# PyTFL: A Python-based Neural Team Formation Toolkit

Radin Hamidi Rad
radin@ryerson.ca
Ryerson University, Canada

Aabid Mitha
aabid.mitha@ontariotechu.net
Ontario Tech University, Canada

Hossein Fani
hfani@uwindsor.ca
University of Windsor, Canada

Mehdi Kargar
kargar@ryerson.ca
Ryerson University, Canada

Jaroslaw Szlichta
jarek@ontariotechu.ca
Ontario Tech University, Canada

Ebrahim Bagheri
bagheri@ryerson.ca
Ryerson University, Canada

## ABSTRACT

We present PyTFL, a library written in Python for the team formation task. In team formation task, the main objective is to form a team of experts given a set of skills. We demonstrate an efficient and well-structured open-source toolkit that can easily be imported into Python. Our toolkit incorporates state of the art approaches for team formation, e.g., neural based team formation, and supports team formation sub-tasks such as collaboration graph preparation, model training and validation, systematic evaluation based on qualitative and quantitative team metrics, and efficient team formation and prediction. While there are strong research papers on team formation problem, PyTFL is the first toolkit to be publicly released for this purpose.

## CCS CONCEPTS

• **Information systems** → *Retrieval models and ranking*; **Expert search**; • **Computing methodologies** → *Search methodologies*.

## KEYWORDS

Team Formation, Expert Networks, Task Assignment

**ACM Reference Format:**
Radin Hamidi Rad, Aabid Mitha, Hossein Fani, Mehdi Kargar, Jaroslaw Szlichta, and Ebrahim Bagheri. 2021. PyTFL: A Python-based Neural Team Formation Toolkit. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

As the nature of work is becoming increasingly interdisciplinary, the need to collaboratively work on shared tasks is gaining more importance. Experts need to effectively engage with others to address problems that are not possible to solve or appear in isolation. This necessitates the formation of expert teams that consists of members that (1) have synergistic and complementary skill sets, and (2) can work together efficiently. The team formation task addresses this specific challenge and sets out to find teams of experts that collectively satisfy an input set of desirable skills.

As use cases, the research literature has looked into application areas where the team formation task can be immediately applied. For brevity, we include only a few representative application areas from the literature:

- **Freelancing [2, 3, 12]:** This employment model allows companies to enter into contractual agreement with self-employed labor for short-term projects. Guru[1] and Freelancer[2] are sample of freelancing websites. For team formation purposes, every freelancer can be considered to be an expert who has a set of skills. Using this analogy, each job posting can be considered to be a team. The process of finding one or more freelancers to satisfy a job posting can be viewed as a team formation process.
- **Collaborative Publishing [6, 9, 10, 14]:** In academic collaboration networks, research papers can be considered to be the output of a successful collaborative research effort. In this context, academic researchers can be considered as experts and a jointly published publication can be the result of the work of a team. The DBLP[3] and arXiv[4] are example datasets that have been used for team formation task in the past.
- **Movies [1, 6]:** The production of a movie is a highly collaborative effort requiring the participation of many people ranging from actors and directors, among others. In the past, researchers have viewed the process of putting the team needed for a movie together as a team formation problem. The IMDB dataset[5] is one of the most popular widely used datasets in this space.
- **Protein-Protein Interactions (PPI) [5, 6]:** Understanding and predicting pairs or groups of protein interactions are important as they can indicate the formation of macromolecular structures. Some researchers have viewed PPI as a team formation task where each protein is considered to be an expert and the genes that each protein carries is its set of skills. BioGrid PPI Dataset[6] is a sample dataset that has been widely used in the literature.

