AI Summer School 2023

July 3rd – 7th, 2023

Center for Artificial Intelligence and Machine Learning

TU Wien, Vienna, Austria

Table of Contents

Foreword 3
Deep Alternate Clustering 4
Knowledge Graphs in Action: From Foundations to Applications in Finance and Sustainability 9
Artificial Intelligence for Optimization 24
Logic for Declarative Problem-Solving and its Applications 40
Knowledge Engineering for Graph Machine Learning 52
An Exercise-Based Introduction to Ethical Issues of AI 66
Natural Language Processing. An Overview 76
Deep Learning 114
Reinforcement Learning 114
Brain-Inspired Computation and Learning 114

Foreword

The first edition of the ASAI National Summer School in Artificial Intelligence took place from July 3 - July 7, 2023, co-organized by CAIML, TU Wien and LogiCS@TUWien; see also https://caiml.dbai.tuwien.ac.at/events/30/.

The school offered introductory courses in all areas of Artificial Intelligence, including symbolic approaches and machine learning, but also in terms of implications of AI to society. The programme was accompanied by a panel discussion, diversity and inclusion events, as well as the annual meeting of the Austrian Society for Artificial Intelligence (ASAI).

Applicants were expected to be at PhD level or advanced master level with a solid background in Computer Science. We received over 60 applications from over 10 countries, 40 of them have been selected. The school was organised as an in-person event; national and international experts taught in half-day tutorials. To ensure accessibility, the summer school was free of charge. In this collection you can find summaries or abstracts of the tutorials, as well as pointers to further literature.

In particular, this volume contains extended abstract of the following lectures: Lukas Miklautz and Claudia Plant from the University of Vienna introduced the students to advanced clustering techniques; Emanuel Sallinger and Georg Gottlob from TU Wien talked about "Knowledge Graphs in Action"; Lucas Kletzander, Florian Michek and Nysret Musliu from TU Wien gave an overview about the use of AI in optimization problems; Gerhard Friedrich and Martin Gebser (AAU Klagenfurt) discussed the use of logics in declarative problem-solving approaches; knowledge engineering for graph machine learning has been the topic of the lecture by Katja Hose, a newly hired professor at TU Wien; Erich Prem's (eutema GmbH) interactive class on ethical issues of AI finally raised the awareness on pressing topics such as fairness, bias, and freedom of speech. Further lectures (abstracts can be found at the end of this volume) ranged from Deep Learning (Günter Klambauer), Reinforcment Learning (Clemens Heitzinger), and Brain-Inspired Computation and Learning (Robert Legenstein). Based on the lecture by Brigtte Krenn and Johann Petrak (OFAI), this volume also contains an extensive survey on recent methods in Natural Language Processing.

A big thanks goes to all lecturers and to the students for their active participation. We further express our acknowledgements to Carina Zehetmaier, Jana Eder, Gabriele Bolek-Fügl and Diana Silvestru from Women in AI Austria for organizing an excellent session on carrer perspectives as well as to Florian Michahelles (TU Wien), Georg Trausmuth (Frequentis), Lukas Weinwurm (IMMOunited GmbH) and Nysret Musliu (TU Wien) for an insightful panel discussion on the potential of AI in industry and society. This all made the summer school a big success!

Finally, we would like to thank the Austrian Society for Artificial Intelligence (ASAI) and the City of Vienna for the kind financial support.

Deep Alternative Clustering

Lukas Miklautz[1,2][0000−0002−2585−5895] and Claudia Plant[1,3][0000−0001−5274−8123]

[1] Faculty of Computer Science, University of Vienna, Vienna, Austria
[2] UniVie Doctoral School Computer Science
[3] ds:UniVie
{firstname.lastname}@univie.ac.at

Abstract. Clustering algorithms learn to categorize similar objects into similar groups and to keep dissimilar objects apart. In this lecture, we present advanced clustering techniques that can learn multiple alternative clusterings for a data set, leveraging the representation learning capabilities of deep neural networks. Further, we present a case study for finding alternative clusterings of images of archaeological glass beads. For example, the glass beads can be clustered by color, shape, size, or decoration. These are all alternative descriptions of the data that may be of value to archaeologists. We conclude the lecture with an outlook on open problems in the field of deep (alternative) clustering, which should serve as inspiration for future work.

Keywords: Deep Learning · Alternative Clustering · Archaeology · Archaeoinformatics

1 Introduction

Clustering is about finding a natural grouping of objects in data and is deeply rooted in human cognition. Already infants can discriminate objects based on their common characteristics [6] and our brain constantly clusters sensory stimuli in order to recognize, monitor and interpret them. Clustering in data mining and machine learning is about formalizing which objects should be grouped together and which should be kept separate. The lecture comprises a theoretical part that introduces the basic concepts of (deep) alternative clustering and a tutorial style exploration of deep clustering methods for archaeological glass beads using the open-source ClustPy [9] package[4]. The participants should get a broad overview of alternative clustering and unsupervised deep learning techniques with practical coding examples. In the following we explain the main topics of the lecture.

2 Alternative Clustering

Automatically clustering massive data is a challenging research problem for multiple reasons. Sparse, high-dimensional and noisy data push state-of-the-art algorithms to their limits. Moreover, complex data can often be clustered in multiple

[4] https://github.com/collinleiber/ClustPy

meaningful ways. For instance, objects can be clustered by their shape or alternatively by their color. Each grouping represents a different view of the data.

The first part of the lecture introduces solutions for finding multiple alternative clusterings. Starting from the classical K-means algorithm, we show how to find multiple subspaces with alternative K-means clusterings in moderate to high-dimensional data, using the NrKMeans algorithm [12]. A drawback of NrK-Means is that the user needs to specify the number of alternative clusterings a priori, which can be difficult to do in practice. The algorithm AutoNR [8] solves this problem by automatically finding the number of alternative clusterings and the corresponding number of clusters in each clustering using the Minimum Description Length principle (MDL) [19]. MDL allows for information-theoretic parameter estimation and is a popular framework for estimating parameters of clustering algorithms in an unsupervised manner [3–5]. For a detailed overview of alternative clustering algorithms see also [17].

3 Deep Clustering

Deep clustering, also called *representation learning for clustering*, combines ideas from deep learning and clustering to improve clustering performance. Deep clustering algorithms learn so-called "cluster-friendly"[20] representations, that increase the separation between distinct groups. Many recent deep clustering methods have transferred ideas from traditional clustering to the deep learning world. Some examples include methods for automatic estimation of the number of clusters [7] or subspace-centered [13], consensus-based [16], meanshift-based [2] and hierarchical-based [10, 11] deep clustering. For a more detailed overview on recent methods we refer to these surveys [1, 18, 21] and this tutorial [9].[5]

To enable alternative clustering on very high-dimensional data, such as image collections, we integrate the ideas of alternative clustering into deep learning. The algorithm ENRC (**E**mbedded **N**on-**R**edundant **C**lustering) [14] learns individual embedded spaces for each clustering.[6] During the training process, ENRC softly assigns each dimension of the embedded space to the different clusterings and jointly optimizes the clustering and the embedding. Results on image data show that ENRC can group the objects by color, material and shape, without the need for explicit feature engineering. This algorithm thus comes quite close to the goal of automatically discovering multiple natural clusterings.

[5] See also `https://collinleiber.de/deepclustering.html` for additional slides and coding examples of [9].

[6] Here, the term *non-redundant clustering* refers to a special case of alternative clustering in which the algorithm ensures that the found clusterings are non-overlapping.

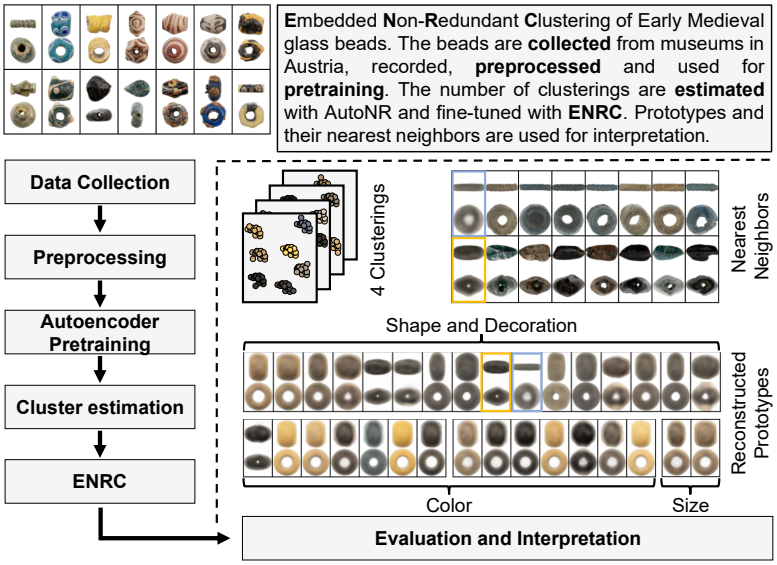


Fig. 1. Workflow described in [15] to automatically find alternative clusterings of Medieval glass beads.

4 Deep Alternative Clustering of Archaeological Glass Beads

In the second part of the lecture, we introduce the result of our methods on real data from archaeology. The motivation for this task is rooted in the interest of archaeologists to objectively study historical artifacts. One large group of archaeological artifacts are glass beads. They were among the most common grave goods in the early Middle Ages and their number is estimated to be in the millions making a manual analysis almost infeasible. The color, size, shape, production technique and decoration of the beads are quite diverse and importantly for our approach can be varied independently of each other. This property makes alternative clusterings methods, like ENRC, suitable. ENRC assumes that the different clusterings of interest are non-redundant, like the clusterings according to colors and shapes of the glass beads. In collaboration with archaeologists from the Austrian Academy of Sciences, we gathered and recorded images of approximately 6,000 beads of the early medieval cemetery of Vienna-Csokorgasse and other burial grounds in Austria. Subsequently, we analysed the images using alternative clustering methods, like AutoNR and ENRC. The results in [15] we present during the lecture are pioneering work as deep clustering methods and alternative clustering methods have not been applied to archaeological glass beads before. In Figure 1 we show an overview of the pipeline used in [15].

5 Conclusion

Deep (alternative) clustering is still a young research field that has many open problems suitable for future work. For example, estimating the parameters for clustering in an end-to-end way, increasing the robustness and stability of results, determining a good trade-off between model complexity and expressiveness or increasing the interpretability of deep clustering methods. We believe our lecture gives a good overview of the capabilities and weaknesses of current deep clustering algorithms and we hope that we can inspire some of the participants to tackle the existing open challenges in the future.

References

1. Aljalbout, E., Golkov, V., Siddiqui, Y., Cremers, D.: Clustering with deep learning: Taxonomy and new methods. CoRR **abs/1801.07648** (2018)
2. Altinigneli, M.C., Miklautz, L., Böhm, C., Plant, C.: Hierarchical quick shift guided recurrent clustering. In: ICDE. pp. 1842–1845. IEEE (2020)
3. Böhm, C., Faloutsos, C., Pan, J., Plant, C.: Robust information-theoretic clustering. In: KDD. pp. 65–75. ACM (2006)
4. Böhm, C., Faloutsos, C., Plant, C.: Outlier-robust clustering using independent components. In: SIGMOD Conference. pp. 185–198. ACM (2008)
5. Böhm, C., Goebl, S., Oswald, A., Plant, C., Plavinski, M., Wackersreuther, B.: Integrative parameter-free clustering of data with mixed type attributes. In: PAKDD (1). Lecture Notes in Computer Science, vol. 6118, pp. 38–47. Springer (2010)
6. Gelman, S.A., Meyer, M.: Child categorization. Wiley Interdisciplinary Reviews: Cognitive Science **2**(1), 95–105 (2011)
7. Leiber, C., Bauer, L.G.M., Schelling, B., Böhm, C., Plant, C.: Dip-based deep embedded clustering with k-estimation. In: KDD. pp. 903–913. ACM (2021)
8. Leiber, C., Mautz, D., Plant, C., Böhm, C.: Automatic parameter selection for non-redundant clustering. In: Proceedings of the 2022 SIAM International Conference on Data Mining (SDM). pp. 226–234. SIAM (2022)
9. Leiber, C., Miklautz, L., Plant, C., Böhm, C.: Application of deep clustering algorithms. In: CIKM. pp. 5208–5211. ACM (2023)
10. Mautz, D., Plant, C., Böhm, C.: Deep embedded cluster tree. In: ICDM. pp. 1258–1263. IEEE (2019)
11. Mautz, D., Plant, C., Böhm, C.: Deepect: The deep embedded cluster tree. Data Sci. Eng. **5**(4), 419–432 (2020)
12. Mautz, D., Ye, W., Plant, C., Böhm, C.: Discovering non-redundant k-means clusterings in optimal subspaces. In: KDD. pp. 1973–1982. ACM (2018)
13. Miklautz, L., Bauer, L.G.M., Mautz, D., Tschiatschek, S., Böhm, C., Plant, C.: Details (don't) matter: Isolating cluster information in deep embedded spaces. In: IJCAI. pp. 2826–2832. ijcai.org (2021)
14. Miklautz, L., Mautz, D., Altinigneli, M.C., Böhm, C., Plant, C.: Deep embedded non-redundant clustering. In: AAAI. vol. 34, pp. 5174–5181 (2020)
15. Miklautz, L., Shkabrii, A., Leiber, C., Tobias, B., Seidl, B., Weissensteiner, E., Rausch, A., Böhm, C., Plant, C.: Non-redundant image clustering of early medieval glass beads. In: DSAA. IEEE (2023)

16. Miklautz, L., Teuffenbach, M., Weber, P., Perjuci, R., Durani, W., Böhm, C., Plant, C.: Deep clustering with consensus representations. In: ICDM. pp. 1119–1124. IEEE (2022)
17. Müller, E., Günnemann, S., Färber, I., Seidl, T.: Discovering multiple clustering solutions: Grouping objects in different views of the data. In: ICDE. pp. 1207–1210 (2012)
18. Ren, Y., Pu, J., Yang, Z., Xu, J., Li, G., Pu, X., Yu, P.S., He, L.: Deep clustering: A comprehensive survey. CoRR **abs/2210.04142** (2022)
19. Rissanen, J.: Modeling by shortest data description. Automatica **14**(5), 465–471 (1978)
20. Yang, B., Fu, X., Sidiropoulos, N.D., Hong, M.: Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In: ICML. vol. 70, pp. 3861–3870. PMLR (2017)
21. Zhou, S., Xu, H., Zheng, Z., Chen, J., Li, Z., Bu, J., Wu, J., Wang, X., Zhu, W., Ester, M.: A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions. CoRR **abs/2206.07579** (2022)

Knowledge Graphs in Action: From Foundations to Applications in Finance and Sustainability*

Emanuel Sallinger[1 [0000−0001−7441−129X]], Jinsong Guo[2], Luigi Bellomarini[3], Georg Gottlob[4,5,1], and Aditya Jami[6]

[1] TU Wien
[2] University College London
[3] Central Bank of Italy
[4] University of Calabria
[5] University of Oxford
[6] Meltwater

Abstract. In this chapter, we are going to see Knowledge Graphs in action. After considering the context of Knowledge Graphs, followed by the theory and systems, we are going to explore and see in action a number of topics, including real-world applications in finance and sustainability, and specifically on competitor data, recycling and decentralized finance.

Keywords: Knowledge graphs · artificial intelligence · Datalog · Vadalog

1 Introduction

In this chapter, we are going to see Knowledge Graphs in action. Motivated by the recent surge of applications [1,4,21,23,24,25,50,43], a number of real-word applications that shall accompany us throughout the chapter, such as from finance and sustainability, we are going to explore and see in action a number of topics:

- The **context**: Knowledge graphs (KGs) have in recent years gained a large momentum both in academic research and in real-world applications. They have become a bridge between databases, artificial intelligence (AI), data science, the (semantic) web and semantic computing, linked data, and many other areas. In particular, in declarative AI, they have become a bridge between logic-based reasoning, and machine learning-based reasoning.
- The **theory** and **systems**: Languages for KGs on the one hand, and systems for KGs i.e., Knowledge Graph Management System (KGMS) on the other hand, have garnered increasing attention. Of particular importance

* This work has been supported by the Vienna Science and Technology Fund (WWTF) [10.47379/ICT2201, 10.47379/VRG18013, 10.47379/NXT22018]; and the Christian Doppler Research Association (CDG) JRC LIVE. Georg Gottlob is a Royal Society Research Professor and acknowledges support by the Royal Society in this role through the "RAISON DATA" project (Reference No. RP/R1/201074). We acknowledge the kind support of Meltwater in this work.

are language and system extensions such as probabilistic reasoning, numeric reasoning, etc. - supporting various real-world applications, and the business applications that can be built using such extensions. We are going to dive into both theory and practice here, including the Vadalog system.

– The real-world **applications**: We focus on seeing Knowledge Graphs in action through a number of real-world and business applications, including finance – also combined with media intelligence and other settings – and sustainability.

Our focus will particularly be on seeing knowledge graphs **in action**. We are going to focus on three applications for this:

– Knowledge graphs for **competitor data** – using KG-based reasoning for inferring competitor relationships between companies
– Knowledge graphs for **sustainability** – using KGs for improving recycling of organig waste, thus supporting a circular economy
– Knowledge graphs for **decentralized finance** – using KGs in the area of derivative contracts, including additional gains in terms of transparency

There are of course many more applications "in action" than we can cover within the limits of this chapter, such as for example web data extraction [42], data wrangling [29,39], crisis response [5,6] and more [32].

This chapter is further structured as follows: First, we will consider briefly the context, namely Knowledge Graphs and AI. Next, we will briefly describe theory and systems, in particular Datalog, Vadalog, and reasoning in general. Finally, we will cover the three applications in one section each. We will finish with a conclusion. We note that this chapter is not primarily intended as a technical introduction, but a way to learn about KGs and see them in action.

2 Context: Knowledge Graphs and Artificial Intelligence

Knowledge Graphs have in recent years become a critical technology that is at the meeting point of databases, data science and artificial intelligence. There are multiple definitions for Knowledge Graphs. In this section, we will take a broad look, describing some of their essential characteristics. Note that this section is not meant as a technical introduction, for this we refer to survey and technical papers (e.g., [35,34,17,18,36]).

Technologically, Knowledge Graphs can be seen by at least these four aspects:

– As a family of **data models** for graph data. In particular, this family includes RDF and its associated technologies as well as property graphs. More widely, this includes any data model able to effectively store graphs.
– Using **logical knowledge** as the "knowledge" in "knowledge graphs". This includes families of logical and rule-based languages, including Datalog, existential rules, etc. More widely, any knowledge representation language is applicable to this.

