attributes, namely a purpose (a_1, free text), a client name (a_2) and a number of requested features (a_3). Let us further assume that the recommender retrieves 3 cases, A, B and C, in the retrieval step, ranks them in that order and that they all have identical values for a_2 and a_3 and nearly identical values for a_1. Finally, let's assume that the user, having had a close look at the complete textual description of all three projects (which we assume to convey more information than the three attributes), decides to exclude B. Obviously, this decision can only be explained by a difference between the projects that is not conveyed by any of the attributes a_1, a_2, a_3. A human knowledge engineer, by studying such an example, may be inspired to include an additional attribute into the computation of similarity that will allow ranking A, B and C correctly by establishing the difference between them that led the user to exclude B, but not A and C.

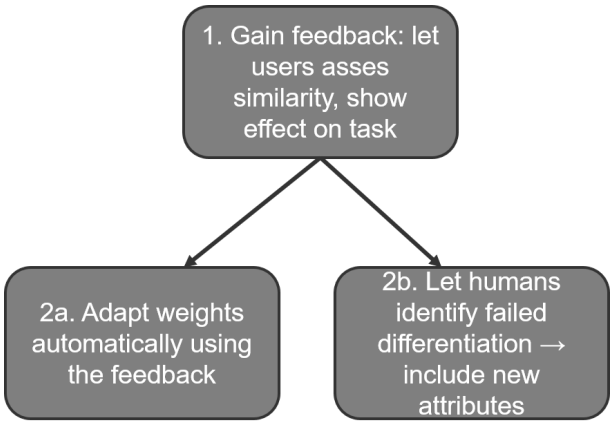

Figure 2: An overview of the suggested approach for similarity metric engineering

We need to show how our approach can be adapted to clustering. For this, we only need to adapt step 1: instead of recommendations and their "explanation", the user of a clustering algorithm will see groups of elements. We recommend that, as in the recommendation scenario, the user should be able to access detailed descriptions of each cluster member. In addition, the system should show pairwise links between all cluster members and allow the user to remove them, as shown in Figure 3. Again, when a user has finished analysing a cluster, such feedback will result in a re-clustering and an updated result.

As an example and to motivate this, consider a very popular application area of clustering in business, namely customer segmentation. Segments identified by a clustering algorithm will be used to address segments in a different way, e.g. in marketing campaigns. A user (e.g. a member of the marketing department of a company) studying the result in Figure 3 might feel that customers 1 and 2 should not be treated in the same way. Being able to express this concern will result in re-clustering (and thus hopefully in a better segmentation) and increase the trust of the user towards that re-clustered result.

Steps 2 and 3 will then work analogously – feedback about pairwise similarity or dissimilarity can be exploited to automatically update weights, whereas situations, where pairs of objects with near-identical feature values are marked as dissimilar, may inspire a human knowledge engineer to introduce additional features.

Obviously, the approach will not be suitable for very large clusters. For these, some "typical" cluster members (e.g. ones close to a centroid) might be selected and displayed.

Discussion and Conclusion

We argue that this approach is suitable because it assigns tasks to both the machine and the human that each of them is good at: the machine uses the feedback to learn weights, the human to engineer additional features, based on examples of

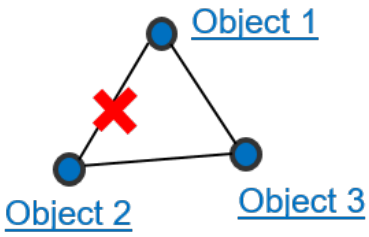

Figure 3: An interaction concept for similarity feedback in clustering – showing a cluster with three members and an interaction where the user declares objects 1 and 2 as dissimilar

failed differentiation.

We furthermore claim that humans who use the recommender will be willing to provide such feedback in step 1, especially because it allows them to improve the resulting recommendations, to gain more control and more trust in the result – something that we have already shown empirically in (von Rohr, Witschel, and Martin, 2018). Thus, we have found a way to gain explicit feedback about similarity, without distracting the human from the goal to receive useful recommendations.

References

Abdel-Aziz, A.; Strickert, M.; and Hüllermeier, E. 2014. Learning solution similarity in preference-based cbr. In *International Conference on Case-Based Reasoning*, 17–31. Springer.