Based on the above sample representative application areas, we note that the team formation task is specially challenging for several reasons: (1) depending on the application area, the number of experts and their skill sets can grow very rapidly and hence

[1]https://www.guru.com/
[2]https://www.freelancer.com/
[3]https://www.aminer.org/citation
[4]https://arxiv.org/help/bulk_data
[5]https://www.imdb.com/interfaces/
[6]https://downloads.thebiogrid.org/BioGRID

the process of matching subsets of experts and skills can become impractical. For instance, several researchers have shown [7] that when modeled through a graph structure and as a subgraph identification problem, the team formation task is NP-hard. As such, with a relatively large search space, heuristic-based graph methods are used with often impractical solutions, (2) the large number of skills and experts leads to a highly sparse network of interactions between experts and skills. This sparsity of the collaboration network makes the process of mapping between skill and expert spaces quite difficult for team formation methods that are based on matrix factorization and neural architectures.

In [10], we found that bayesian neural network based methods form higher-quality teams compared to state-of-art methods. In this demo paper, we introduce PyTFL[7], which is an open-source Python toolkit that provides an easy-to-use and effective neural-based method for the team formation task. This software toolkit addresses the issues associated with the large size of the skill and expert spaces by allowing the users to automatically learn embedding representations for skills and experts, which are then used for mapping between the two spaces. PyTFL also addresses the sparsity problem by offering a variational Bayes neural architecture that is resilient against overfitting for sparse networks.

The demonstration will include:

- Showcasing the extensible architecture offered by PyTFL that comes with off the shelf neural methods for team formation and can also be easily extended to include new methods;
- Presenting how the PyTFL: toolkit can be easily installed from PyPI with a single installation command and then seamlessly integrated into any Python program;
- Displaying that PyTFL supports for many subtasks involved with team formation such as collaboration graph preparation, model training, team formation (inference) and evaluation of formed teams, just to name a few.

We will specifically show the audience of this demonstration how they can effectively use or extend PyTFL for the following sub-tasks in team formation: **Collaboration Network Preparation:** Users can import their custom datasets to the library and transform it into the standard defined format in the form of a heterogeneous graph structure. PyTFL can support various input data presentations such as CSV. The standard format stores the embedding vectors of skill, expert sets and along with the unique IDs for each record; **Training Team Formation Models** Once the collaboration network is loaded, PyTFL allows to train a team formation model. The modular implementation makes it also possible for users to modify the training and create variations of it. In addition, the users can switch out and replace the training procedure used to build a team formation model without hurting the other parts of the pipeline; **Team Inference (generation)** PyTFL can infer new teams based on a trained team formation model given an input set of skills. The users can predict teams after the training process either in a pipeline or in parallel. A user can save a snapshot of model at anytime and serialize it for later use; **Evaluation** The toolkit also allows the users to systematically evaluate the team formation models using various types of quantitative and qualitative metrics. These include, but are not limited to Mean Average Precision (MAP), Recall, Normalized Discounted Cumulative Gain (NDCG), and Mean Reciprocal Rank (MRR), to name a few; **Benchmarking** In order to compare different models with each other, PyTFL provides functionalities to help users compare two team formation models with each other with significant depth. For instance, it allows for finding the correlation between correct candidate teams and the expected teams, and by plotting metric trends based on the input test datasets.

[7]https://pypi.org/project/pytfl/

## 2 RELATED WORK

From a high-level perspective, work in the area of team formation can be broadly categorized into three classes, namely graph-based, neural-based and recommendation-based methods.

On of the earliest approaches for team formation was first introduced in Lappas et al. [9] where the authors proposed optimization functions to measure the communication cost of a team. They used the DBLP dataset for evaluation. Moreover, by adding more objectives to the problem a new set of solutions were later introduced in [5, 6, 9]. The downside of these approaches is that they are all heuristic-based methods that attempt to reduce the time complexity of graph-based search through some optimization mechanisms. From an implementation point of view, the code for these methods are not publicly available; however, we have been able to receive Java-based implementation of these methods directly from the authors. The implementations are primarily proof-of-concept and lack documentation.

More recent approaches to team formation focus on neural architectures. These methods rely on a neural network to learn a mapping from the skills space to the experts spaces. In our own prior work [10], we hypothesised that a neural network can be trained using past collaboration between experts such that it satisfies two main criteria: (1) maximal coverage for a given set of skills, and (2) maximal collaboration history among team members. Similarly, Sapienza et. al. [11], have used an autoencoder architecture to create a mapping between skill and expert domains. Their neural network learns the mapping using the adjacency matrix that represents the expert directed networks. While these neural methods do have Github repositories[8], they are designed to serve as proof of concept and not for practical use by experts.