- Using **Knowledge Graph Embeddings** (KGEs) to represent the latent knowledge of Knowledge Graphs. KGEs are a broad family of machine-learning models, including geometrical models, etc.
- Using **Graph Neural Networks** (GNNs) to represent the latent knowledge of Knowledge Graphs. While there is no absolute division between KGEs and GNNs, one typical distinguishing feature is that KGEs are typically transductive while GNNs are inductive. There is, however, no absolute distinction.

More recently, it has become clear that there is a likely fifth aspect:

- The interaction of **Large Language Models** and KGs. This includes a multitude of techniques, broadly divided into KGs for LLMs, LLMs for KGs and combined approaches.

Finally, we note the connection to AI here: the latter four of the five points we discussed are typically associated with AI techniques, while the first one is typically associated with data management.

3 Theory and Systems: Datalog, Vadalog and Reasoning

In this chapter, we specifically showcase the application of logical knowledge in Knowledge Graphs. Thus, we will give a high-level introduction to the logical languages most used here. This will be a tutorial-style introduction, for a more technical coverage we refer to respective papers (e.g., [50]).

We will specifically frequently use the language **Datalog**, and in particular Datalog rules which formally are based on the language of (full) tuple-generating dependencies, or logical rules of the form

$$\forall \vec{x}, \vec{y}(\varphi(\vec{x}, \vec{y}) \rightarrow \psi(\vec{x}))$$

where φ is a conjunction of relational atoms, and ψ is a single relational atom.

The usual technical notation for Datalog rules is as follows:

```
head(V0) :- body1(V1), body2(V2), ..., bodyN(VN).
```

where `head` and `body1` to `bodyN` are relational atoms, where `V0` to `VN` are lists of variables, and where "`,`" denotes logical conjunction.

This language can expressive *recursion* naturally – e.g., by head and one body atom coinciding. What it lacks is a way of expressing *object creation*, which is added to in the language of existential rules, or tuple-generating dependencies, of the logical form

$$\forall \vec{x}, \vec{y}(\varphi(\vec{x}, \vec{y}) \rightarrow \exists \vec{z} \psi(\vec{x}, \vec{z}))$$

The usual technical notation for such rules is unchanged, i.e., as follows:

```
head(V0,Z0) :- body1(V1), body2(V2), ..., bodyN(VN).
```

with the addition highlighted above in that the `head` atom, in addition to variables `V0` that are a subset of the union of `V1` to `VN`, also contains so-called existentially quantified variables `Z0`.

The language that we are going to use in most of this chapter is called **Vadalog** [7], which is an extension of the above existential rules with a number of features needed in practice. This includes the critical feature of aggregation [14], mathematical operators, etc.

After having introduced the language, the main question is the tasks one wants to solve with it. The typical tasks asked in the context of such logical languages is a family of tasks called **reasoning** (cf. e.g. [49]). One typical such reasoning task is *query answering*, i.e., returning the results of some query not only over some given data, but over data plus the result of applying the rules.

For a formal considerations of this, we refer to the respective papers (e.g., [21,31,22,41]), but we do not that query answering over existential rules is in general undecidable, hence all of the practical languages based on existential rules, including Vadalog, put some mild restrictions over the language to achieve decidability [3].

We remark that this chapter, focused on applications, by design does not go into the very interesting intricacies of the theoretical foundations, such as equivalence [47,28], the notion of limits [38], nesting of rules [37], uncertainty [13], consistency [2], space efficiency [19], etc. For each of these, separate sections of at least this size could be used.

4 Application of Knowledge Graphs on Competitor Data

In this section, we present a practical case study showcasing the application of a rule-based method for inferring new company competitor pairs from existing competitor data stored within a knowledge graph about companies. Additionally, we demonstrate the seamless integration of the rule-based method with other complementary processes, resulting in enhanced outcomes. This section is based on and uses material from the papers [30,33].

In today's global competitive environment, competitor data constitutes information useful to many business applications, such as *Competitive Intelligence*, *Lead Generation*, *Recommender Systems*, and so on. Competition-data sellers usually maintain a manually curated database or knowledge graph containing competitor pairs as part of a companies information system (CIS) that also maintains other useful information about companies such as the industry sectors in which they operate. The Owler[7] knowledge graph (a.k.a., the Owler competitive graph, see

[7] https://corp.owler.com/

the graph on the left of Fig. 1 as an example) is one of the world's largest CIS. It contains data about 16+ million companies crowd-sourced from over 1 million experts. However, competitor relations in such crowd-sourced CIS are naturally incomplete. The **main goal** that we target is to infer new competitor pairs from existing competitor pairs in the Owler CIS.

Datalog programs that perform inference based on knowledge about companies are naturally suitable for inferring new competitor pairs from existing ones in a CIS. In this section, we present CompeGen that applies applies Vadalog [16] (a particular variant of Datalog) rules to infer new competitor pairs from existing ones in the Owler CIS. CompeGen combines its inference process with a "learning" process to acquire some required logical facts and further validates the inference results via an empirical validation process. Fig. 1 illustrates the workflow of CompeGen, which will be detailed in the next two sections.

4.1 Knowledge-based Inference

The inference process of CompeGen uses several different types of knowledge (represented as logical facts) about companies: (*a*) Competitor(A, B, *s*) which represents an existing competitor pair in Owler that a company A has a competitor of B with a proximity score *s* ranging from 0 (extremely unlikely to compete) to 100000 (sure competitors); (*b*) CompSector(C1,S1) which represents the fact that a company identified by the company ID C1 belongs to the industry sector S1; (*c*) CompatSec(S1,S2,*c*) which represents the fact that two sectors S1 and S2 are compatible sectors with a sector compatibility score of *c*. Two sectors, S and S', are compatible if there are more than a certain amount of competitor pairs in each of which one company belongs to S and the other belongs to S' (see Section 4.2). For example, in Figure 1, the sectors S1 and S2 are compatible. Knowledge of types (*a*) and (*b*) comes directly from the Owler CIS while the knowledge about sector compatibilities is learned from existing knowledge in the Owler CIS, which is explained in the next section. Based on such knowledge, CompeGen computes candidate competitor pairs via Vadalog, which is a particular variant of Datalog well-suited for knowledge graphs [16]:

```
Cand(C1,C2,PScore):-Competitor(C1,C2,PScore).
Cand(C1,C3,PScore):-Cand(C1,C2,PS12),Competitor(C2,C3,PS23),
   CompSector(C1,SEC1),CompSector(C3,SEC3),C1!=C3,
   CompatSec(SEC1,SEC3,SeCoScore),Penalty(A), Cutoff(B),
   PScore=max((PS12+PS23-100000-A)*SeCoScore), PScore>B.
```

Each candidate competitor pair (C1,C3) computed by the above program is represented by the fact Cand(C1,C3,PScore) where PScore is a plausibility score expressing a degree of plausibility that company C3 is a competitor of company C1. The above Vadalog program computes a new fact Cand(C1,C3,PScore), either if such a fact is already in the Competitor relation, or if there is an already computed fact Cand(C1,C2,PS12) and a fact Competitor(C2,C3,PS23), where the certain conditions are satisfied. These conditions require that C1 be different from C3, and that the computed plausibility score PScore be larger than some cutoff constant B (represented as Cutoff(B)), where PScore is the maximum value

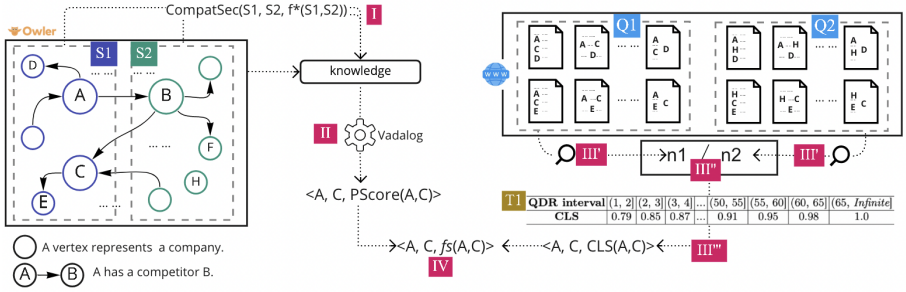


Fig. 1: A simple example of the workflow of CompeGen.

of (PS12 + PS23 − 100000 − A) × $SeCoScore$ (SeCoScore is the compatibility score of the sectors of C1 and of C3) over all matching choices of C2, SEC1, and SEC3. A penalty constant A (represented as Penalty(A)), with $0 < A < 100000$, lowers the proximity scores of new candidate pairs generated by transitivity.

4.2 System Overview

The CompeGen approach can be intuitively explained via the main steps described below.

First, at Step I, a knowledge-learning process is performed to learn the knowledge about sector compatibilities, i.e., sector compatibility scores, from the data in the Owler CIS. Let S be a sector and $S_1, \ldots, S_n$ be all sectors such that there is at least one competitor pair from S and S_i, i.e., there is one edge from S to S_i in the competitive graph, for $1 \leq i \leq n$. For every such S_i, N_i is defined to be the number of edges from S to S_i. Let $N = \max_{1 \leq i \leq n}(N_i)$. Let $c = 0.2$ be some empirically decided cutoff-constant. If $\frac{N_i}{N} < c$ this then means that S_i and S are not compatible, and thus the compatibility weight $f(S, S_i) = 0$. Otherwise, $f(S, S_i)$ is calculated via an empirically determined function $f(S, S_i) = 1 - (1 - \frac{N_i}{N})^m$, where $m = 3$. A smaller m, such as 1, may cause $f(S, S_i)$ to be lower than expected, especially when N is very large while N_i is also large but much smaller than N. For example, when $N = 20000$ and $N_i = 10000$, $f(S, S_i) = 0.5$ if $m = 1$, while $f(S, S_i) = 0.875$ if $m = 3$, and the latter is more reasonable. The compatibility score $f^*(S, S_i)$ is the maximum of the compatibility weights $f(S, S_i)$ and $f(S_i, S)$.

Next, at Step II, candidate competitor pairs, e.g., (A,C) in Fig. 1, are generated via the Vadalog program described in Section 4.1.

The next task (Step III) is to validate each generated candidate competitor pair, e.g., (A,C), against a document repository. The Web is used as the document repository in CompeGen. A Competition Likelihood Score CLS(A,C) of A and C, ranges from 0 (not competitors) to 1 (competitors), is determined based on the co-occurrences of A and C in different Web pages. CLS(A,C) is calculated based on a comparison of a number of search results for two groups of queries to the document repository (Step III'): (i) a first group of queries, for co-occurrences of names of A and of C together with names of some competitors A_i^* of A (if any),

such as D, or some competitors C_i^* of C (if any), such as E. (ii) a second group of queries, corresponding to the first queries, where either A or C is replaced by random companies R(A) or R(C) from the CIS not known to be in a competitor relationship with A or C, such as H. Examples of web pages that match these two groups of queries are in dashed boxes labeled Q1 and Q2, respectively. From the average number of search results of queries in query groups (i) and (ii), denoted by n_1 and n_2, respectively, a Query-result Difference Ratio (QDR) is calculated by n_1/n_2 (Step III'). Based on the QDR, the likelihood score CLS(A,C) is calculated (Step III'') according to a predefined QDR-to-CLS lookup table, such as T1, whereby a higher CLS is achieved if the QDR is larger.

The final part (Step IV) computes for each candidate pair (A,C) a final proximity score via: $fs(A,C) = \frac{PScore(A,C)+CLS(A,C)\times 10^5}{2}$. If (i) $fs(A,C)$ is larger than a given constant (e.g., 90000), and (ii) (A,C) is not already stored in the Owler CIS with a score $s \geq fs(A,C)$ then (A,C) is inserted into the CIS with $fs(A,C)$.

For more detailed introductions to CompeGen and their performance evaluations, we recommend readers refer to the original papers [30,33].

5 Application of Knowledge Graphs on Sustainability

In this section, we present the application of knowledge graphs to the area of sustainability, and more specifically to recycling of organic waste. This is a key part of achieving a circular economy, which is often seen as one of the central parts in achieving a sustainable economy and society.

As the focus in this chapter shall be on the **technologies employed and their scientific foundations**, let us first consider two of the key scientific core points:

- the use of *temporal reasoning* for understanding the time-based patterns critical in recycling
- the use of *neuro-symbolic reasoning*, that is the combination of logic-based artificial intelligence (AI) methods and machine learning-based AI methods

Before exploring them in more detail, let us briefly discuss the extension of Vadalog to Temporal Vadalog [8]. This in particular incorporates the operators of a language called DatalogMTL [20,51], with "MTL" refering to the so-called metric temporal logic.

Let us consider an example of a (simple) temporal rule. Note that we deliberately use simplified rules for the examples, the actual ones are more complex.

```
green(X) :- [-](0,10) organicOnly(X).
```

In the above rule, we see on the left side (the head) the predicate `green`, indicating that at a specific collection point `X` (which can be a particular garbage bin at a house, or other waste collection point), the tour of a particular waste collection truck is to be classified as "green" (i.e., suitable for organic waste collection). On the right hand side we see the determination `organicOnly` which

is a determination whether all the waste collected at a particular collection point `X` consisted of organic waste only. This is, in the real case, determined automatically via computer vision methods, but in our simple example can be determined in any desirable way.

The most interesting part of the example is the operator `<->` and its so-called interval `(0,10)`. What the operator `[-]` indicates is that the term after it holds "always in the past" – the stylized box `[]` indicating the "always" part similar to the modal logic box operator, and the sign "`-`" that it relates to the past. The interval `(0,10)` modifies this by stating that the term should hold "always in the past" but only checking the time interval starting now (0) and extending ten time units (10) into the past. The round brackets indicate an interval that excludes the ending points, i.e., excluding time point 0 and 10.

What this statement thus effectively says is that, if waste at a particular collection points has been determined to be organic only for the last ten time points, then it is classified as a "green" tour point for a waste collection truck. We do not specify here whether the time units refer to time in the physical world, or collection events (i.e., the last 10 collections). Neither do we go into details about the exact nature of time – which can be integer time units, continuous time, etc. We refer the interested reader to the respective papers for more details [20,51,53,52,48,40] and want to particularly point at the modular open benchmark generator iTemporal [15] for readers interested in developing approaches in this area.

Finally, we note that the real-world application consists of a combination of logic-based AI methods (as we have seen examples here so far) and machine learning-based AI methods [9,10,46]. In particular the ML-based AI methods are used for recognizing the type of waste, determining the degree of certainty of such recognitions, etc., based on image and video data.

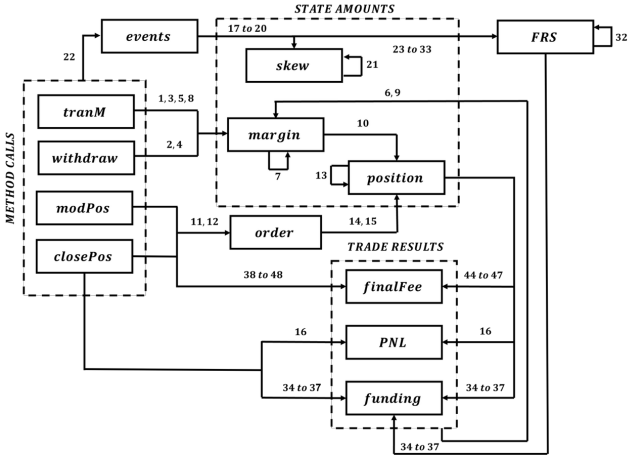


Fig. 2: Dependency graph of the example [26].

6 Application of Knowledge Graphs on Decentralized Finance

In this final application, we will consider decentralized finance. The particular application we focus on in this space is smart contracts representing financial derivatives.

This application is also based on Temporal Vadalog and DatalogMTL, as in the previous application. However, this time we want to consider the bigger picture, especially given that we only considered very simple rules in the previous section.

In Figure 2 we see the overall dependency graph of the Temporal Vadalog / DatalogMTL rules. In this graph, boxes represent predicates and arrows represent rule applications (i.e., the tails of the arrows representing the body or bodies, the heads of the arrows representing heads of Temporal Vadalog / DatalogMTL rules.

The specific rules, for which we refer the interested reader to the respective paper [26], encode the detailed temporal and non-temporal effects of a smart derivative contract.

We still show here some examples to give a feeling for how such rules can look like in this context, such as a simple rule propagating margin over time when no change occurs:

```
margin(A,M) :- <-> margin(A,M), not changeM(A).
```

In the above, we see A as the account, and M as the margin. While we do not attempt to make this a short course on finance and economics, we do note that margin is a concept in finance relating to how much a particular account can have negative balances resp. borrow. In terms of temporal operators, we note that here they are used without intervals, indicating the interval from 0 to infinity (i.e., arbitrary time periods in the past for `[-]` and `<->`). We can also consider a rule that deals with updating margins based on deposits:

```
margin(A,M) :- [-] isOpen(A), <-> margin(A,X),
               tranM(A,Y), M = X+Y.
```

where we have A as account, X as the previous margin, Y as an update to margins, and M as the new margin. Observe that the above rule uses two temporal operators, as well as arithmetic operators.

In summary, and the interested reader can get more details in the respective paper, it is possible to completely model the intricacies of decentralized financial derivatives using a combination of temporal and arithmetic reasoning in Datalog-based Knowledge Graphs.

Finally, we remark that there are multiple interesting aspects of decentralized finance and Knowledge Graphs that were beyond what we presented here, such as

in smart contracts [45,44] or analyzing blockchains with Knowledge Graphs in general [11]. Similarly interesting are applications in traditional (non-decentralized) finance, such as anti-money laundering [12], as well as extending it to other forms of protocols [27].

7 Conclusion

In this chapter, we have seen Knowledge Graphs in action in a number of ways. After determining the context and introducing the theory and systems, we focused on three exemplary applications to see "knowledge graphs in action": first in competitor data, then in sustainability and recycling (focusing on the small-scale of temporal processing) and decentralized finance (focusing on the big-picture of temporal processing).