Boriah, S.; Chandola, V.; and Kumar, V. 2008. Similarity measures for categorical data: A comparative evaluation. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, 243–254. SIAM.

Bridge, D.; Göker, M. H.; McGinty, L.; and Smyth, B. 2005. Case-based recommender systems. *The Knowledge Engineering Review* 20(3):315–320.

Cheung, Y.-M., and Jia, H. 2013. Categorical-and-numerical-attribute data clustering based on a unified similarity metric without knowing cluster number. *Pattern Recognition* 46(8):2228–2238.

Choi, S.-S.; Cha, S.-H.; and Tappert, C. C. 2010. A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics* 8(1):43–48.

Cunningham, P. 2009. A Taxonomy of Similarity Mechanisms for Case-Based Reasoning. *IEEE Transactions on Knowledge and Data Engineering* 21(11):1532–1543.

Felfernig, A., and Burke, R. 2008. Constraint-based recommender systems: technologies and research issues. In *Proceedings of the 10th international conference on Electronic commerce*, 3. ACM.

Hefke, M.; Zacharias, V.; Abecker, A.; Wang, Q.; Biesalski, E.; and Breiter, M. 2006. An Extendable Java Framework for Instance Similarities in Ontologies. In *Proceedings of ICEIS 2006*, 263–269.

Jarmulak, J.; Craw, S.; and Rowe, R. 2000. Genetic algorithms to optimise cbr retrieval. In *European Workshop on Advances in Case-Based Reasoning*, 136–147. Springer.

Lamontagne, L., and Guyard, A. B. 2014. Learning case feature weights from relevance and ranking feedback. In *FLAIRS Conference*.

Lesot, M.-J.; Rifqi, M.; and Benhadda, H. 2009. Similarity measures for binary and numerical data: a survey. *International Journal of Knowledge Engineering and Soft Data Paradigms* 1(1):63.

Li, H. 2011. Learning to rank for information retrieval and natural language processing. *Synthesis Lectures on Human Language Technologies* 4(1):1–113.

Martin, A., and Hinkelmann, K. 2018. *Case-Based Reasoning for Process Experience*. Cham: Springer International Publishing. 47–63.

Martin, A.; Emmenegger, S.; Hinkelmann, K.; and Thönssen, B. 2017. A viewpoint-based case-based reasoning approach utilising an enterprise architecture ontology for experience management. *Enterprise Information Systems* 11(4):551–575.

Martin, A. 2016. *A combined Case-based Reasoning and Process Execution Approach for Knowledge-Intensive Work*. Phd thesis, University of South Africa.

Ontañón, S., and Plaza, E. 2012. Similarity measures over refinement graphs. *Machine learning* 87(1):57–92.

Schultz, M., and Joachims, T. 2004. Learning a distance metric from relative comparisons. In *Advances in neural information processing systems*, 41–48.

Smyth, B. 2007. Case-based recommendation. In *The adaptive web*. Springer. 342–376.

Stahl, A. 2001. Learning feature weights from case order feedback. In *International Conference on Case-Based Reasoning*, 502–516. Springer.

Stahl, A. 2002. Defining similarity measures: Top-down vs. bottom-up. In *European Conference on Case-Based Reasoning*, 406–420. Springer.

Strehl, A.; Ghosh, J.; and Mooney, R. 2000. Impact of similarity measures on web-page clustering. In *Workshop on artificial intelligence for web search (AAAI 2000)*, volume 58, 64.

von Rohr, C. R.; Witschel, H.; and Martin, A. 2018. Training and re-using human experience: a recommender for more accurate cost estimates in project planning. In *Proceedings of the 10th International Conference on Knowledge Management and Information Sharing (KMIS)*.

Witschel, H., and Martin, A. 2018. Random Walks on Human Knowledge: Incorporating Human Knowledge into Data-Driven Recommenders. In *Proceedings of the 10th International Conference on Knowledge Management and Information Sharing (KMIS)*.

Witschel, H.; Martin, A.; Emmenegger, S.; and Lutz, J. 2015. A new Retrieval Function for Ontology-Based Complex Case Descriptions. In *Proceedings of CBR-MD'15*.

Ying, Y., and Li, P. 2012. Distance metric learning with eigenvalue optimization. *Journal of Machine Learning Research* 13(Jan):1–26.