The third group of methods consider team formation as a task that can be modeled through a recommendation process. For instance, the recurrent recommender network (rrn) [13] that employs an LSTM-based autoregressive model has been used to learn the mapping between skills and experts. Other methods such as `svd++` method [8] have also been used for this task in the past. The efficient implementation for these methods are widely available and are quite stable; however, these are generic recommender system implementations and are not specifically built and delivered for the team formation task. Therefore users need to customize these tools before they can be used for team formation. Furthermore, these methods have shown much weaker performance compared to neural methods on the team formation task [10] as such they are not ideal choices for this task.

To the best of our knowledge, this is the first library dedicated to the team formation task that provides its users with (1) access to

[8]https://github.com/radinhamidi/Retrieving-Skill-Based-Teams-from-Collaboration-Networks

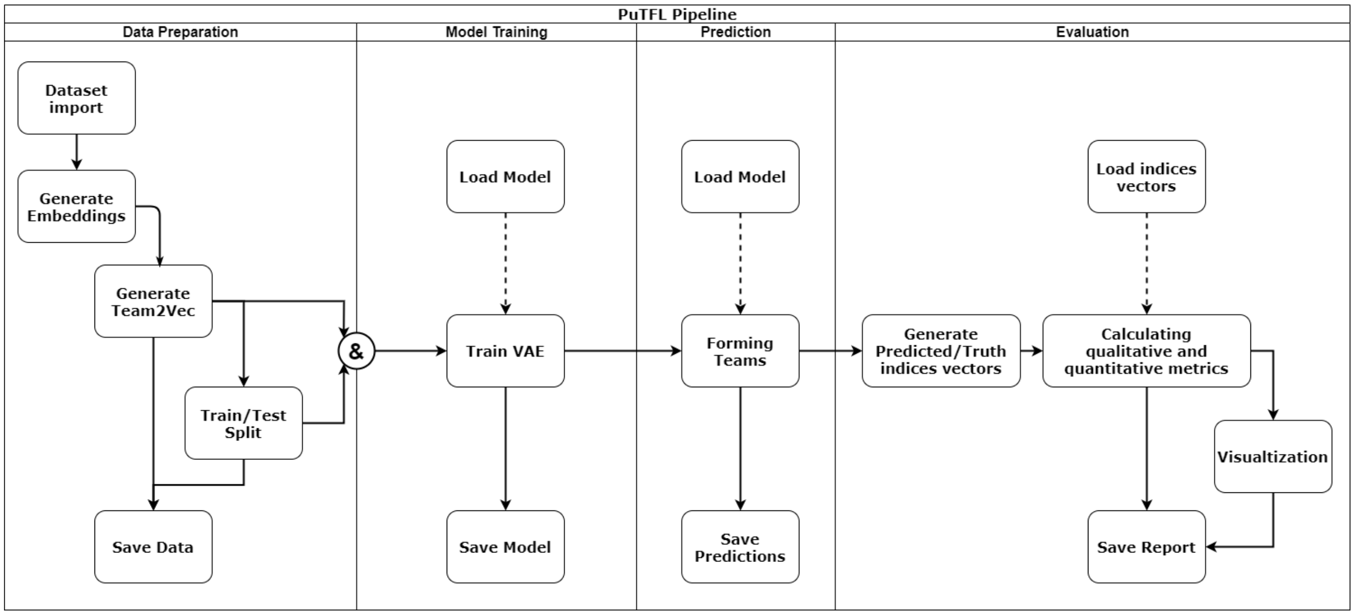


**Figure 1: An Overview of the PyTFL Stages.**

a well-maintained state-of-the-art implementation; and (2) easy to use programmatic interface for model training and team generation; and (3) the effective benchmark of the outcomes using different methods.