References

1. Aref, M., ten Cate, B., Green, T.J., Kimelfeld, B., Olteanu, D., Pasalic, E., Veldhuizen, T.L., Washburn, G.: Design and implementation of the LogicBlox system. In: SIGMOD. pp. 1371–1382 (2015)
2. Arming, S., Pichler, R., Sallinger, E.: Complexity of repair checking and consistent query answering. In: Martens, W., Zeume, T. (eds.) 19th International Conference on Database Theory, ICDT 2016, Bordeaux, France, March 15-18, 2016. LIPIcs, vol. 48, pp. 21:1–21:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2016). https://doi.org/10.4230/LIPICS.ICDT.2016.21, https://doi.org/10.4230/LIPIcs.ICDT.2016.21
3. Baldazzi, T., Bellomarini, L., Sallinger, E., Atzeni, P.: Eliminating harmful joins in warded datalog+/-. In: Moschoyiannis, S., Peñaloza, R., Vanthienen, J., Soylu, A., Roman, D. (eds.) Rules and Reasoning - 5th International Joint Conference, RuleML+RR 2021, Leuven, Belgium, September 13-15, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12851, pp. 267–275. Springer (2021). https://doi.org/10.1007/978-3-030-91167-6_18, https://doi.org/10.1007/978-3-030-91167-6_18
4. Barceló, P., Pichler, R. (eds.): Datalog in Academia and Industry, LNCS, vol. 7494. Springer (2012)
5. Bellomarini, L., Benedetti, M., Ceri, S., Gentili, A., Laurendi, R., Magnanimi, D., Nissl, M., Sallinger, E.: Reasoning on company takeovers during the COVID-19 crisis with knowledge graphs. In: Moschoyiannis, S., Fodor, P., Vanthienen, J., Inclezan, D., Nikolov, N., Martín-Recuerda, F., Toma, I. (eds.) Proceedings of the 14th International Rule Challenge, 4th Doctoral Consortium, and 6th Industry Track @ RuleML+RR 2020. CEUR Workshop Proceedings, vol. 2644, pp. 145–156. CEUR-WS.org (2020), https://ceur-ws.org/Vol-2644/paper41.pdf
6. Bellomarini, L., Benedetti, M., Gentili, A., Laurendi, R., Magnanimi, D., Muci, A., Sallinger, E.: COVID-19 and company knowledge graphs: Assessing golden powers and economic impact of selective lockdown via AI reasoning. CoRR **abs/2004.10119** (2020), https://arxiv.org/abs/2004.10119
7. Bellomarini, L., Benedetto, D., Gottlob, G., Sallinger, E.: Vadalog: A modern architecture for automated reasoning with large knowledge graphs. Inf. Syst. **105**, 101528 (2022). https://doi.org/10.1016/J.IS.2020.101528, https://doi.org/10.1016/j.is.2020.101528

8. Bellomarini, L., Blasi, L., Nissl, M., Sallinger, E.: The temporal vadalog system. In: Governatori, G., Turhan, A. (eds.) Rules and Reasoning - 6th International Joint Conference on Rules and Reasoning, RuleML+RR 2022, Berlin, Germany, September 26-28, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13752, pp. 130–145. Springer (2022). https://doi.org/10.1007/978-3-031-21541-4_9, https://doi.org/10.1007/978-3-031-21541-4_9
9. Bellomarini, L., Fayzrakhmanov, R.R., Gottlob, G., Kravchenko, A., Laurenza, E., Nenov, Y., Reissfelder, S., Sallinger, E., Sherkhonov, E., Vahdati, S., Wu, L.: Data science with vadalog: Knowledge graphs with machine learning and reasoning in practice. Future Gener. Comput. Syst. **129**, 407–422 (2022). https://doi.org/10.1016/J.FUTURE.2021.10.021, https://doi.org/10.1016/j.future.2021.10.021
10. Bellomarini, L., Fayzrakhmanov, R.R., Gottlob, G., Kravchenko, A., Laurenza, E., Nenov, Y., Reissfelder, S., Sallinger, E., Sherkhonov, E., Wu, L.: Data science with vadalog: Bridging machine learning and reasoning. In: Abdelwahed, E.H., Bellatreche, L., Golfarelli, M., Méry, D., Ordonez, C. (eds.) Model and Data Engineering - 8th International Conference, MEDI 2018, Marrakesh, Morocco, October 24-26, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11163, pp. 3–21. Springer (2018). https://doi.org/10.1007/978-3-030-00856-7_1, https://doi.org/10.1007/978-3-030-00856-7_1
11. Bellomarini, L., Galano, G., Nissl, M., Sallinger, E.: Rule-based blockchain knowledge graphs: Declarative AI for solving industrial blockchain challenges. In: Soylu, A., Tamaddoni-Nezhad, A., Nikolov, N., Toma, I., Fensel, A., Vennekens, J. (eds.) Proceedings of the 15th International Rule Challenge, 7th Industry Track. CEUR Workshop Proceedings, vol. 2956. CEUR-WS.org (2021), https://ceur-ws.org/Vol-2956/paper60.pdf
12. Bellomarini, L., Laurenza, E., Sallinger, E.: Rule-based anti-money laundering in financial intelligence units: Experience and vision. In: Moschoyiannis, S., Fodor, P., Vanthienen, J., Inclezan, D., Nikolov, N., Martín-Recuerda, F., Toma, I. (eds.) Proceedings of the 14th International Rule Challenge. CEUR Workshop Proceedings, vol. 2644, pp. 133–144. CEUR-WS.org (2020), https://ceur-ws.org/Vol-2644/paper40.pdf
13. Bellomarini, L., Laurenza, E., Sallinger, E., Sherkhonov, E.: Reasoning under uncertainty in knowledge graphs. In: Gutiérrez-Basulto, V., Kliegr, T., Soylu, A., Giese, M., Roman, D. (eds.) Rules and Reasoning - 4th International Joint Conference, RuleML+RR 2020, Oslo, Norway, June 29 - July 1, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12173, pp. 131–139. Springer (2020). https://doi.org/10.1007/978-3-030-57977-7_9, https://doi.org/10.1007/978-3-030-57977-7_9
14. Bellomarini, L., Nissl, M., Sallinger, E.: Monotonic aggregation for temporal datalog. In: Soylu, A., Tamaddoni-Nezhad, A., Nikolov, N., Toma, I., Fensel, A., Vennekens, J. (eds.) Proceedings of the 15th International Rule Challenge, 7th Industry Track, and 5th Doctoral Consortium @ RuleML+RR 2021 co-located with 17th Reasoning Web Summer School (RW 2021) and 13th DecisionCAMP 2021 as part of Declarative AI 2021, Leuven, Belgium (virtual due to Covid-19 pandemic), 8 - 15 September, 2021. CEUR Workshop Proceedings, vol. 2956. CEUR-WS.org (2021), https://ceur-ws.org/Vol-2956/paper30.pdf
15. Bellomarini, L., Nissl, M., Sallinger, E.: itemporal: An extensible generator of temporal benchmarks. In: 38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022. pp. 2021–2033. IEEE (2022).

https://doi.org/10.1109/ICDE53745.2022.00197, https://doi.org/10.1109/ICDE53745.2022.00197
16. Bellomarini, L., Sallinger, E., Gottlob, G.: The vadalog system: datalog-based reasoning for knowledge graphs. Proceedings of the VLDB Endowment **11**(9), 975–987 (2018)
17. Bellomarini, L., Sallinger, E., Vahdati, S.: Knowledge graphs: The layered perspective. In: Janev, V., Graux, D., Jabeen, H., Sallinger, E. (eds.) Knowledge Graphs and Big Data Processing, Lecture Notes in Computer Science, vol. 12072, pp. 20–34. Springer (2020). https://doi.org/10.1007/978-3-030-53199-7_2, https://doi.org/10.1007/978-3-030-53199-7_2
18. Bellomarini, L., Sallinger, E., Vahdati, S.: Reasoning in knowledge graphs: An embeddings spotlight. In: Janev, V., Graux, D., Jabeen, H., Sallinger, E. (eds.) Knowledge Graphs and Big Data Processing, Lecture Notes in Computer Science, vol. 12072, pp. 87–101. Springer (2020). https://doi.org/10.1007/978-3-030-53199-7_6, https://doi.org/10.1007/978-3-030-53199-7_6
19. Berger, G., Gottlob, G., Pieris, A., Sallinger, E.: The space-efficient core of vadalog. In: Suciu, D., Skritek, S., Koch, C. (eds.) Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019. pp. 270–284. ACM (2019). https://doi.org/10.1145/3294052.3319688, https://doi.org/10.1145/3294052.3319688
20. Brandt, S., Kalaycı, E.G., Ryzhikov, V., Xiao, G., Zakharyaschev, M.: Querying log data with metric temporal logic. J. Artif. Intell. Res. pp. 829–877 (2018)
21. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. J. Artif. Intell. Res. **48**, 115–174 (2013)
22. Calì, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. In: PODS. pp. 77–86 (2009)
23. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. J. Web Sem. **14**, 57–83 (2012)
24. Calì, A., Gottlob, G., Lukasiewicz, T., Marnette, B., Pieris, A.: Datalog+/-: A family of logical knowledge representation and query languages for new applications. In: LICS. pp. 228–242 (2010)
25. Calì, A., Gottlob, G., Pieris, A.: Towards more expressive ontology languages: The query answering problem. Artificial Intelligence **193**, 87–128 (2012)
26. Colombo, A., Bellomarini, L., Ceri, S., Laurenza, E.: Smart derivative contracts in datalogmtl. In: Stoyanovich, J., Teubner, J., Mamoulis, N., Pitoura, E., Mühlig, J., Hose, K., Bhowmick, S.S., Lissandrini, M. (eds.) Proceedings 26th International Conference on Extending Database Technology, EDBT 2023, Ioannina, Greece, March 28-31, 2023. pp. 773–781. OpenProceedings.org (2023). https://doi.org/10.48786/EDBT.2023.65, https://doi.org/10.48786/edbt.2023.65
27. Csar, T., Lackner, M., Pichler, R., Sallinger, E.: Winner determination in huge elections with mapreduce. In: Singh, S., Markovitch, S. (eds.) Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA. pp. 451–458. AAAI Press (2017). https://doi.org/10.1609/AAAI.V31I1.10606, https://doi.org/10.1609/aaai.v31i1.10606
28. Feinerer, I., Pichler, R., Sallinger, E., Savenkov, V.: On the undecidability of the equivalence of second-order tuple generating dependencies. Inf. Syst. **48**, 113–129 (2015). https://doi.org/10.1016/J.IS.2014.09.003, https://doi.org/10.1016/j.is.2014.09.003

29. Furche, T., Gottlob, G., Neumayr, B., Sallinger, E.: Data wrangling for big data: Towards a lingua franca for data wrangling. In: Pichler, R., da Silva, A.S. (eds.) Proceedings of the 10th Alberto Mendelzon International Workshop on Foundations of Data Management, Panama City, Panama, May 8-10, 2016. CEUR Workshop Proceedings, vol. 1644. CEUR-WS.org (2016), https://ceur-ws.org/Vol-1644/paper20.pdf
30. Gottlob, G., Guo, J., Jami, A., Kröll, M., Reissfelder, S., Schweizer, L., Aichinger, E., Sferrazza, S.: Compege: Computing company competitor pairs by knowledge based inference combined with empirical validation (2022)
31. Gottlob, G., Pieris, A.: Beyond sparql under owl 2 ql entailment regime: Rules to the rescue. In: Proceedings of the 24th International Conference on Artificial Intelligence. p. 2999–3007. IJCAI'15, AAAI Press (2015)
32. Gottlob, G., Pieris, A., Sallinger, E.: Vadalog: Recent advances and applications. In: Calimeri, F., Leone, N., Manna, M. (eds.) Logics in Artificial Intelligence - 16th European Conference, JELIA 2019, Rende, Italy, May 7-11, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11468, pp. 21–37. Springer (2019). https://doi.org/10.1007/978-3-030-19570-0_2, https://doi.org/10.1007/978-3-030-19570-0_2
33. Guo, J., Jami, A., Kröll, M., Schweizer, L., Paramonov, S., Aichinger, E., Sferrazza, S., Scaccia, M., Reissfelder, S., Cicek, E., et al.: When automatic filtering comes to the rescue: Pre-computing company competitor pairs in owler. Proceedings of the ACM on Management of Data **1**(2), 1–23 (2023)
34. Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., Ngomo, A.N., Polleres, A., Rashid, S.M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., Zimmermann, A.: Knowledge Graphs. Synthesis Lectures on Data, Semantics, and Knowledge, Morgan & Claypool Publishers (2021). https://doi.org/10.2200/S01125ED1V01Y202109DSK022
35. Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., Ngomo, A.N., Polleres, A., Rashid, S.M., Rula, A., Schmelzeisen, L., Sequeda, J.F., Staab, S., Zimmermann, A.: Knowledge graphs. ACM Comput. Surv. **54**(4), 71:1–71:37 (2022). https://doi.org/10.1145/3447772, https://doi.org/10.1145/3447772
36. Janev, V., Graux, D., Jabeen, H., Sallinger, E. (eds.): Knowledge Graphs and Big Data Processing, Lecture Notes in Computer Science, vol. 12072. Springer (2020). https://doi.org/10.1007/978-3-030-53199-7, https://doi.org/10.1007/978-3-030-53199-7
37. Kolaitis, P.G., Pichler, R., Sallinger, E., Savenkov, V.: Nested dependencies: structure and reasoning. In: Hull, R., Grohe, M. (eds.) Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014. pp. 176–187. ACM (2014). https://doi.org/10.1145/2594538.2594544, https://doi.org/10.1145/2594538.2594544
38. Kolaitis, P.G., Pichler, R., Sallinger, E., Savenkov, V.: Limits of schema mappings. Theory Comput. Syst. **62**(4), 899–940 (2018). https://doi.org/10.1007/S00224-017-9812-7, https://doi.org/10.1007/s00224-017-9812-7
39. Konstantinou, N., Abel, E., Bellomarini, L., Bogatu, A., Civili, C., Irfanie, E., Koehler, M., Mazilu, L., Sallinger, E., Fernandes, A.A.A., Gottlob, G., Keane, J.A., Paton, N.W.: VADA: an architecture for end user informed data preparation.

J. Big Data **6**, 74 (2019). https://doi.org/10.1186/S40537-019-0237-9, https://doi.org/10.1186/s40537-019-0237-9

40. Lanzinger, M., Nissl, M., Sallinger, E., Walega, P.A.: Temporal datalog with existential quantification. In: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China. pp. 3277–3285. ijcai.org (2023). https://doi.org/10.24963/IJCAI.2023/365, https://doi.org/10.24963/ijcai.2023/365
41. Leone, N., Manna, M., Terracina, G., Veltri, P.: Efficiently computable $datalog^{\exists}$ programs. In: Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning. p. 13–23. KR'12, AAAI Press (2012)
42. Michels, C., Fayzrakhmanov, R.R., Ley, M., Sallinger, E., Schenkel, R.: Oxpath-based data acquisition for dblp. In: 2017 ACM/IEEE Joint Conference on Digital Libraries, JCDL 2017, Toronto, ON, Canada, June 19-23, 2017. pp. 319–320. IEEE Computer Society (2017). https://doi.org/10.1109/JCDL.2017.7991609, https://doi.org/10.1109/JCDL.2017.7991609
43. Moustafa, W.E., Papavasileiou, V., Yocum, K., Deutsch, A.: Datalography: Scaling datalog graph analytics on graph processing systems. In: BigData. pp. 56–65. IEEE (2016)
44. Nissl, M., Sallinger, E.: Modelling smart contracts with datalogmtl. In: Ramanath, M., Palpanas, T. (eds.) Proceedings of the Workshops of the EDBT/ICDT 2022 Joint Conference, Edinburgh, UK, March 29, 2022. CEUR Workshop Proceedings, vol. 3135. CEUR-WS.org (2022), https://ceur-ws.org/Vol-3135/EcoFinKG_2022_paper4.pdf
45. Nissl, M., Sallinger, E.: Towards bridging traditional and smart contracts with datalog-based languages. In: Alviano, M., Pieris, A. (eds.) Proceedings of the 4th International Workshop on the Resurgence of Datalog in Academia and Industry (Datalog-2.0 2022). CEUR Workshop Proceedings, vol. 3203, pp. 68–82. CEUR-WS.org (2022), https://ceur-ws.org/Vol-3203/paper5.pdf
46. Pavlovic, A., Sallinger, E.: Expressive: A spatio-functional embedding for knowledge graph completion. In: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net (2023), https://openreview.net/pdf?id=xkev3_np08z
47. Pichler, R., Sallinger, E., Savenkov, V.: Relaxed notions of schema mapping equivalence revisited. Theory Comput. Syst. **52**(3), 483–541 (2013). https://doi.org/10.1007/S00224-012-9397-0, https://doi.org/10.1007/s00224-012-9397-0
48. Ryzhikov, V., Wałęga, P.A., Zakharyaschev, M.: Data complexity and rewritability of ontology-mediated queries in metric temporal logic under the event-based semantics. In: Proc. of IJCAI. pp. 1851–1857 (2019)
49. Sallinger, E.: Reasoning about schema mappings. In: Kolaitis, P.G., Lenzerini, M., Schweikardt, N. (eds.) Data Exchange, Integration, and Streams, Dagstuhl Follow-Ups, vol. 5, pp. 97–127. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2013). https://doi.org/10.4230/DFU.VOL5.10452.97, https://doi.org/10.4230/DFU.Vol5.10452.97
50. Vianu, V.: Datalog unchained. In: PODS. pp. 57–69. ACM (2021)
51. Wałęga, P.A., Cuenca Grau, B., Kaminski, M., Kostylev, E.V.: DatalogMTL: Computational complexity and expressive power. In: Proc. of IJCAI. pp. 1886–1892 (2019)
52. Wałęga, P.A., Cuenca Grau, B., Kaminski, M., Kostylev, E.V.: DatalogMTL over the integer timeline. In: Proc. of KR. pp. 768–777 (2020)

53. Wałęga, P.A., Cuenca Grau, B., Kaminski, M., Kostylev, E.V.: Tractable fragments of Datalog with metric temporal operators. In: Proc. of IJCAI. pp. 1919–1925 (2020)

Artificial Intelligence for Optimization

Lucas Kletzander[0000−0002−2100−7733], Florian Mischek[0000−0003−1166−3881], and Nysret Musliu[0000−0002−3992−8637]

Christian Doppler Laboratory for AI and Optimization for Planning and Scheduling, DBAI, TU Wien, Vienna, Austria
{firstname.lastname}@tuwien.ac.at

Abstract. Optimization problems arise in a variety of areas in industry, business, engineering, health care, etc. In this tutorial, we will give an overview of different AI methods and application areas/problems, such as planning and scheduling, timetabling, and other optimization problems. In the first part of the tutorial, we will discuss various case studies and methods based on AI techniques for solving such problems. These topics include solver-independent modeling, constraint programming strategies, heuristic methods, and hybrid techniques. In the second part of the tutorial, we will present methods that use machine learning techniques for automatic algorithm selection and heuristic algorithm design. We will demonstrate the application of the presented techniques on several real-world application domains.

Keywords: Artificial Intelligence · Constraint Programming · Heuristic Methods · Hybrid Methods · Algorithm Selection.

1 Introduction

Optimization problems arise in a variety of areas of industry, business, engineering, healthcare, etc. Such problems are often very challenging and their solutions impact the people involved as well as the efficiency and organizational cost of operations.

Researchers in the fields of artificial intelligence and operations research and others have proposed a range of techniques to solving optimization problems. These include constraint programming (CP) [42], answer set programming (ASP) [3], metaheuristics [15], SAT (Boolean Satisfiability Problem) [2], integer programming (IP) [16], and hybrid techniques [53, 50].

In this tutorial, we focus on several optimization problems that arise in the area of planning and scheduling. We will give a partial overview of our work in the Christian Doppler Laboratory for Artificial Intelligence and Optimization for Planning and Scheduling (CD-Lab ARTIS) for solving diverse planning and scheduling problems. Such problems include personnel planning and scheduling problems, test laboratory scheduling, and production planning problems. We will briefly describe our main approaches to solving these problems, such as CP, metaheuristics, and hybrid techniques. In the second part of the tutorial, we will

give a short introduction to algorithm selection, instance space analysis, and hyper-heuristics. This part also describes some of our work in CD-Lab ARTIS on these topics. The text in the following sections is based on our selected papers, which are cited in these sections.

2 Constraint Programming Approaches

In CD-Lab ARTIS, we have been utilizing constraint programming for several real-world applications. Below, we present a selection of three papers.

In [23] we deal with the extended rotating workforce scheduling (RWS) problem. Due to high practical relevance various versions of employee scheduling problems have been investigated for several decades. The rotating workforce scheduling problem can be classified as a single-activity tour scheduling problem with non-overlapping shifts and rotation constraints and is known to be NP-complete.

The problem has been addressed with a range of different methods. A complete method based on CP for standard RWS was introduced by [35]. It uses a solver independent formulation in the MiniZinc constraint language [37], either with a direct representation or using a regular automaton, and applies both the lazy clause generation solver Chuffed and the MIP solver Gurobi. It was the first complete method able to solve the standard benchmark set of 20 instances.

Existing work on RWS mostly deals with the standard version of the problem, requiring any feasible solution, or delegates the selection of preferred solutions to the user in an interactive process. While standard RWS already has practical relevance, the extensions allow to deal with more complex issues and provide solutions that are of higher value in real-life applications.

In several applications it can be beneficial to obtain a rotating schedule where each employee rotates through the same sequence of shifts and days off across several weeks, however, at different offsets within the rotation. As the design of shift schedules highly influences the work-life balance of the employees, such problems are subject to a wide range of constraints, dealing not only with the demand for employees in different shifts, but also legal and organizational constraints that determine allowed shift assignments.

The contributions of this work are twofold. We introduce and solve a new extended problem that includes several new additions based on the experience from working with these problems in practice. Extensions include new constraints for fast detection of infeasible instances, complex constraints to respect weekly rest times, as well as soft constraints optimizing free weekends in the schedule, turning the satisfaction problem into an optimization problem. To solve the problem we provide a new constraint model and implement it in the constraint modelling language MiniZinc.

Both the core model and the extended models are then evaluated on a standard set of benchmark instances based on real-life examples using the constraint solver Chuffed and compared to recent literature. The results show that the extended models greatly improve the handling of infeasible instances and allow

to incorporate complex rest time constraints and optimization goals providing optimal solutions for the majority of the benchmark instances in short computational time. With our work we further improve the state-of-the-art solver for rotating workforce scheduling and enable this solver to be used in more complex real-life situations.

The paper [12] presents a work which was done at the beginning of the pandemic in 2020. We consider a real-world problem occurring in the St. Anna Children's Hospital in Vienna, where the normal scheduling procedures had to be rapidly adapted due to an ongoing pandemic. In addition to ordinary operational requirements, we also need to take additional constraints about infection risk into account. For example, it would be unwise for doctors to change their assigned stations frequently, as they would risk coming into contact with more of their colleagues and more patients. Also, an important goal is to maintain a low total number of working doctors to reduce unnecessary infection risk and keep some physicians in reserve to cover for colleagues in quarantine.

Recently, the emergency scenarios during the pandemic have been considered in the pharmaceutical industry and for nurse rostering. However, in this paper we focus on a new physician scheduling problem, which to the best of our knowledge includes unique features and pandemic related constraints.

To produce schedules that can deal with these new and complex requirements, we developed a CP model that allowed us to quickly prototype a model and adapt it to changing requirements. CP approaches have been applied for other physician scheduling and related problems, although without consideration of pandemic-related constraints. We also included several redundant constraints to the model and evaluated it on several state-of-the-art CP and mixed integer programming (MIP) solvers. We were able to quickly find high-quality solutions under several different configurations. Our model was used to produce the schedule for the physicians in two hospital wards of the St. Anna Children's hospital. We show that the additional constraints can help in reducing the contacts both between physicians and with their patients, compared to work schedules generated under normal conditions.

In [14] we investigate solving a real-world project scheduling problem that arises in an industrial test laboratory. Industrial Test Laboratory Scheduling (TLSP) [28] is an extension of the well known Resource-Constrained Project Scheduling Problem (RCPSP). It consists of a grouping stage, where smaller activities (tasks) are joined into larger jobs, and a scheduling stage, where those jobs are scheduled and have resources assigned to them. In this work, we deal with the second stage and assume that a grouping of tasks into jobs is already provided. Since we focus on the scheduling part, we denote the resulting problem TLSP-S.

The investigated problem has several features of previous project scheduling problems in the literature, but also includes some specific features imposed by the real-world situation, which have rarely been studied before. Among others, these include heterogeneous resources, with availability restrictions on the activities each unit of a resource can perform. While work using similar restrictions

exists, most problem formulations either assume homogeneous, identical units of each resource or introduce additional activity modes for each feasible assignment, which quickly becomes impractical for higher resource requirements and multiple resources. Another specific feature of TLSP(-S) is that of linked activities, which require identical assignments on a subset of the resources. We also deal with several non-standard objectives instead of the usual makespan minimization, which arise from various business objectives of our industrial partner. Most notably, we try to minimize the total completion time of each project, i.e. the time between the start of the first and the end of the last job in the project.

In practice, exact solutions for this problem are desired especially in situations where it is necessary to check if a feasible solution exists at all. In the application that we consider, checking quickly if activities of additional projects can be added on top of an existing schedule is very important. In this paper we investigate exact methods for solving this problem.

We provide a CP model for our problem by exploiting some previous ideas for a similar problem from [51, 62] and extend it to model the additional features of TLSP-S. This includes, for example, the handling of the problem specific differences discussed above but also new redundant constraints as well as search procedures tailored to the problem. Using the MiniZinc [37] constraint programming language we experiment with various strategies involving the formulation of resource constraints, the reduction of the search space, and search procedures based on heuristics.

Our final experiments show that constraint programming techniques can reach very good results for realistic instances and outperform MIP solvers on the same model. Our results strengthen the conclusion of previous studies and show that CP technology can be applied successfully for solving large project scheduling problems.

In addition to the papers presented in this section, other works within CD-Lab explore the application of constraint programming to various other problems, such as Paint Shop Scheduling [56], Parallel Machine Scheduling with Contamination Constraints [55], and Oven Scheduling [25].

3 Metaheuristics and Hybrid Approaches

In our lab we have been utilizing metaheuristics and hybrid methods for several real-world applications. Below, we present a selection of three papers.

In [31] we consider the Test Laboratory Scheduling Problem (TLSP). A restricted version of the problem, called TLSP-S, has been tackled by [28] and [14]. We investigate the possibility of using a local search approach for the general problem. To this aim, we develop four new complex neighborhoods that modify the grouping and combine them with neighborhoods affecting the schedule of the jobs. The general idea is that the two components of the problem are solved simultaneously in a cooperative fashion.

As metaheuristics for guiding the search, we experiment with the Min-Conflicts heuristic (MC) [26] and Simulated Annealing (SA) [18]. On top of both of them,

we also design an iterated local search (ILS) procedure that interleaves the underlying metaheuristic with perturbation steps. As a result, we have four candidate solution methods, namely MC and SA both with and without the ILS perturbations.

All four methods, properly tuned using a statistically-principled tuning procedure, are compared among each other and with the results of [9] and [28] on a dataset composed of artificial and real-world instances. All instances used for this evaluation are publicly available for download.

The general outcome is that we find high-quality solutions even without the benefit of a known good grouping as in the TLSP-S, which are competitive with those of [28]. Comparing to the state-of-the-art solver for the TLSP by [9], we improve results on several instances. We see that we are able to obtain better results in particular on large instances, or under tight time limits. The algorithms described in this paper are used successfully in the daily scheduling of our industrial partner's laboratory.

In [19] we investigated a driver scheduling problem. When scheduling drivers for public transport, in addition to covering the demand and dealing with the spatial dimension, a range of legal requirements, collective agreements and company policies need to be respected. The level of concentration required while driving leads to strict rules for break assignments. This results in a complex problem where creating cost-efficient and employee-friendly schedules is challenging. This paper deals with bus driver scheduling using the rules of the Austrian collective agreement for private omnibus providers. The contributions are the formalization of the complex Austrian rules for bus drivers, a new set of publicly available instances based on the characteristics of real-life instances, and a metaheuristic solution approach for the problem. The algorithm was able to significantly improve the solutions of real-life instances and is evaluated on the generated instances. Further we provide insight in the necessity of objectives for employee satisfaction and their effects. Our method can even be successfully applied to improve results on a problem with very different constraints from Brasil.

In [8, 9] we present an AI system for automated scheduling in test laboratories. We introduce an innovative scheduling system that allows the efficient and flexible generation of schedules for TLSP. It features a new Constraint Programming model that covers both the grouping and the scheduling aspect, as well as a hybrid Very Large Neighborhood Search that internally uses the CP model. Our experimental results on generated and real-world benchmark instances show that good results can be obtained even compared to settings which have a good grouping already provided, including several new best known solutions for these instances. Our algorithms for TLSP have been successfully implemented in a real-world industrial test laboratory. We provide a detailed description of the deployed system as well as additional useful soft constraints supported by the solvers and general lessons learned. This includes a discussion of the choice of soft constraint weights, with an analysis on the impact and relation of different objectives to each other. Our experiments show that some soft constraints com-

plement each other well, while others require explicit trade-offs via their relative weights.

In addition to the papers presented in this section, other works within CD-Lab explore the application of heuristic techniques and hybrid methods to various other problems, such as Paint Shop Scheduling [57, 59], General Employee Scheduling [20], Parallel Machine Scheduling [32], and Production Leveling Problem [54].

4 Automated Algorithm Selection

Whenever multiple solution methods are available to tackle a problem, the question arises which algorithm to use to solve a given problem instance. While some methods might entirely outperform others on all instances, typically there is not a single method that is best on all instances. This fact has been formalized as the "No free lunch theorem", stating that "... for any algorithm, any elevated performance over one class of problems is offset by performance over another class" [60], or in other words that "... any two algorithms are equivalent when their performance is averaged across all possible problems" [61].

However, the problems we are interested in typically have some structure, and based on this structure we might be able to decide for each instance, which algorithm might be best suited to solve this particular instance. This core idea of algorithm selection has been formalized by John R. Rice [41] already back in 1976.

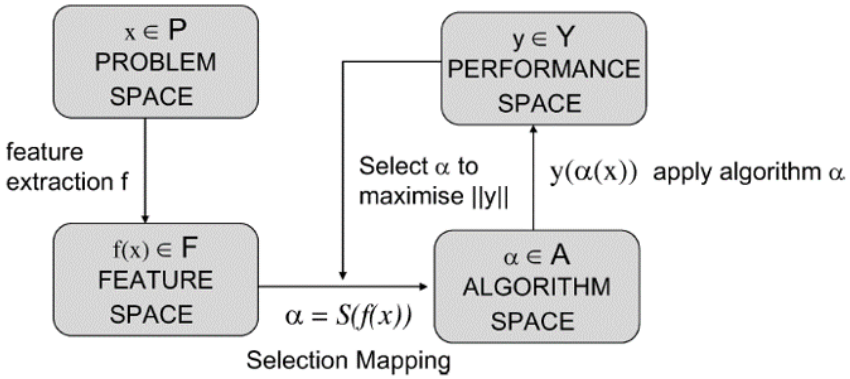


Fig. 1: Rice's framework as shown in [48]

The main components in the framework are:

– A problem space P representing a set of instances that contains an instance x.

- A feature space F that contains measurable characteristics of the instances generated by a computational feature extraction process applied to $x \in P$. Measurable means that these features need to be quantifiable by numbers or at least categories in a well-defined manner. The computational feature extraction process has to be computable in reasonable time - if the time it takes to extract features is longer than the potential improvement of choosing a better algorithm, it is not worth the effort. Features can include different aspects of an instance like instance size, instance structure, or probing features that run an algorithm for a short time and report some metrics of this run. See [43, 36] for a case study with 78 different features on graphs.
- A is the algorithm space containing the set of solution method for our problem.
- Y is the performance space allowing to measure the performance of an algorithm α on an instance x by $y(\alpha(x))$. This measure can be solution quality, runtime to find a feasible or optimal solution, a combination of these, or any other measurable aspect of algorithm execution.

The goal of algorithm selection methods is now to learn a mapping $S(f(x))$, based on the performance of algorithms in A according to the performance metric y such that this performance metric is maximized by choosing the best suited algorithm $\alpha \in A$ for each instance $x \in P$ based on the features $f(x)$ of this instance.

Typically the most complex part of this process is finding features that capture what makes an instance of a problem hard or easy to solve for a particular problem. In order to learn the selection mapping, any supervised machine learning technique can be used, including methods like Bayesian Networks, Decision Trees, k-Nearest Neighbor, Random Forests, Multilayer Perceptrons, Support Vector Machines, or Deep Neural Networks. It is important that an appropriately large training set is used to train this selection mapping.

Since finding adequate features is one of the main challenges in algorithm selection, there are also recent research directions about algorithm selection without features, e.g., by directly using instance data as time series for Recurrent Neural Networks [1].

5 Instance Space Analysis

However, there might be more questions to answer than selecting an algorithm for a particular instance. When evaluating a new method for a given problem, the basic research question of how to judge the performance of this algorithm needs to be answered. Usually evaluation is done on a set of benchmark instances. Is it enough to be better in the average? What about only being better in certain cases? Do the benchmark instances even cover all interesting areas of potential instances? How can we check our instances and features to make sure that we can properly identify strengths and weaknesses of different algorithms?

An answer to these questions is provided by Instance Space Analysis (ISA), a methodology developed by Smith-Miles and co-workers [44, 45, 33, 47] in re-

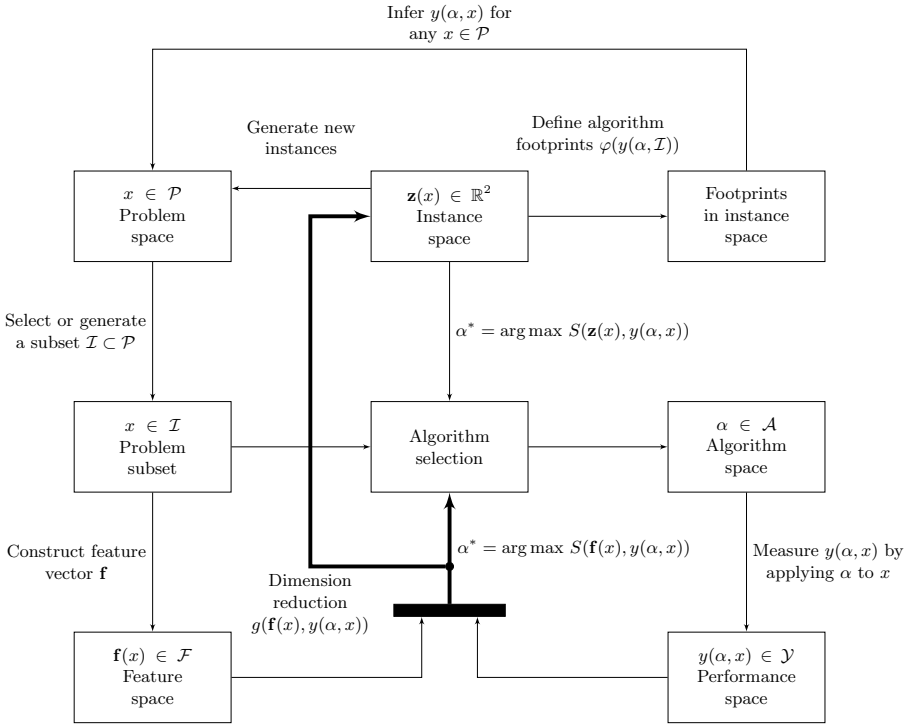


Fig. 2: ISA framework [47] extending the original by Rice [41]

cent years, by extending the algorithm selection problem framework of Rice. Instances are again represented as a feature vector that captures the intrinsic difficulty of instances for various algorithms (or models or parameter settings). By constructing a 2-d projection of a feature-vector representation of instances, ISA allows us to:

1. visualize the distribution and diversity of existing benchmark instances;
2. assess the adequacy of the features;
3. identify and measure the algorithm's regions of strength *footprint* and weaknesses; and
4. distinguish areas of the space where it may be useful to generate additional instances to support greater insights.

Figure 2 illustrates the framework and its component spaces. On the top left is the ill-defined problem space $\mathcal{P}$, which contains all the relevant problems to be solved. However, we only have computational results for a subset $\mathcal{I}$, which is modelled separately in ISA. Just like before, further components are the feature space $\mathcal{F}$, in the bottom left, the algorithm space $\mathcal{A}$ on the center right, and the performance space, $\mathcal{Y}$, below $\mathcal{A}$.

The meta-data, composed of the features and algorithm performance for all the instances in $\mathcal{I}$, is used to learn the mapping $\mathbf{z}(x) = g(\mathbf{f}(x), y(\alpha, x))$

that projects an instance x from a high-dimensional feature space to a two-dimensional space, which we call the instance space (bold arrow). Now algorithm selection can be performed, trying to learn a mapping S to find the best algorithm α^* for a given instance x, either on the feature vector $\mathbf{f}(x)$ (as before), or on the projection to the instance space $\mathbf{z}(x)$. The instance space can further be used to define footprints (regions of strength) for the algorithms, which can be used to infer the performance $y(\alpha, x)$ of an algorithm α on an unseen problem instance $x \in \mathcal{P}$. The information from the instance space can also be used to identify where new instances are needed to augment $\mathcal{I}$.

In earlier work, the projection $\mathbf{z}$ was achieved using principal component analysis, and applied to problems as diverse as graph colouring [44], time series forecasting [17], and software test case generation methods [39]. In an application to machine learning algorithms [34] a customized projection algorithm was developed to obtain an optimal projection that aims to expose linear trends in both features and algorithm performance to aid interpretability. The latest version of the evolving methodology can be found in a more recent publication [47]. While the ISA methodology is broadly applicable, it needs to be customised through careful choice of instance features and an understanding of what makes the problem hard [46].

Instance Space Analysis is available via a web interface[1], and the code is directly available on Github[2].