## 3 PIPELINE

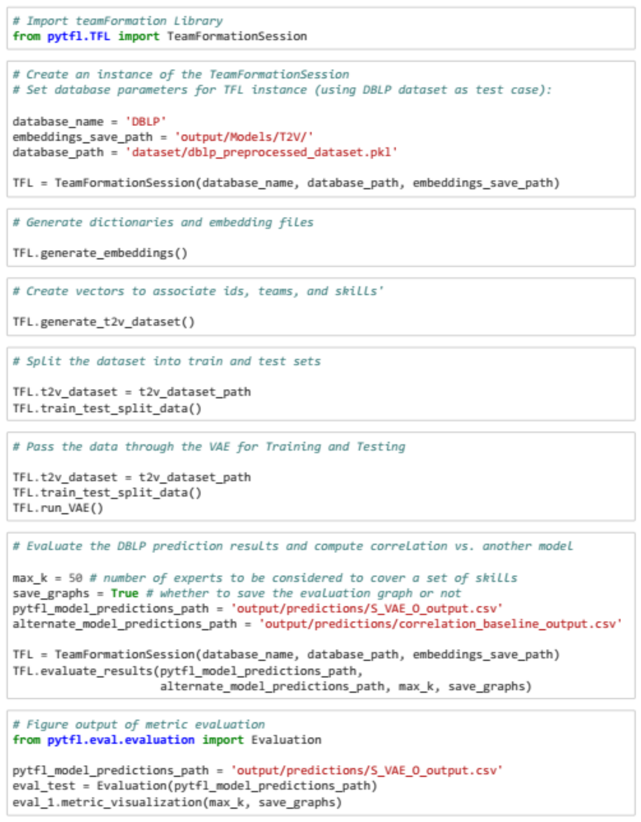
We visually depict the pipeline of the PyTFL package in Figure 1. As shown in this figure, the pipeline consists of four major stages, including collaboration network preparation, model training, prediction and evaluation stages. In the following, we briefly demonstrate relevant code snippets related to each stage of the pipeline.

The PyTFL package is available on PyPI and can be easily imported in Python upon installation (e.g., through pip). Upon importing the *TeamFormationSession* from *TFL*, it is possible to create an instance of the team formation task session, which requires the specification of the task name, the path where models are loaded and saved, as well as path to the dataset pickle. In the collaboration network preparation stage, PyTFL allows the user to easily learn neural embedding representations for skills and experts and generate representations for teams. The process of learning representations for experts, skills and teams is simplified into two commands. As is the case for every stage of PyTFL, the user can opt to serialize and store the products of this stage and load them as necessary in the future.

For the model training stage, we adapt a similar strategy as *sklearn* and provide a method to systematically create test and train datasets. This is not a straightforward task in team formation, as it is important to make sure that skills and experts are distributed across the test and train without leakage. Leakage can happen when experts who have collaborated in multiple past projects are listed across test and train sets. Our `TFL.train_test_split_data()` function ensures that such cases are taken care off effectively. Based on the dataset splits, PyTFL provides the means to train a variational model to map between the skill and expert spaces. The benefit of a variational Bayes neural architecture is that it is robust to overfitting and resilient to sparse collaboration networks [4]. The added benefit of PyTFL is that it provides a clear interface for adding new training modules by the users beyond the built-in variational Bayes neural architecture. Any new methods can be incorporated with the data encoded through the collaboration network preparation stage and models generated based on the expected outputs of the model training stage. This way users can seamlessly add new methods and easily benchmark them.

We also note that PyTFL is quite efficient in terms of generating new teams given a set of desirable input skills. In our earlier work [10], we have benchmarked various team formation methods in terms of their efficiency. We have empirically shown that for different skill and team sizes, the variational Bayes neural architecture is among the most efficient (in terms of team inference time) compared to existing team formation techniques; therefore, the offered team generation (prediction) implementation is fast to be used in practice. In addition to generating teams, in the demo users can also easily benchmark the produced teams through both qualitative and quantitative metrics, which can be exported as reports or visualized as charts. A sample visualization generated by PyTFL on quantitative metrics are shown in Figure 3.