We regularly apply ISA on problems with multiple solution methods, or to investigate and extend benchmark sets, including work on Rotating Workforce Scheduling [24], Job Shop Scheduling [49], Course Timetabling [10], and the Generalized Assignment Problem [13].

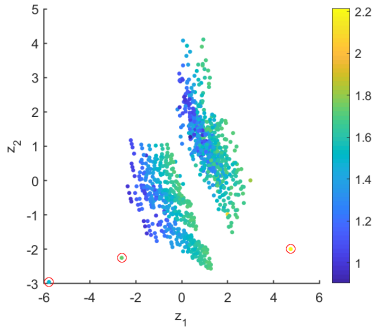
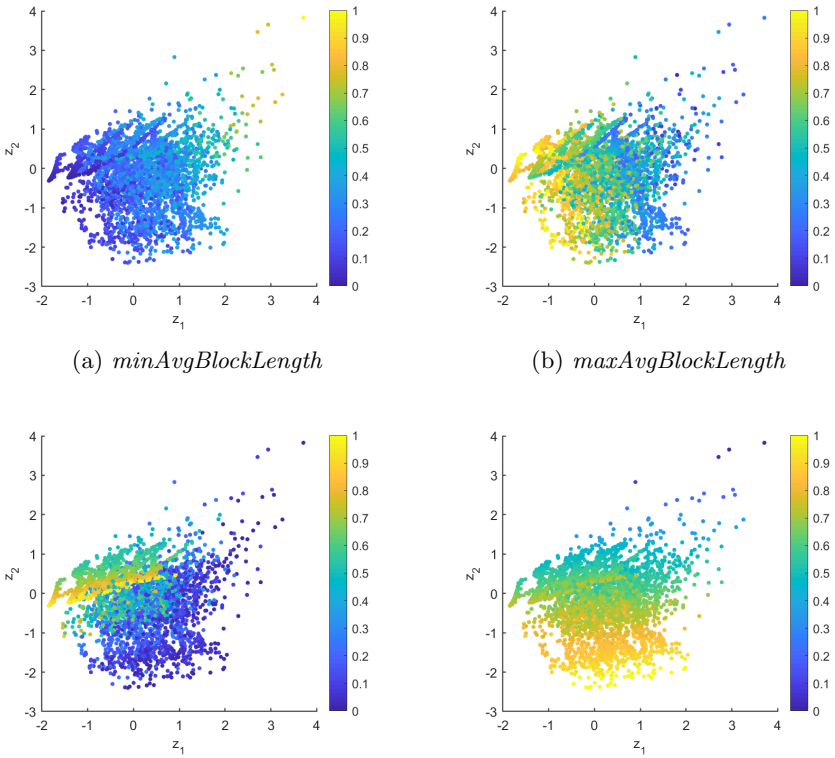


Fig. 3: Feature distribution for RWS, number of employees, original instances

Figure 3 shows the distribution of the instances (using their projection into the 2-dimensional plane based on the 5 most important features) and the distri-

[1] https://matilda.unimelb.edu.au/matilda/
[2] https://github.com/andremun/InstanceSpace

bution of the (normalized) number of employees (by the color scale). Note that the instances are separated into two clusters with a gap in-between, and with several real-life instances not covered by the artificial instances (red circles).



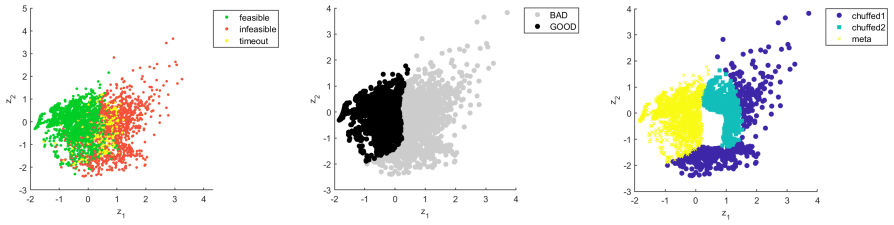
(a) *minAvgBlockLength* (b) *maxAvgBlockLength*

(c) *minDayFraction* (d) *maxDayFraction*

Fig. 4: Feature distribution for RWS, extended instances

Based on this analysis, additional instances where generated to cover the space more thoroughly. Figure 4 shows the new instance distribution in the instance space, and the distribution of the four main features that were identified by ISA as the most important ones. They represent variation between lengths of different working blocks on the z_1-axis, and variation of demand between different days on the z_2-axis.

Figure 5 shows several examples of analysis results. The investigation of feasibility reveals a clear boundary between feasible and infeasible instances in the space, with more difficult instances on this boundary. Footprints show the strong and weak areas of individual methods. A portfolio allows to select the best solution method for each instance based on the location in the instance space.



(a) Feasibility (b) Footprint metaheuristic (c) SVM Portfolio

Fig. 5: Examples of ISA results

6 Hyper-heuristics

Hyper-heuristics are a class of high-level problem solving techniques, which operate over a search space of other heuristic components, called *low-level heuristics*, instead of directly over the space of (potential) solutions [40]. Typically, this additional layer of indirection allows for the design of very general methods that perform well on and adapt to a wide range of problems, or even the automated design of new algorithms for specific problem domains.

A classification scheme by Burke et al. [6] distinguishes hyper-heuristics according to three dimensions: The main distinction is that between *selection* hyper-heuristics, where the heuristic space is defined as a discrete set of low-level heuristics, which are iteratively selected and applied by the hyper-heuristic, and *generation* hyper-heuristics, which automatically assemble low-level heuristics from smaller algorithmic primitives. The second dimension is the distinction between *constructive* hyper-heuristics, where the low-level heuristics assemble a (partial) solution from scratch, and *perturbative* hyper-heuristics, which start out from a complete (though not necessarily feasible) solution and modify it to find better solutions. Finally, hyper-heuristics can be classified according to the nature of the learning mechanisms they use, including *online* learning, *offline* learning, or no learning at all. A survey of hyper-heuristics of all types was performed in 2013 by Burke at al. [5], with an updated and revised version of the classification scheme published in 2019 [7].

A survey specifically focusing on selection hyper-heuristics was performed by Drake et al. [11]. Here, a major direction of research is into problem-independent hyper-heuristics, which are able to achieve a high performance on a wide variety and even previously unknown problem domains, where only the set of low-level heuristics is problem-specific. An important concept in this regard is that of a *domain barrier*, which limits the amount of (problem-specific) information passed from the problem domain to the hyper-heuristic to ensure its problem-independence. The most popular software framework following this approach is HyFlex [38], which was originally developed for the Cross-Domain Heuristic Search Challenge 2011 [4].

For generation hyper-heuristics, the focus typically falls on the automatic design of new algorithms for a specific problem. A popular mechanism for this approach is Genetic Programming, where either full algorithms or core algorithmic components (e.g. particular operators or evaluation functions) are automatically assembled from algorithmic primitives such as problem-specific instance attributes, arithmetic operators, and flow-control constructs (loops, if-else, ...). Examples of applications can be found in the tutorial by Tauritz and Woodward [52] or the corresponding chapter of the book on hyper-heuristics by Pillay and Qu [40].

Our own work on hyper-heuristics includes several new selection hyper-heuristics based on reinforcement learning [22, 29] or self-adaptive large neighborhood search [27], which were developed for the HyFlex framework. In addition, we investigated the application of hyper-heuristic methods on several real-world problem domains, such as artificial teeth scheduling [58], several variants of personnel scheduling problems [21], and test laboratory scheduling [30]. Here, we were able to show that problem-independent hyper-heuristics, operating on a suitable portfolio of (problem-specific) low-level heuristics, were able to produce results matching, and sometimes even surpassing those of the state-of-the-art problem-specific algorithms.

References

1. Alissa, M., Sim, K., Hart, E.: Automated algorithm selection: from feature-based to feature-free approaches. Journal of Heuristics **29**(1), 1–38 (2023)
2. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press (2009)
3. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Commun. ACM **54**(12), 92–103 (2011). https://doi.org/10.1145/2043174.2043195, http://doi.acm.org/10.1145/2043174.2043195
4. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., McCollum, B., Ochoa, G., Parkes, A.J., Petrovic, S.: The cross-domain heuristic search challenge – an international research competition. In: Coello, C.A.C. (ed.) Learning and Intelligent Optimization. pp. 631–634. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
5. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: A survey of the state of the art. Journal of the Operational Research Society **64**(12), 1695–1724 (2013)
6. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R.: A Classification of Hyper-heuristic Approaches, pp. 449–468. Springer US, Boston, MA (2010). https://doi.org/10.1007/978-1-4419-1665-5_15, https://doi.org/10.1007/978-1-4419-1665-5_15
7. Burke, E.K., Hyde, M.R., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R.: A Classification of Hyper-Heuristic Approaches: Revisited, pp. 453–477. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-319-91086-4_14, https://doi.org/10.1007/978-3-319-91086-4_14
8. Danzinger, P., Geibinger, T., Janneau, D., Mischek, F., Musliu, N., Poschalko, C.: A system for automated industrial test laboratory scheduling. ACM Trans. Intell. Syst. Technol. **14**(1), 3:1–3:27 (2023). https://doi.org/10.1145/3546871, https://doi.org/10.1145/3546871

9. Danzinger, P., Geibinger, T., Mischek, F., Musliu, N.: Solving the test laboratory scheduling problem with variable task grouping. In: Beck, J.C., Buffet, O., Hoffmann, J., Karpas, E., Sohrabi, S. (eds.) Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020. pp. 357–365. AAAI Press (2020), https://aaai.org/ojs/index.php/ICAPS/article/view/6681
10. De Coster, A., Musliu, N., Schaerf, A., Schoisswohl, J., Smith-Miles, K.: Algorithm selection and instance space analysis for curriculum-based course timetabling. Journal of Scheduling pp. 1–24 (2022)
11. Drake, J.H., Kheiri, A., Özcan, E., Burke, E.K.: Recent advances in selection hyper-heuristics. European Journal of Operational Research **285**(2), 405 – 428 (2020). https://doi.org/https://doi.org/10.1016/j.ejor.2019.07.073, http://www.sciencedirect.com/science/article/pii/S0377221719306526
12. Geibinger, T., Kletzander, L., Krainz, M., Mischek, F., Musliu, N., Winter, F.: Physician scheduling during a pandemic. In: Stuckey, P.J. (ed.) Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 18th International Conference, CPAIOR 2021, Vienna, Austria, July 5-8, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12735, pp. 456–465. Springer (2021). https://doi.org/10.1007/978-3-030-78230-6_29, https://doi.org/10.1007/978-3-030-78230-6_29
13. Geibinger, T., Kletzander, L., Musliu, N.: Instance space analysis for the generalized assignment problem. In: Metaheuristics International Conference. pp. 421–435. Springer (2022)
14. Geibinger, T., Mischek, F., Musliu, N.: Investigating constraint programming for real world industrial test laboratory scheduling. In: Proceedings of the Sixteenth International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR 2019) (2019)
15. Gendreau, M., Potvin, J.Y.E.: Handbook of metaheuristics. Springer (2010)
16. Jünger, M., Liebling, T.M., Naddef, D., Nemhauser, G.L., Pulleyblank, W.R., Reinelt, G., Rinaldi, G., Wolsey, L.A. (eds.): 50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art. Springer (2010). https://doi.org/10.1007/978-3-540-68279-0, http://dx.doi.org/10.1007/978-3-540-68279-0
17. Kang, Y., Hyndman, R.J., Smith-Miles, K.: Visualising forecasting algorithm performance using time series instance spaces. International Journal of Forecasting **33**(2), 345–358 (2017). https://doi.org/10.1016/j.ijforecast.2016.09.004
18. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983). https://doi.org/10.1126/science.220.4598.671, https://science.sciencemag.org/content/220/4598/671
19. Kletzander, L., Musliu, N.: Solving large real-life bus driver scheduling problems with complex break constraints. In: Beck, J.C., Buffet, O., Hoffmann, J., Karpas, E., Sohrabi, S. (eds.) Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020. pp. 421–430. AAAI Press (2020), https://aaai.org/ojs/index.php/ICAPS/article/view/6688
20. Kletzander, L., Musliu, N.: Solving the general employee scheduling problem. Comput. Oper. Res. **113** (2020). https://doi.org/10.1016/J.COR.2019.104794, https://doi.org/10.1016/j.cor.2019.104794
21. Kletzander, L., Musliu, N.: Hyper-heuristics for personnel scheduling domains. Proceedings of the International Conference on Automated Planning and

Scheduling **32**(1), 462–470 (Jun 2022). https://doi.org/10.1609/icaps.v32i1.19832, https://ojs.aaai.org/index.php/ICAPS/article/view/19832
22. Kletzander, L., Musliu, N.: Large-state reinforcement learning for hyper-heuristics. Proceedings of the AAAI Conference on Artificial Intelligence **37**(10), 12444–12452 (Jun 2023). https://doi.org/10.1609/aaai.v37i10.26466, https://ojs.aaai.org/index.php/AAAI/article/view/26466
23. Kletzander, L., Musliu, N., Gärtner, J., Krennwallner, T., Schafhauser, W.: Exact methods for extended rotating workforce scheduling problems. In: Proceedings of the International Conference on Automated Planning and Scheduling. vol. 29, pp. 519–527 (2019)
24. Kletzander, L., Musliu, N., Smith-Miles, K.: Instance space analysis for a personnel scheduling problem. Annals of Mathematics and Artificial Intelligence **89**, 617–637 (2021)
25. Lackner, M., Mrkvicka, C., Musliu, N., Walkiewicz, D., Winter, F.: Exact methods for the oven scheduling problem. Constraints An Int. J. **28**(2), 320–361 (2023). https://doi.org/10.1007/S10601-023-09347-2, https://doi.org/10.1007/s10601-023-09347-2
26. Minton, S., Johnston, M.D., Philips, A.B., Laird, P.: Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. Artificial Intelligence **58**, 161–205 (1992)
27. Mischek, F., Musliu, N.: A collection of hyper-heuristics for the hyflex framework. Tech. rep., TU Wien (2021)
28. Mischek, F., Musliu, N.: A local search framework for industrial test laboratory scheduling. Ann. Oper. Res. **302**(2), 533–562 (2021). https://doi.org/10.1007/S10479-021-04007-1, https://doi.org/10.1007/s10479-021-04007-1
29. Mischek, F., Musliu, N.: Reinforcement learning for cross-domain hyper-heuristics. In: Raedt, L.D. (ed.) Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022. pp. 4793–4799. ijcai.org (2022). https://doi.org/10.24963/ijcai.2022/664, https://doi.org/10.24963/ijcai.2022/664
30. Mischek, F., Musliu, N.: Leveraging problem-independent hyper-heuristics for real-world test laboratory scheduling. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 321–329 (2023)
31. Mischek, F., Musliu, N., Schaerf, A.: Local search approaches for the test laboratory scheduling problem with variable task grouping. J. Sched. **26**(5), 457–477 (2023). https://doi.org/10.1007/S10951-021-00699-2, https://doi.org/10.1007/s10951-021-00699-2
32. Moser, M., Musliu, N., Schaerf, A., Winter, F.: Exact and metaheuristic approaches for unrelated parallel machine scheduling. J. Sched. **25**(5), 507–534 (2022). https://doi.org/10.1007/S10951-021-00714-6, https://doi.org/10.1007/s10951-021-00714-6
33. Muñoz, M.A., Smith-Miles, K.A.: Performance analysis of continuous black-box optimization algorithms via footprints in instance space. Evolutionary Computation **25**(4), 529–554 (2017). https://doi.org/10.1162/evco_a_00194
34. Muñoz, M.A., Villanova, L., Baatar, D., Smith-Miles, K.: Instance spaces for machine learning classification. Machine Learning **107**(1), 109–147 (2018)
35. Musliu, N., Schutt, A., Stuckey, P.J.: Solver independent rotating workforce scheduling. In: van Hoeve, W.J. (ed.) Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 15th Inter-

national Conference, CPAIOR 2018, Delft, The Netherlands, June 26-29, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10848, pp. 429–445. Springer (2018). https://doi.org/10.1007/978-3-319-93031-2_31, https://doi.org/10.1007/978-3-319-93031-2_31
36. Musliu, N., Schwengerer, M.: Algorithm selection for the graph coloring problem. In: International conference on learning and intelligent optimization. pp. 389–403. Springer (2013)
37. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: Minizinc: Towards a standard CP modelling language. In: Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings. pp. 529–543 (2007). https://doi.org/10.1007/978-3-540-74970-7_38
38. Ochoa, G., Hyde, M., Curtois, T., Vazquez-Rodriguez, J.A., Walker, J., Gendreau, M., Kendall, G., McCollum, B., Parkes, A.J., Petrovic, S., Burke, E.K.: Hyflex: A benchmark framework for cross-domain heuristic search. In: Hao, J.K., Middendorf, M. (eds.) Evolutionary Computation in Combinatorial Optimization. pp. 136–147. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
39. Oliveira, C., Aleti, A., Grunske, L., Smith-Miles, K.: Mapping the effectiveness of automated test suite generation techniques. IEEE Transactions on Reliability **67**(3), 771–785 (2018)
40. Pillay, N., Qu, R.: Hyper-heuristics: theory and applications. Springer (2018)
41. Rice, J.R.: The algorithm selection problem. In: Advances in computers, vol. 15, pp. 65–118. Elsevier (1976)
42. Rossi, F., Van Beek, P., Walsh, T.: Handbook of constraint programming. Elsevier (2006)
43. Schwengerer, M.: Algorithm selection for the graph coloring problem. Ph.D. thesis, TU Wien (2012)
44. Smith-Miles, K., Baatar, D., Wreford, B., Lewis, R.: Towards objective measures of algorithm performance across instance space. Computers & Operations Research **45**, 12–24 (2014)
45. Smith-Miles, K., Bowly, S.: Generating new test instances by evolving in instance space. Computers & Operations Research **63**, 102–113 (2015). https://doi.org/10.1016/j.cor.2015.04.022
46. Smith-Miles, K., Lopes, L.: Measuring instance difficulty for combinatorial optimization problems. Computers & Operations Research **39**(5), 875–889 (2012)
47. Smith-Miles, K., Muñoz, M.A.: Instance space analysis for algorithm testing: Methodology and software tools. ACM Comput. Surv. **55**(12) (3 2023). https://doi.org/10.1145/3572895, https://doi.org/10.1145/3572895
48. Smith-Miles, K.A.: Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Computing Surveys (CSUR) **41**(1), 1–25 (2009)
49. Strassl, S., Musliu, N.: Instance space analysis and algorithm selection for the job shop scheduling problem. Computers & Operations Research **141**, 105661 (2022)
50. Stuckey, P.J. (ed.): Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 18th International Conference, CPAIOR 2021, Vienna, Austria, July 5-8, 2021, Proceedings, Lecture Notes in Computer Science, vol. 12735. Springer (2021). https://doi.org/10.1007/978-3-030-78230-6, https://doi.org/10.1007/978-3-030-78230-6
51. Szeredi, R., Schutt, A.: Modelling and solving multi-mode resource-constrained project scheduling. In: Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings. pp. 483–492 (2016). https://doi.org/10.1007/978-3-319-44953-1_31

52. Tauritz, D.R., Woodward, J.: Generative hyper-heuristics. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. p. 1111–1140. GECCO '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3520304.3533646, https://doi.org/10.1145/3520304.3533646
53. Van Hentenryck, P., Milano, M.: Hybrid optimization: the ten years of CPAIOR, vol. 45. Springer Science & Business Media (2010)
54. Vass, J., Lackner, M., Mrkvicka, C., Musliu, N., Winter, F.: Exact and meta-heuristic approaches for the production leveling problem. J. Sched. **25**(3), 339–370 (2022). https://doi.org/10.1007/S10951-022-00721-1, https://doi.org/10.1007/s10951-022-00721-1
55. Winter, F., Meiswinkel, S., Musliu, N., Walkiewicz, D.: Modeling and solving parallel machine scheduling with contamination constraints in the agricultural industry. In: Solnon, C. (ed.) 28th International Conference on Principles and Practice of Constraint Programming, CP 2022, July 31 to August 8, 2022, Haifa, Israel. LIPIcs, vol. 235, pp. 41:1–41:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPICS.CP.2022.41, https://doi.org/10.4230/LIPIcs.CP.2022.41
56. Winter, F., Musliu, N.: Constraint-based scheduling for paint shops in the automotive supply industry. ACM Trans. Intell. Syst. Technol. **12**(2), 17:1–17:25 (2021). https://doi.org/10.1145/3430710, https://doi.org/10.1145/3430710
57. Winter, F., Musliu, N.: A large neighborhood search approach for the paint shop scheduling problem. J. Sched. **25**(4), 453–475 (2022). https://doi.org/10.1007/S10951-021-00713-7, https://doi.org/10.1007/s10951-021-00713-7
58. Winter, F., Musliu, N.: An investigation of hyper-heuristic approaches for teeth scheduling. In: Di Gaspero, L., Festa, P., Nakib, A., Pavone, M. (eds.) Metaheuristics. pp. 274–289. Springer International Publishing, Cham (2023)
59. Winter, F., Musliu, N., Demirovic, E., Mrkvicka, C.: Solution approaches for an automotive paint shop scheduling problem. In: Benton, J., Lipovetzky, N., Onaindia, E., Smith, D.E., Srivastava, S. (eds.) Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS 2019, Berkeley, CA, USA, July 11-15, 2019. pp. 573–581. AAAI Press (2019), https://ojs.aaai.org/index.php/ICAPS/article/view/3524
60. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE transactions on evolutionary computation **1**(1), 67–82 (1997)
61. Wolpert, D.H., Macready, W.G.: Coevolutionary free lunches. IEEE Transactions on evolutionary computation **9**(6), 721–735 (2005)
62. Young, K.D., Feydy, T., Schutt, A.: Constraint programming applied to the multi-skill project scheduling problem. In: Principles and Practice of Constraint Programming - 23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings. pp. 308–317 (2017). https://doi.org/10.1007/978-3-319-66158-2_20

Logic for declarative problem-solving and its applications*

Gerhard Friedrich[0000−0002−1992−4049] and Martin Gebser[0000−0002−8010−4752]

University of Klagenfurt, Universitätsstraße 65-67, 9020 Klagenfurt, Austria
{gerhard.friedrich,martin.gebser}@aau.at

Abstract. This note gives a brief overview of topic areas addressed in our lecture on 6 July at the AI Summer School 2023, hosted by the Center for Artificial Intelligence and Machine Learning (CAIML) at TU Wien. We start with a recap of propositional satisfiability as a groundbreaking and prominent representative for the field of computational logic. The central focus of our lecture and this overview is dedicated to answer set programming, a declarative problem-solving paradigm geared for knowledge representation and reasoning. We provide an account of the syntax and semantics of logic programs, which constitute the specification language of answer set programming, and illustrate their application on an optimization problem in the context of software package configuration. While the scope and level of detail of this short note are limited, we hope to equip the reader with literature references pointing out relevant publications for in-depth reading.

1 Introduction

With the inception of complexity theory [16], *propositional satisfiability* (SAT) [9] emerged as a universal representation formalism for NP-complete search problems [30]. Thanks to the steady increase of computing power and hardware availability, the interest in SAT has shifted from a primarily theoretical perspective to practical implementations and applications. For example, SAT approaches proved to be effective for solving complex planning [48], scheduling [17], and configuration [29] problems, where modern SAT solving techniques range from *conflict-driven clause learning* (CDCL) [8, 23, 60, 62] over *local search* methods [49, 66, 69] to algorithm *portfolios* [41, 55, 58, 68].

While SAT solvers are thoroughly engineered and empirically efficient, the limited *knowledge representation* capacities [42, 61, 65] of classical logic often call for procedural front-ends and expert programmers to bootstrap solver inputs [11]. For a high-level approach to knowledge representation and reasoning, the *answer set programming* (ASP) paradigm [53, 59, 63] builds on the *stable model semantics* [26, 27, 40, 64] for *logic programming* [5, 52, 56] instead of the classical semantics of logical formulas. In a nutshell, ASP is an approach to *declarative problem-solving* [57], integrating concise and readable first-order input specifications, written by users, with powerful search methods at the propositional level, inspired by solving techniques pioneered for SAT.

In this note, we briefly present the syntax and semantics of *logic programs*, which specify the solutions to (search) problems in ASP, and illustrate their practical use on

* This work was partially funded by KWF project 28472, cms electronics GmbH, FunderMax GmbH, Hirsch Armbänder GmbH, incubed IT GmbH, Infineon Technologies Austria AG, Isovolta AG, Kostwein Holding GmbH, and Privatstiftung Kärntner Sparkasse.

an *optimization problem* in the application area of software package configuration. See, e.g., [7, 33, 39, 54] for much more elaborate introductions to ASP, [10, 24, 25, 38, 50] for best practice modeling methodology and its application areas, as well as [12, 37, 45, 47] for computational approaches to *grounding* and *solving* implemented by ASP systems.

2 Syntax and semantics of logic programs

The approach of ASP is to describe the solutions to (search) problems in terms of logic programs, which are sets of rules specifying the properties of solutions but no procedure how to compute them. In fact, the solution computation is automated by ASP systems like CLINGO [34], DLV [12], and WASP [22] that provide domain-independent solving techniques off-the-shelf. In the following, we introduce the subset of these systems' input languages [13, 14, 31] sufficient to model our example problem in Section 3.

We consider (nested) *rules* written in the form

$$a_1, \ldots, a_k, \text{not } a_{k+1}, \ldots, \text{not } a_l \text{ :- } a_{l+1}, \ldots, a_m, \text{not } a_{m+1}, \ldots, \text{not } a_n. \quad (1)$$

where $0 \leq k \leq l \leq m \leq n$, the connective not denotes *default negation*, and :– resembles a *converse implication* arrow. (If $l = n$, we skip :– and write the rule as $a_1, \ldots, a_k, \text{not } a_{k+1}, \ldots, \text{not } a_l$.) Each a_i, for $1 \leq i \leq n$, is a (first-order) *atom* of the form $p(t_1, \ldots, t_h)$ in which p is a *predicate* and every t_j, for $1 \leq j \leq h$, is a *term*, i.e., a *constant* or a *variable*. Predicates as well as constants are written as alphanumeric strings starting with a lowercase letter, constants can also be integers, and alphanumeric strings starting with an uppercase letter denote variables.

The intuitive meaning of a rule (1) is that some of the atoms $a_1, \ldots, a_k$ must be true if all atoms $a_{l+1}, \ldots, a_m$ are (provably) true and if it is consistent to assume that a_{k+1}, $\ldots$, a_l are true and $a_{m+1}, \ldots, a_n$ are false. In other words, the rule *body* $a_{l+1}, \ldots, a_m$, not $a_{m+1}, \ldots, \text{not } a_n$ is a conjunction that holds when the atoms occurring positively are provable and the negated atoms are false. The rule *head* $a_1, \ldots, a_k, \text{not } a_{k+1}, \ldots,$ not a_l is a disjunction, where either some of the negated atoms needs to be false or an atom occurring positively must be true if the rule body holds.

Two prominent special cases of a rule (1) are *facts* a_1., i.e., $k = l = m = n = 1$, and *integrity constraints* :– $a_{l+1}, \ldots, a_m, \text{not } a_{m+1}, \ldots, \text{not } a_n$., where $k = l = 0$. While facts provide definite knowledge in the form of head atoms that must unconditionally be true, the empty head of an integrity constraint cannot be satisfied so that its body must not hold. Another frequent special case is a *choice rule*

$$a, \text{not } a \text{ :- } a_3, \ldots, a_m, \text{not } a_{m+1}, \ldots, \text{not } a_n.$$

and we will write it as

$$\{a\} \text{ :- } a_3, \ldots, a_m, \text{not } a_{m+1}, \ldots, \text{not } a_n.$$

in the sequel, where $\{a\}$ indicates that the head atom a can be chosen to be true or false.

In order to represent optimization objectives, *weak constraints* of the form

$$\text{:\sim } a_{l+1}, \ldots, a_m, \text{not } a_{m+1}, \ldots, \text{not } a_n. \; [w@v, t_1, \ldots, t_h] \quad (2)$$

have an empty head, similar to integrity constraints, yet incur a *weight* w at *level* v as penalty if the body $a_{l+1}, \ldots, a_m, \text{not } a_{m+1}, \ldots, \text{not } a_n$ holds, where $t_1, \ldots, t_h$ are terms disambiguating recurrences of the penalty $w@v$.

A *logic program* P is a set of rules and weak constraints. The *ground instantiation* $ground(P)$ of P is the set of all rules and weak constraints that can be obtained by substituting variables in P with constants. That is, variables in rules and weak constraints of P are universally quantified, and $ground(P)$ includes respective (variable-free) instances taking all constants as possible variable values.

We are now ready to characterize the semantics of a logic program P in terms of *interpretations* X, i.e., sets of ground (variable-free) atoms that are true. To this end, the *reduct* P^X of P relative to some interpretation X is defined as:

$$P^X = \left\{ a_1, \ldots, a_k \coloneq a_{l+1}, \ldots, a_m. \;\middle|\; \begin{array}{l} \text{there is a rule (1) in } ground(P) \text{ with} \\ \{a_{k+1}, \ldots, a_l\} \subseteq X \text{ and} \\ \{a_{m+1}, \ldots, a_n\} \cap X = \emptyset \end{array} \right\}$$

An interpretation X' is a *model* of P^X if we have that $\{a_1, \ldots, a_k\} \cap X' \neq \emptyset$ or $\{a_{l+1}, \ldots, a_m\} \nsubseteq X'$ holds for every rule in P^X. That is, a model X' of P^X satisfies the head of a rule in P^X if its body holds. The interpretation X is an *answer set*, also called *stable model*, of P if X is a model of P^X such that there is no model $X' \subset X$ of P^X. Note that the minimality of an answer set X among models of its reduct P^X reflects provability, i.e., the true atoms in X must be derivable by means of the rules in P^X.

It remains to identify the optimal answer sets of a logic program P in the presence of weak constraints. Hence, by $levels(P)$, we denote the set of all integers v such that some weak constraint (2) with the penalty $w@v$ occurs in $ground(P)$. For any interpretation X and $v \in levels(P)$, let $weak(P, v, X)$ be the set of all term tuples $(w, t_1, \ldots, t_h)$ such that there is a weak constraints (2) in $ground(P)$ for which w in the penalty $w@v$ is an integer, $\{a_{l+1}, \ldots, a_m\} \subseteq X$, and $\{a_{m+1}, \ldots, a_n\} \cap X = \emptyset$. Summing up the weights w over tuples in $weak(P, v, X)$ yields a score for X at level v:

$$score(P, v, X) = \textstyle\sum_{(w, t_1, \ldots, t_h) \in weak(P, v, X)} w$$

Then, an answer set X of P is *optimal* if there is no answer set X' of P such that, for some $v \in levels(P)$, $score(P, v, X') < score(P, v, X)$ and $score(P, v', X') = score(P, v', X)$ for any $v' \in levels(P)$ with $v < v'$. That is, scores are minimized in decreasing order of levels, and the scores obtained for optimal answer sets are minimal.

For illustration, consider a (ground) logic program P consisting of the following rules and weak constraints:

$$\{bottleChosen(white)\}. \quad (3)$$
$$\{bottleChosen(rose)\}. \quad (4)$$
$$\{bottleChosen(red)\}. \quad (5)$$
$$hasBottle(axel) \text{ :– } bottleChosen(white). \quad (6)$$
$$hasBottle(axel) \text{ :– } bottleChosen(rose). \quad (7)$$
$$hasBottle(roman) \text{ :– } bottleChosen(rose). \quad (8)$$
$$hasBottle(roman) \text{ :– } bottleChosen(red). \quad (9)$$
$$\text{:– not } hasBottle(axel). \quad (10)$$
$$\text{:– not } hasBottle(roman). \quad (11)$$
$$\text{:~ } bottleChosen(white). [5@1, white] \quad (12)$$
$$\text{:~ } bottleChosen(rose). [8@1, rose] \quad (13)$$
$$\text{:~ } bottleChosen(red). [5@1, red] \quad (14)$$

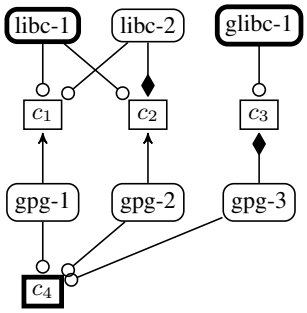
The choice rules (3)–(5) state that bottles with $white$, $rose$, and red wine can each be chosen (unconditionally). If the bottle with $white$ or $rose$ wine is chosen, the person $axel$ has an appropriate bottle, and likewise with the $rose$ or red wine bottle for $roman$, as expressed by the rules (6)–(9). The integrity constraints (10)–(11) require that some appropriate bottle must be chosen for each of the persons $axel$ and $roman$. In view of these rules, the logic program P has five answer sets, each of which extends the set $X = \{hasBottle(axel), hasBottle(roman)\}$ of atoms:

$$X_1 = X \cup \{bottleChosen(rose)\}$$
$$X_2 = X \cup \{bottleChosen(white), bottleChosen(rose)\}$$
$$X_3 = X \cup \{bottleChosen(white), bottleChosen(red)\}$$
$$X_4 = X \cup \{bottleChosen(rose), bottleChosen(red)\}$$
$$X_5 = X \cup \{bottleChosen(white), bottleChosen(rose), bottleChosen(red)\}$$

Given the weak constraints (12)–(14), since all answer sets X_i except for X_3 contain the atom $bottleChosen(rose)$, the corresponding sets $weak(P, 1, X_i)$ include the term tuple $(8, rose)$. Similarly, the term tuple $(5, white)$ belongs to $weak(P, 1, X_2)$, $weak(P, 1, X_3)$, and $weak(P, 1, X_5)$, while $weak(P, 1, X_3)$ to $weak(P, 1, X_5)$ include $(5, red)$. As a consequence, we obtain $score(P, 1, X_1) = 8$, $score(P, 1, X_2) = 13$, $score(P, 1, X_3) = 10$, $score(P, 1, X_4) = 13$, and $score(P, 1, X_5) = 18$ as scores (at level 1) for the five answer sets. Since the score 8 for X_1 is minimal and the other scores are strictly greater, X_1 with $bottleChosen(rose)$ is the only optimal answer set of P.

3 Optimal Linux package configuration with ASP

The Linux package configuration problem has been posed as a real-world challenge in the context of the MANCOOSI project [20], as part of the research on free and open source software management. In fact, when a package manager like APT, which is included in Debian and Ubuntu Linux distributions, is run for updating an installation,



(a) Graph representation of the instance

$package(libc, 1).\ package(libc, 2).$
$package(glibc, 1).\ package(gpg, 1).$
$package(gpg, 2).\ package(gpg, 3).$
$satisfies(libc, 1, c_1).\ satisfies(libc, 2, c_1).$
$satisfies(libc, 1, c_2).\ satisfies(glibc, 1, c_3).$
$satisfies(gpg, 1, c_4).\ satisfies(gpg, 2, c_4).$
$satisfies(gpg, 3, c_4).\ request(c_4).$
$depends(gpg, 1, c_1).\ conflicts(libc, 2, c_2).$
$depends(gpg, 2, c_2).\ conflicts(gpg, 3, c_3).$
$installed(libc, 1).\ installed(glibc, 1).$
$utility(delete, 1).\ utility(change, 2).$

(b) Fact representation of the instance and objectives

Fig. 1: Problem instance where a package satisfying the condition c_4 has to be installed

it evaluates package meta-data to fulfil dependencies and avoid conflicts among the installed packages. As the underlying installability problem is NP-complete, greedy approaches as the one implemented in APT cannot guarantee to find a solution, even if an installation exists, nor reduce required package additions and removals to a minimum.

For promoting the design of more powerful and reliable methods, the MANCOOSI *international solver competition* (MISC) [1] has been initiated to evaluate and showcase elaborate optimization approaches in a common setting. To this end, several optimization objectives and collections thereof are predefined and assessed in separate tracks: *paranoid* for a server machine, *trendy* for a desktop machine, and *user* for a customized collection of objectives. In the following, we present a simplified version of the ASPCUD solving approach [35] for the *paranoid* track. Notably, ASPCUD utilizes a *uniform problem encoding* for all tracks, i.e., a single (first-order) logic program includes rules and weak constraints specifying all of the predefined optimization objectives, while the objectives to apply along with their levels are configured by facts in a problem instance.[1]

An instance of the Linux package configuration problem is visualized in Figure 1(a) and formally specified by the facts in Figure 1(b). The six versioned packages *libc*-1, *libc*-2, *glibc*-1, *gpg*-1, *gpg*-2, and *gpg*-3 are available for installation, of which *libc*-1 and *glibc*-1 constitute the current installation. An installed package satisfies associated conditions, i.e., c_1 and c_2 are satisfied by *libc*-1 as well as c_3 by *glibc*-1. Moreover, the installation of a package can depend on conditions, such as *gpg*-1 and *gpg*-2 requiring the satisfaction of c_1 or c_2, respectively, while the conflicts of *libc*-2 with c_2 and of *gpg*-3 with c_3 prohibit the conditions to be satisfied when these packages are installed. The task for the instance in Figure 1 consists of updating the current installation such that the condition c_4 gets satisfied, for which some of the packages *gpg*-1, *gpg*-2, and *gpg*-3 needs to be installed. When comparing a follow-up to the current installation, the *paranoid* optimization objectives are to minimize package changes in the first place, meaning that the installed versions of a package should remain as is whenever possible,

[1] For the ASPCUD implementation, see: https://potassco.org/aspcud/

$$\{install(P,V)\} \text{ :- } package(P,V). \quad (15)$$
$$install(P) \text{ :- } install(P,V). \quad (16)$$
$$satisfy(C) \text{ :- } install(P,V), satisfies(P,V,C). \quad (17)$$
$$exclude(C) \text{ :- } install(P,V), conflicts(P,V,C). \quad (18)$$
$$include(C) \text{ :- } install(P,V), depends(P,V,C). \quad (19)$$
$$\text{:- } exclude(C), satisfy(C). \quad (20)$$
$$\text{:- } include(C), \text{not } satisfy(C). \quad (21)$$
$$\text{:- } request(C), \text{not } satisfy(C). \quad (22)$$
$$violate(delete,L,P) \text{ :- } utility(delete,L), installed(P,V), \text{not } install(P). \quad (23)$$
$$violate(change,L,P) \text{ :- } utility(change,L), installed(P,V), \text{not } install(P,V). \quad (24)$$
$$violate(change,L,P) \text{ :- } utility(change,L), install(P,V), \text{not } installed(P,V). \quad (25)$$
$$\text{:~ } violate(U,L,P). \; [1@L,U,P] \quad (26)$$

Fig. 2: Linux package configuration encoding for *paranoid* optimization objectives

and to further omit package deletions as secondary objective, reflecting the intent to preserve at least some version of an installed package in case a change is unavoidable.