For complete illustration of the programming involved to produce a functional PyTFL implementation and to show how convenient it is to use PyTFL, we have included all of the code required to perform the whole process of data loading to model training to team prediction and evaluation in Figure 2. The simplicity of the code relieves the user from the many details involved in training and testing a team formation task.



```
# Import teamFormation Library
from pytfl.TFL import TeamFormationSession
```

```
# Create an instance of the TeamFormationSession
# Set database parameters for TFL instance (using DBLP dataset as test case):

database_name = 'DBLP'
embeddings_save_path = 'output/Models/T2V/'
database_path = 'dataset/dblp_preprocessed_dataset.pkl'

TFL = TeamFormationSession(database_name, database_path, embeddings_save_path)
```

```
# Generate dictionaries and embedding files

TFL.generate_embeddings()
```

```
# Create vectors to associate ids, teams, and skills'

TFL.generate_t2v_dataset()
```

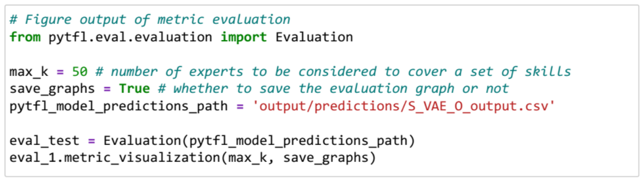
```
# Split the dataset into train and test sets

TFL.t2v_dataset = t2v_dataset_path
TFL.train_test_split_data()
```

```
# Pass the data through the VAE for Training and Testing

TFL.t2v_dataset = t2v_dataset_path
TFL.train_test_split_data()
TFL.run_VAE()
```

```
# Evaluate the DBLP prediction results and compute correlation vs. another model

max_k = 50 # number of experts to be considered to cover a set of skills
save_graphs = True # whether to save the evaluation graph or not
pytfl_model_predictions_path = 'output/predictions/S_VAE_O_output.csv'
alternate_model_predictions_path = 'output/predictions/correlation_baseline_output.csv'

TFL = TeamFormationSession(database_name, database_path, embeddings_save_path)
TFL.evaluate_results(pytfl_model_predictions_path,
                     alternate_model_predictions_path, max_k, save_graphs)
```

```
# Figure output of metric evaluation
from pytfl.eval.evaluation import Evaluation

pytfl_model_predictions_path = 'output/predictions/S_VAE_O_output.csv'
eval_test = Evaluation(pytfl_model_predictions_path)
eval_1.metric_visualization(max_k, save_graphs)
```

**Figure 2: Sample Code Snippet for PyTFL illustrating simplicity of our library.**



```
# Figure output of metric evaluation
from pytfl.eval.evaluation import Evaluation

max_k = 50 # number of experts to be considered to cover a set of skills
save_graphs = True # whether to save the evaluation graph or not
pytfl_model_predictions_path = 'output/predictions/S_VAE_O_output.csv'

eval_test = Evaluation(pytfl_model_predictions_path)
eval_1.metric_visualization(max_k, save_graphs)
```

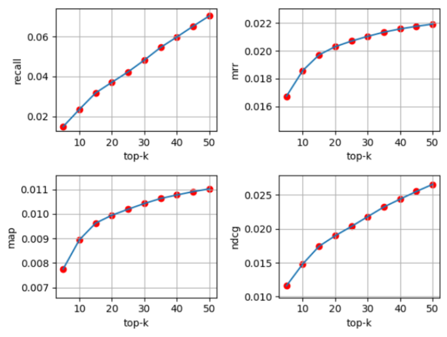


**Figure 3: Code snippet for Evaluation visualization of a Team Formation model.**

## 4 CONCLUSIONS

In this demonstration paper, we showcase the first open-source Python-based toolkit for the task of team formation. The team formation task can find extensive applications ranging from assembling teams for freelancing opportunities to predicting protein-protein interactions. The toolkit is accessible through PyPI and has been designed with extensibility and ease of use in mind. During the demo, we will show the process involved with (1) formatting and loading past collaboration networks in PyTFL, (2) the light-weight training of neural architectures for the sake of team formation, (3) generating teams given input set of skills, as well as evaluating and generating reports on the utility of the generated teams.