The first-order logic program, also called *encoding*, in Figure 2 specifies optimal follow-up installations under *paranoid* optimization objectives for arbitrary instances of the Linux package configuration problem. Its outline follows the common *generate-and-test* modeling pattern [24, 38, 50, 53], where choice rules provide solution candidates, further rules and integrity constraints verify the properties of solutions, and weak constraints determine the solutions' scores. In more detail, the choice rule (15) states that every versioned package may belong to a follow-up installation, and the rule (16) indicates packages of which some version will be installed. For the packages of a follow-up installation, the rules (17)–(19) derive conditions that are satisfied, must not be satisfied due to conflicts, or need to be satisfied to fulfil dependencies, respectively. The integrity constraints (20)–(21) then check the absence of conflicts as well as the satisfaction of required dependencies, and the integrity constraint (22) makes sure that installation requests are fulfilled. Taken together, the rules (15)–(22) thus establish that answer sets match problem solutions, i.e., they represent follow-up installations satisfying all package dependencies and installation requests while avoiding conflicts.

It remains to associate answer sets with scores in order to optimize objectives configured by facts in a problem instance. To this end, the rule (23) indicates packages of which some version is currently installed, while none of them is included in the follow-up installation (provided that package deletions are to be minimized at some instance-specific level). Similarly, the rules (24)–(25) reflect package changes, occurring when a currently installed version is to be dropped or some new version will be installed. As the weak constraint (26) incurs the penalty $1@L$ for any atom of the form $violate(U,L,P)$, expressing that an optimization objective U is violated for the package P at level L, the $install(P,V)$ atoms in an optimal answer set yield a best choice follow-up installation.

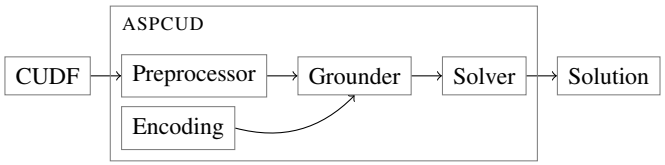


Fig. 3: Complete workflow of the ASPCUD solver for Linux package configuration

Among fifteen feasible follow-up installations that fulfil the request c_4 for the problem instance in Figure 1, three are optimal. They include the versioned packages $libc$-1 and $glibc$-1 from the current installation in order to avoid package changes for $libc$ and $glibc$. In addition, gpg-1, gpg-2, or both get installed to satisfy the condition c_4, which gives the three optimal follow-up installation possibilities. The resulting package change for gpg cannot be omitted, yet refraining from an installation of gpg-3, which conflicts with the condition c_3, avoids a package change (and deletion) for $glibc$.

After focussing on the core optimization by means of ASP, let us briefly outline the complete workflow of the ASPCUD solver, which is displayed in Figure 3. For interoperability, ASPCUD reads in package meta-data in the *common upgradability description format* (CUDF) [1], and its *preprocessor* takes care of producing a fact representation as illustrated in Figure 1(b). Beyond pure textual conversion from CUDF to ASP facts, the preprocessor analyzes package dependencies and optimization objectives to skip facts for *irrelevant packages*, i.e., packages that are not currently installed and cannot help to fulfil dependencies and installation requests. Moreover, the preprocessor applies a greedy algorithm to partition the packages whose installation may be relevant into *conflict cliques*, where at most one package per clique can possibly be installed. Both of these deterministic preprocessing steps could in principle also be encoded in ASP by adding respective rules to Figure 2, but their procedural implementation in the preprocessor is computationally way more efficient. While the encoding in Figure 2 suffices for the *paranoid* track, the full encoding utilized by ASPCUD specifies all MISC optimization objectives, consisting of five applicable measures that can each be combined with six selection criteria on packages.

As common for declarative problem-solving with ASP, the uniform problem encoding and the instance obtained from the preprocessor first undergo a *grounding* step [12, 45, 47], responsible for producing a concise yet equivalent representation of the (theoretical) ground instantiation of the logic program comprising the instance and encoding. This step substantially benefits from the omission of facts for irrelevant packages, as large Linux distributions with 50.000 packages or more can often be condensed to 10–20% fractions of packages whose installation matters, thus sparing the generation of rule instances for the majority of available packages. The subsequent *solving* step [22, 32, 37] performs the search for optimal answer sets, where ASP systems support both *model-* and *core-guided optimization* strategies [2], which are also successfully applied to optimization versions of SAT. Considering that dependency graphs like in Figure 1(a) can be large but tend to be sparse for the Linux package configuration problem, core-guided optimization that identifies local structures inducing the scores for answer sets is particularly advantageous, and thus ASPCUD runs the ASP system CLINGO with a core-

guided optimization strategy. The conflict cliques obtained from the preprocessor are additionally incorporated to break down scores, given that the encoding can attribute the score for an entire clique to a single or no package to be installed, while other (impossible) cases are ruled out. By means of its carefully devised preprocessing and encoding techniques, ASPCUD succeeded to win all tracks of the last MISC competition in 2012 by a noticeable margin, owed to the knowledge representation capacities of a uniform problem encoding ready to deal with various optimization objectives in inputs.

4 Conclusion

The ASP paradigm for declarative problem-solving integrates high-level knowledge representation by uniform problem encodings in a first-order input language with powerful search and optimization techniques pioneered in the closely tied area of SAT solving. A grounding step automates the instantiation of a uniform problem encoding relative to facts describing a specific instance, where the underlying stable model semantics is particularly beneficial for computational efficiency. This approach enables concise and readable first-order input specifications, which promote the rapid prototyping of new applications by domain rather than programming experts.

Active research and development directions include, e.g., flexible interfaces for the customization of heuristics and interoperation of ASP system components [15, 21, 22, 34, 36]. Further conceptual and practical extensions address the integration of constraint and integer programming methods [18, 44, 46, 51], reasoning techniques for computational tasks and preference rankings of elevated complexity [3, 4, 28, 43], and lazy-grounding approaches for demand-driven instantiation during search [6, 18, 19, 67].

References

1. Abate, P., Di Cosmo, R., Treinen, R., Zacchiroli, S.: Dependency solving: A separate concern in component evolution management. Journal of Systems and Software **85**(10), 2228–2240 (2012)
2. Alviano, M., Dodaro, C., Marques-Silva, J., Ricca, F.: Optimum stable model search: Algorithms and implementation. Journal of Logic and Computation **30**(4), 863–897 (2020)
3. Alviano, M., Romero, J., Schaub, T.: On the integration of CP-nets in ASPRIN. In: Kraus, S. (ed.) Proceedings of the Twenty-eighth International Joint Conference on Artificial Intelligence (IJCAI'19). pp. 1495–1501. ijcai.org (2019)
4. Amendola, G., Cuteri, B., Ricca, F., Truszczyński, M.: Solving problems in the polynomial hierarchy with ASP(Q). In: Gottlob, G., Inclezan, D., Maratea, M. (eds.) Proceedings of the Sixteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'22). Lecture Notes in Artificial Intelligence, vol. 13416, pp. 373–386. Springer-Verlag (2022)
5. Apt, K., Bol, R.: Logic programming and negation: A survey. Journal of Logic Programming **19-20**, 9–71 (1994)
6. Arias, J., Carro, M., Gupta, G.: Towards dynamic consistency checking in goal-directed predicate answer set programming. In: Cheney, J., Perri, S. (eds.) Proceedings of the Twenty-fourth International Symposium on Practical Aspects of Declarative Languages (PADL'22). Lecture Notes in Computer Science, vol. 13165, pp. 117–134. Springer-Verlag (2022)

7. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
8. Biere, A., Fröhlich, A.: Evaluating CDCL variable scoring schemes. In: Heule, M., Weaver, S. (eds.) Proceedings of the Eighteenth International Conference on Theory and Applications of Satisfiability Testing (SAT'15). Lecture Notes in Computer Science, vol. 9340, pp. 405–422. Springer-Verlag (2015)
9. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability - Second Edition. Frontiers in Artificial Intelligence and Applications, vol. 336. IOS Press (2021)
10. Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. Communications of the ACM **54**(12), 92–103 (2011)
11. Cadoli, M., Schaerf, A.: Compiling problem specifications into SAT. Artificial Intelligence **162**(1-2), 89–120 (2005)
12. Calimeri, F., Dodaro, C., Fuscà, D., Perri, S., Zangari, J.: Efficiently coupling the I-DLV grounder with ASP solvers. Theory and Practice of Logic Programming **20**(2), 205–224 (2020)
13. Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Maratea, M., Ricca, F., Schaub, T.: ASP-Core-2 input language format. Theory and Practice of Logic Programming **20**(2), 294–309 (2020)
14. Calimeri, F., Fuscà, D., Perri, S., Zangari, J.: I-DLV: The new intelligent grounder of DLV. Intelligenza Artificiale **11**(1), 5–20 (2017)
15. Comploi-Taupe, R., Friedrich, G., Schekotihin, K., Weinzierl, A.: Domain-specific heuristics in answer set programming: A declarative non-monotonic approach. Journal of Artificial Intelligence Research **76**, 59–114 (2023)
16. Cook, S.: The complexity of theorem-proving procedures. In: Harrison, M., Banerji, R., Ullman, J. (eds.) Proceedings of the Third Annual ACM Symposium on Theory of Computing (STOC'71). pp. 151–158. ACM Press (1971)
17. Crawford, J., Baker, A.: Experimental results on the application of satisfiability algorithms to scheduling problems. In: Hayes-Roth, B., Korf, R. (eds.) Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI'94). pp. 1092–1097. AAAI Press (1994)
18. Cuteri, B., Dodaro, C., Ricca, F., Schüller, P.: Overcoming the grounding bottleneck due to constraints in ASP solving: Constraints become propagators. In: Bessiere, C. (ed.) Proceedings of the Twenty-ninth International Joint Conference on Artificial Intelligence (IJCAI'20). pp. 1688–1694. ijcai.org (2020)
19. De Cat, B., Denecker, M., Bruynooghe, M., Stuckey, P.: Lazy model expansion: Interleaving grounding with search. Journal of Artificial Intelligence Research **52**, 235–286 (2015)
20. Di Cosmo, R., Treinen, R., Zacchiroli, S.: Formal aspects of free and open source software components: A short survey. In: Giachino, E., Hähnle, R., de Boer, F., Bonsangue, M. (eds.) Proceedings of the Eleventh International Symposium on Formal Methods for Components and Objects (FMCO'12). Lecture Notes in Computer Science, vol. 7866, pp. 216–239. Springer-Verlag (2012)
21. Dodaro, C., Gasteiger, P., Leone, N., Musitsch, B., Ricca, F., Schekotihin, K.: Combining answer set programming and domain heuristics for solving hard industrial problems. Theory and Practice of Logic Programming **16**(5-6), 653–669 (2016)
22. Dodaro, C., Ricca, F.: The external interface for extending WASP. Theory and Practice of Logic Programming **20**(2), 225–248 (2020)
23. Eén, N., Sörensson, N.: An extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds.) Proceedings of the Sixth International Conference on Theory and Applications of Satisfiability Testing (SAT'03). Lecture Notes in Computer Science, vol. 2919, pp. 502–518. Springer-Verlag (2004)

24. Eiter, T., Ianni, G., Krennwallner, T.: Answer Set Programming: A Primer. In: Tessaris, S., Franconi, E., Eiter, T., Gutierrez, C., Handschuh, S., Rousset, M., Schmidt, R. (eds.) Proceedings of the Fifth International Reasoning Web Summer School (RW'09). Lecture Notes in Computer Science, vol. 5689, pp. 40–110. Springer-Verlag (2009)
25. Erdem, E., Gelfond, M., Leone, N.: Applications of ASP. AI Magazine **37**(3), 53–68 (2016)
26. Faber, W., Pfeifer, G., Leone, N.: Semantics and complexity of recursive aggregates in answer set programming. Artificial Intelligence **175**(1), 278–298 (2011)
27. Ferraris, P.: Logic programs with propositional connectives and aggregates. ACM Transactions on Computational Logic **12**(4), 25:1–25:40 (2011)
28. Fichte, J., Hecher, M., Morak, M., Thier, P., Woltran, S.: Solving projected model counting by utilizing treewidth and its limits. Artificial Intelligence **314**, Article ID 103810 (2023)
29. Fréchette, A., Newman, N., Leyton-Brown, K.: Solving the station repacking problem. In: Schuurmans, D., Wellman, M. (eds.) Proceedings of the Thirtieth National Conference on Artificial Intelligence (AAAI'16). pp. 702–709. AAAI Press (2016)
30. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. Freeman and Co. (1979)
31. Gebser, M., Kaminski, R., Kaufmann, B., Lindauer, M., Ostrowski, M., Romero, J., Schaub, T., Thiele, S., Wanko, P.: Potassco User Guide, version 2.2.0. https://potassco.org (2019)
32. Gebser, M., Kaminski, R., Kaufmann, B., Romero, J., Schaub, T.: Progress in clasp series 3. In: Calimeri, F., Ianni, G., Truszczyński, M. (eds.) Proceedings of the Thirteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'15). Lecture Notes in Artificial Intelligence, vol. 9345, pp. 368–383. Springer-Verlag (2015)
33. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Answer Set Solving in Practice. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers (2012)
34. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Multi-shot ASP solving with clingo. Theory and Practice of Logic Programming **19**(1), 27–82 (2019)
35. Gebser, M., Kaminski, R., Schaub, T.: aspcud: A Linux package configuration tool based on answer set programming. In: Drescher, C., Lynce, I., Treinen, R. (eds.) Proceedings of the Second International Workshop on Logics for Component Configuration (LoCoCo'11). Electronic Proceedings in Theoretical Computer Science (EPTCS), vol. 65, pp. 12–25. Open Publishing Association (2011)
36. Gebser, M., Kaufmann, B., Otero, R., Romero, J., Schaub, T., Wanko, P.: Domain-specific heuristics in answer set programming. In: desJardins, M., Littman, M. (eds.) Proceedings of the Twenty-seventh National Conference on Artificial Intelligence (AAAI'13). pp. 350–356. AAAI Press (2013)
37. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artificial Intelligence **187-188**, 52–89 (2012)
38. Gebser, M., Schaub, T.: Modeling and language extensions. AI Magazine **37**(3), 33–44 (2016)
39. Gelfond, M., Kahl, Y.: Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach. Cambridge University Press (2014)
40. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing **9**, 365–385 (1991)
41. Hamadi, Y., Jabbour, S., Sais, L.: ManySAT: A parallel SAT solver. Journal on Satisfiability, Boolean Modeling and Computation **6**, 245–262 (2009)
42. Janhunen, T.: Some (in)translatability results for normal logic programs and propositional theories. Journal of Applied Non-Classical Logics **16**(1-2), 35–86 (2006)
43. Janhunen, T.: Implementing stable-unstable semantics with asptools and clingo. In: Cheney, J., Perri, S. (eds.) Proceedings of the Twenty-fourth International Symposium on Practi-

cal Aspects of Declarative Languages (PADL'22). Lecture Notes in Computer Science, vol. 13165, pp. 135–153. Springer-Verlag (2022)
44. Janhunen, T., Kaminski, R., Ostrowski, M., Schaub, T., Schellhorn, S., Wanko, P.: Clingo goes linear constraints over reals and integers. Theory and Practice of Logic Programming **17**(5-6), 872–888 (2017)
45. Kaminski, R., Schaub, T.: On the foundations of grounding in answer set programming. Theory and Practice of Logic Programming **23**(6), 1138–1197 (2023)
46. Kaminski, R., Schaub, T., Wanko, P.: A tutorial on hybrid answer set solving with clingo. In: Ianni, G., Lembo, D., Bertossi, L., Faber, W., Glimm, B., Gottlob, G., Staab, S. (eds.) Proceedings of the Thirteenth International Reasoning Web Summer School (RW'17). Lecture Notes in Computer Science, vol. 10370, pp. 167–203. Springer-Verlag (2017)
47. Kaufmann, B., Leone, N., Perri, S., Schaub, T.: Grounding and solving in answer set programming. AI Magazine **37**(3), 25–32 (2016)
48. Kautz, H., Selman, B.: Planning as satisfiability. In: Neumann, B. (ed.) Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92). pp. 359–363. John Wiley & sons (1992)
49. KhudaBukhsh, A., Xu, L., Hoos, H., Leyton-Brown, K.: SATenstein: Automatically building local search SAT solvers from components. Artificial Intelligence **232**, 20–42 (2016)
50. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. ACM Transactions on Computational Logic **7**(3), 499–562 (2006)
51. Lierler, Y.: Constraint answer set programming: Integrational and translational (or SMT-based) approaches. Theory and Practice of Logic Programming **23**(1), 195–225 (2023)
52. Lifschitz, V.: Foundations of logic programming. In: Brewka, G. (ed.) Principles of Knowledge Representation. pp. 69–127. CSLI Publications (1996)
53. Lifschitz, V.: Answer set programming and plan generation. Artificial Intelligence **138**(1-2), 39–54 (2002)
54. Lifschitz, V.: Answer Set Programming. Springer-Verlag (2019)
55. Lindauer, M., Hoos, H., Leyton-Brown, K., Schaub, T.: Automatic construction of parallel portfolios via algorithm configuration. Artificial Intelligence **244**, 272–290 (2017)
56. Lloyd, J.: Foundations of Logic Programming. Symbolic Computation. Springer-Verlag (1987)
57. Lloyd, J.: Practical advantages of declarative programming. In: Alpuente, M., Barbuti, R., Ramos, I. (eds.) Proceedings of the Joint Conference on Declarative Programming (GULP-PRODE'94). pp. 3–17. Technical University of Valencia (1994)
58. Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M.: Algorithm portfolios based on cost-sensitive hierarchical clustering. In: Rossi, F. (ed.) Proceedings of the Twenty-third International Joint Conference on Artificial Intelligence (IJCAI'13). pp. 608–614. IJCAI/AAAI Press (2013)
59. Marek, V., Truszczyński, M.: Stable models and an alternative logic programming paradigm. In: Apt, K., Marek, V., Truszczyński, M., Warren, D. (eds.) The Logic Programming Paradigm: A 25-Year Perspective. pp. 375–398. Springer-Verlag (1999)
60. Marques-Silva, J., Sakallah, K.: GRASP: A search algorithm for propositional satisfiability. IEEE Transactions on Computers **48**(5), 506–521 (1999)
61. McCarthy, J.: Elaboration tolerance. http://jmc.stanford.edu/articles/elaboration/elaboration.pdf (1998)
62. Moskewicz, M., Madigan, C., Zhao, Y., Zhang, L., Malik, S.: Chaff: Engineering an efficient SAT solver. In: Proceedings of the Thirty-eighth Conference on Design Automation (DAC'01). pp. 530–535. ACM Press (2001)
63. Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. Annals of Mathematics and Artificial Intelligence **25**(3-4), 241–273 (1999)