## REFERENCES

[1] Aris Anagnostopoulos, Luca Becchetti, Carlos Castillo, Aristides Gionis, and Stefano Leonardi. 2012. Online team formation in social networks. In *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012*, Alain Mille, Fabien Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab (Eds.). ACM, 839–848. https://doi.org/10.1145/2187836.2187950

[2] Aris Anagnostopoulos, Carlos Castillo, Adriano Fazzone, Stefano Leonardi, and Evimaria Terzi. 2018. Algorithms for Hiring and Outsourcing in the Online Labor Market. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Yike Guo and Faisal Farooq (Eds.). ACM, 1109–1118. https://doi.org/10.1145/3219819.3220056

[3] Giorgio Barnabò, Adriano Fazzone, Stefano Leonardi, and Chris Schwiegelshohn. 2019. Algorithms for Fair Team Formation in Online Labour Marketplaces10033. In *Companion of The 2019 World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, Sihem Amer-Yahia, Mohammad Mahdian, Ashish Goel, Geert-Jan Houben, Kristina Lerman, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (Eds.). ACM, 484–490. https://doi.org/10.1145/3308560.3317587

[4] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. 2015. Weight Uncertainty in Neural Networks. (2015). http://arxiv.org/abs/1505.05424

[5] Spencer Bryson, Heidar Davoudi, Lukasz Golab, Mehdi Kargar, Yuliya Lytvyn, Piotr Mierzejewski, Jaroslaw Szlichta, and Morteza Zihayat. 2020. Robust keyword search in large attributed graphs. *Information Retrieval Journal* 23, 5 (2020), 502–524.

[6] Mehdi Kargar, Lukasz Golab, Divesh Srivastava, Jaroslaw Szlichta, and Morteza Zihayat. 2020. Effective Keyword Search over Weighted Graphs. *IEEE Transactions on Knowledge and Data Engineering* early access (2020). https://doi.org/10.1109/TKDE.2020.2985376

[7] Richard M. Karp. 1972. Reducibility Among Combinatorial Problems. In *Proceedings of a symposium on the Complexity of Computer Computations*. Plenum Press, New York, 85–103.

[8] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *KDD*.

[9] Theodoros Lappas, Kun Liu, and Evimaria Terzi. 2009. Finding a team of experts in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, John F. Elder IV, Françoise Fogelman-Soulié, Peter A. Flach, and Mohammed Javeed Zaki (Eds.). ACM, 467–476. https://doi.org/10.1145/1557019.1557074

[10] Radin Hamidi Rad, Ebrahim Bagheri, Mehdi Kargar, Divesh Srivastava, and Jaroslaw Szlichta. 2021. Retrieving Skill-Based Teams from Collaboration Networks. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 5. https://doi.org/10.1145/3404835.3463105

[11] Anna Sapienza, Palash Goyal, and Emilio Ferrara. 2019. Deep Neural Networks for Optimal Team Composition. *Frontiers Big Data* 2 (2019), 14. https://doi.org/10.3389/fdata.2019.00014

[12] Wanyuan Wang, Zhanpeng He, Peng Shi, Weiwei Wu, and Yichuan Jiang. 2016. Truthful Team Formation for Crowdsourcing in Social Networks: (Extended Abstract). In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, May 9-13, 2016*, Catholijn M. Jonker, Stacy Marsella, John Thangarajah, and Karl Tuyls (Eds.). ACM, 1327–1328. http://dl.acm.org/citation.cfm?id=2937143

[13] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *WSDM*. 495–503.

[14] Morteza Zihayat, Aijun An, Lukasz Golab, Mehdi Kargar, and Jaroslaw Szlichta. 2017. Authority-based Team Discovery in Social Networks. In *Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017*, Volker Markl, Salvatore Orlando, Bernhard Mitschang, Periklis Andritsos, Kai-Uwe Sattler, and Sebastian Breß (Eds.). OpenProceedings.org, 498–501. https://doi.org/10.5441/002/edbt.2017.54