64. Pearce, D.: Equilibrium logic. Annals of Mathematics and Artificial Intelligence **47**(1-2), 3–41 (2006)
65. Schlipf, J.: The expressive powers of the logic programming semantics. Journal of Computer and System Sciences **51**, 64–86 (1995)
66. Selman, B., Kautz, H., Cohen, B.: Noise strategies for improving local search. In: Hayes-Roth, B., Korf, R. (eds.) Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI'94). pp. 337–343. AAAI Press (1994)
67. Weinzierl, A., Taupe, R., Friedrich, G.: Advancing lazy-grounding ASP solving techniques — restarts, phase saving, heuristics, and more. Theory and Practice of Logic Programming **20**(5), 609–624 (2020)
68. Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K.: SATzilla: Portfolio-based algorithm selection for SAT. Journal of Artificial Intelligence Research **32**, 565–606 (2008)
69. Zhang, W., Sun, Z., Zhu, Q., Li, G., Cai, S., Xiong, Y., Zhang, L.: NLocalSAT: Boosting local search with solution prediction. In: Bessiere, C. (ed.) Proceedings of the Twenty-ninth International Joint Conference on Artificial Intelligence (IJCAI'20). pp. 1177–1183. ijcai.org (2020)

Knowledge Engineering for Graph Machine Learning*

Katja Hose

TU Wien, Austria
katja.hose@tuwien.ac.at

Abstract. Knowledge graphs and graph data in general are becoming more and more essential components of machine learning pipelines and use cases. Apart from data naturally occurring as graphs, such as social networks or Linked Data on the Web, the flexibility of the graph model and its ability to store data relationships explicitly enables integrating and exploiting data from very diverse sources. Nevertheless, to enable graph machine learning, we first need to understand and manage the graph data itself. In this paper, we will therefore look into the basics of modeling knowledge graphs using ontologies, managing graph data using graph stores, ensuring quality, managing provenance, and querying knowledge graphs using structured query languages.

1 Introduction

Knowledge graphs have emerged as powerful tools for representing structured knowledge in various domains. Their ability to model nearly arbitrary types of data have made knowledge graphs a universal tool for a broad range of applications. Hence, knowledge graph technologies have long been embraced as methods for storing and managing knowledge in a structured and easily accessible way. In fact, the term "knowledge graph" was coined by Google when they launched their Knowledge Graph in 2012. In the meantime, industry-scale knowledge graphs have grown to billions of entities and assertions about them [56]. While Google's Knowledge Graph fuels the information displayed in infoboxes that are shown along with the links to websites matching the user's keywords, Microsoft for instance does not only have a knowledge graph to power its search engine Bing but also manages the LinkedIn graph and the Academic graph – the latter covering information about people, publications, conferences, etc. Enterprise Knowledge Graphs [56] are another prominent example providing knowledge within the scope of a restricted organization or community. Other knowledge graphs, such as YAGO [31], DBpedia [45], or Wikidata [72], are freely accessible to the general public and mainly resulting from academic research and community efforts.

On the other hand, knowledge graphs also have roots in knowledge representation, description logics, and the Semantic Web [67]. And indeed there is a broad body of related work in this context. The vision of the Semantic Web, for instance, as outlined by Sir Tim Berners-Lee et al. [14] sketched an environment where intelligent agents could gather information from the Web (Linked Data) that comes along with semantics

* Large parts of this paper were first published in volume 13985, pages 3–15, 2023, by Springer Nature.

and links to other sources of information so that the agents would be able to understand the information that is being found and find more if needed. Such agents are also somewhat intelligent in the sense that they can use the information and apply logical reasoning to achieve a given task, and even independently "negotiate" with other agents to for example make appointments with a doctor. While this sounds similar to personal assistants (Siri, Echo, etc.), which are operating as black boxes on machine learning models and undisclosed data owned by a company, agents on the Semantic Web were supposed to work based on mostly Open Data accessible to everyone and in that way providing some kind of verifiability since, in principle, everybody could access the data that was used to solve a task.

Most recently, large language models, and in particular ChatGPT , have gained a lot of attention. Obviously, it is very appealing to simply formulate questions in natural language and receive elaborate and detailed replies that explain an extremely broad range of complex topics. While this system seems to be intelligent, it suffers from a similar problem as other large language models and machine learning approaches in general: the answer it returns is the most probable answer, it cannot be certain about its correctness. In the context of ChatGPT the latter is commonly referred to as hallucinations [13], i.e., the answer does not necessarily reflect reality but can be "made up". Furthermore, because of the way these systems are built, they operate as black boxes that cannot really explain how they arrived at a certain answer. Hence, verifying claims can so far only be done by accessing external information and methods. Moreover, reasoning capabilities of such systems are still limited. However, with advances in areas such as neurosymbolic AI, aiming to combine the strengths of logical reasoning and machine learning, we are slowly getting closer to arriving at intelligent systems.

In any case, approaches based on machine learning can only be as accurate, correct, and precise as the data they are trained upon; commonly referred to as the "garbage in, garbage out" principle. Hence, knowledge engineering, along with quality enhancement and provenance management, plays an important role to improve and maintain the quality of the data delivered as input to such a system for training. Graphs and knowledge graphs play a dual role in this contact by either providing raw input data for training or helping to access and extract additional information to enhance other datasets. In this context, machine learning approaches help construct knowledge graphs, expand, and improve them; NLP techniques, for example, are used to extract facts from natural language text, machine learning plays an inherent role in entity resolution and matching, etc.

In this paper, we therefore claim that more advances in knowledge engineering are needed to provide systems with accurate knowledge so that they can become more intelligent. Hence, we review current challenges in modeling and storing knowledge (Section 2), querying knowledge (Section 3), knowledge quality, provenance, and metadata (Section 4), and conclude the paper in Section 5.

2 Modeling and Storing Knowledge

While knowledge graphs are designed to capture information by representing entities (persons, places, concepts, etc.) as nodes, and relationships (bornIn, locatedIn, etc.)

https://openai.com/chatgpt

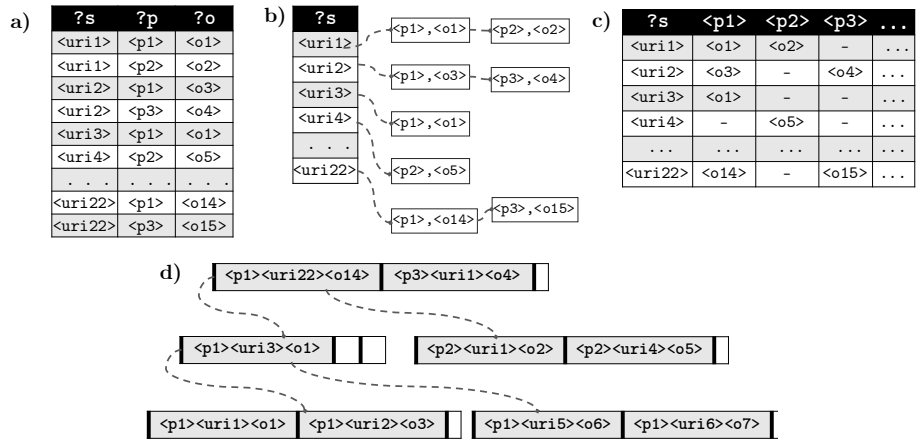


Fig. 1: Example RDF representations [65]: a) sorted file, b) hash map, c) property table, and d) B+ tree

as directed edges between them, they can be modeled using alternative data models; typically these are either RDF [1] or property graphs. Depending on the chosen data model and the users' information needs, knowledge graphs are typically managed and queried by triple stores (in case of RDF) or graph stores (in case of property graphs). Given path and reachability queries [74], e.g., determining the shortest path between a given set of nodes, property graphs are typically preferable whereas triple stores have advantages for graph pattern matching involving the labels of multiple nodes and edges.

Whereas commercial systems are available for both data models, the research body supporting RDF is larger. This might be due to the fact that RDF and the corresponding query language SPARQL [2] have long been W3C standards and well defined whereas property graphs are mostly driven by commercial systems and lack common standards – there are some recent advances though to define such standards [9, 17].

Still, even if there is a well-defined common data model, as it is the case for RDF, there are still several design options and alternative ways for representing and organizing the data – a few of them are illustrated in Figure 1. In general, the design space for RDF data representations can be organized in a three-dimensional space [65] defined by (i) subdivision (fragmentation and partitioning of the data), compression (how compressed is the data, how many bits are needed), and redundancy (does the system store multiple copies). In the end, it is the use case along with its query load, usage patterns, data characteristics, and other constraints that determine which design solution is the best. While some recommendations can be made based on an the expected

[1] https://www.w3.org/RDF/
In RDF edges are represented as triples (subject, predicate, object), where subject and object represent a pair of nodes and the predicate describes the relationship (edge label) between them.

[2] https://www.w3.org/TR/rdf-sparql-query/

workload [65], existing benchmarks often under-represent particular use cases, such as multi-hop traversals and existence checks, and therefore cannot help all use cases. Hence, developing more comprehensive benchmarks is one of the open challenges. Moreover, developing adaptive solutions that either manually or semi-automatically choose and adapt the design of an graph store to its needs – maybe in the sense of self-designing data structures [38] or self-organizing database systems [71], which both often involve machine learning approaches to guide optimizations – remains an interesting challenge for future work.

Apart from questions on how to store RDF in a single-server environment, there are many more use cases and storage paradigms for distributed environments. Such systems involve multiple servers in different configurations of autonomy [57] and range from cluster setups via federated scenarios to P2P systems. One of the well-understood challenges [66] in this context is that existing ecosystems are designed for tabular data or relational databases, while we are still lacking an ecosystem for big graph processing covering all components of a pipeline incl. extracting and converting data into a native graph format, integrating different datasets into graphs, enabling OLTP as well as OLAP operations on graphs, and finally also providing input for graph-based applications, for instance specialized in machine learning, business intelligence, scientific computing, or augmented reality. There are still many challenges to build such an ecosystem. Naturally, scalability is one of them and the question how new hardware can be used for this purpose. But there are also interoperability issues between the different graph models, query languages, and standards that hamper efficient use of graph data. There is also still very little research on efficiently supporting dynamic and streaming graphs.

Some use cases require to integrate large amounts of heterogeneous data. This can include converting knowledge from one graph model to the other [3,10,43]. On the other hand, this also includes converting data originally provided in other formats into graphs, e.g., using R2ML [24] or RML [18], followed by some integration and homogenization to ensure quality. However, instead of transforming the data directly, knowledge graphs can also be used as a (virtual) integration layer – similar to traditional virtual data integration scenarios [57]. This scenario, often applied in the context of data fabrics and data lakes [53], uses knowledge graphs to provide an integrated interface where information can be accessed under a unified view. When the data then needs to be accessed, it can be searched, retrieved, and if needed converted on demand from the sources. Yet, finding and matching entities across different datasets is a challenging task.

When converting all data into an integrated knowledge graph directly, it can be queried in a single system – not only with standard queries. There are some works [25,54] on setting up semantic data warehouses – as illustrated in Figure 2 incl. spatio-temporal extensions. The setup is similar to traditional data warehouses on relational data; there needs to be an ETL process as well as a way to define data cubes along with several analytical dimensions. The data can then be analyzed, cleansed, further data can be included, we can chose to keep track of metadata and provenance, links to external data sources can be created, and finally the data can be published, for instance on the Semantic Web. This is related in spirit to the lack of graph processing ecosystems mentioned above and therefore meets similar challenges.

Alternatively, knowledge can also be organized in federations of data sources ac-

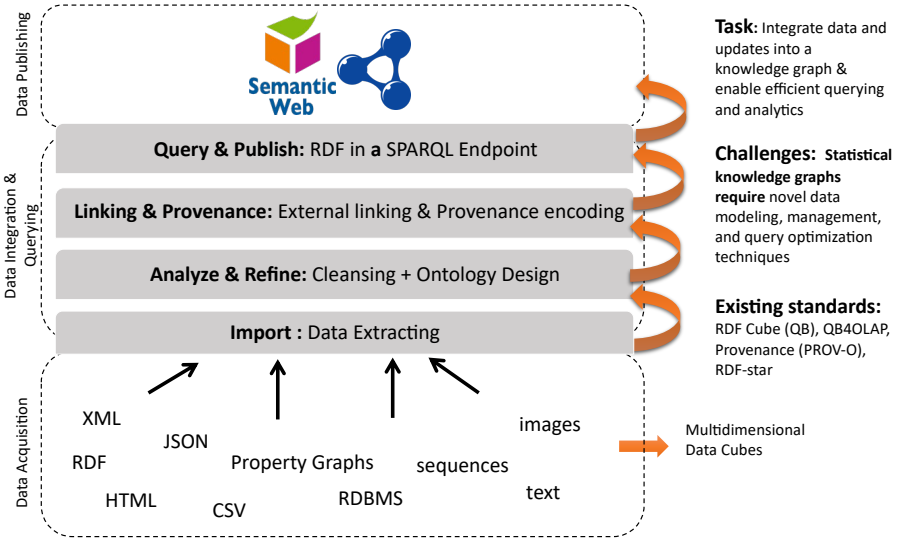


Fig. 2: Semantic Data Warehousing [25, 54]

cessible over the Web, i.e., each provider hosts a public endpoint (typically SPARQL) and makes it accessible and queryable to everyone. While SPARQL queries can be formulated over a combination of data from multiple remote sources, it is the task of the query optimizer (federation engine) to identify which publicly available sources should be queried with which part of the query to compute the final answer. An interesting observation here is that publishing data and making it available in this way is very easy as the publishers do not need to conform to a common integrated schema. However, this comes at the expense of query formulation and optimization, which then is considerably more complex. To formulate a query, users themselves have to know how the information in the different sources are connected – whereas this would typically be done when defining a common schema or table in a traditional relational database scenario. Likewise, the query optimizer has to decide for each query individually (on individual node-to-node connections) which data sources to select – whereas in a traditional relational setup such information could be captured by mappings valid for all tuples in a table, which therefore allows for preselecting sources.

Finally, solutions building upon P2P systems can be used to share data. Depending on the network structure and the replication rate, such systems can keep data available despite node failures. Some systems assume a fixed or previously known set of peers with some central agreement on where to store which kind of data [41] while others employ the unstructured P2P paradigm, where peers have the largest degree of autonomy [5, 6], i.e., there is no global knowledge or control and peers are free to join with their data.

As discussed in this section, there are many ways to model, integrate, store, and manage knowledge. Each of these options has different advantages, disadvantages, and goals, e.g., if access efficiency is more important, then a single-server or cluster-system might be preferable – if the main goal is to keep the data available, then P2P systems

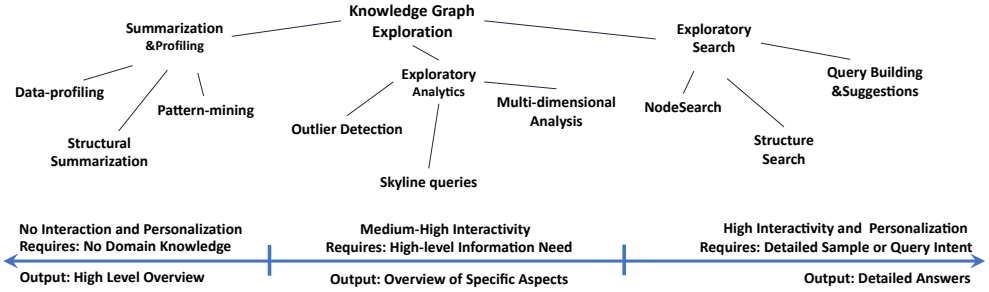


Fig. 3: Knowledge Graph Exploration Techniques [47]

are a good choice. Nevertheless, each of these systems faces a range of open challenges in each of the steps mentioned above.

3 Querying Knowledge

The way in which knowledge is queried very much depends on the chosen data model and the way the data is physically stored. Most commonly either Cypher or SPARQL queries are used as query languages. Then, similar to traditional relational database systems, it is the task of the query optimizer to find an efficient query execution plan to answer the query.

However, many users are not familiar with the details, content, and schema of a knowledge graph and therefore have difficulties formulating structured queries. To help such users, the literature has proposed exploratory query techniques and the query-by-example paradigm [47, 48]. In this case, users do not formulate structured queries directly but provide the system with examples of potential answers – the system then tries to reverse engineer a query from the desired output, executes it, and presents the results to the user who can then iteratively refine the query until the information need is met. This is even possible for complex setups incl. analytical queries over statistical knowledge graphs [46]. Exploratory techniques for knowledge graphs cover a broad range of methods (see Figure 3) that include data profiling [1] as well as skyline queries [42].

Assuming that the user was able to formulate a structured query that expresses the information need, there is a broad range of query optimization techniques depending on the architecture of the system. In this respect, we can distinguish the architectures illustrated in Figure 4: (i) centralized systems, where all data is stored on a single server, (ii) client/server architectures, where we have multiple clients querying the data on a single server, (iii) parallel systems, where we have a cluster of servers on which the knowledge graph is partitioned and queried exploiting parallel computation, (iv) federated systems, where we have multiple independent data sources with knowledge graphs and queries spanning multiple of these knowledge graphs, and (v) P2P systems, where we have a number of autonomous peers that might join and leave the network at any time.

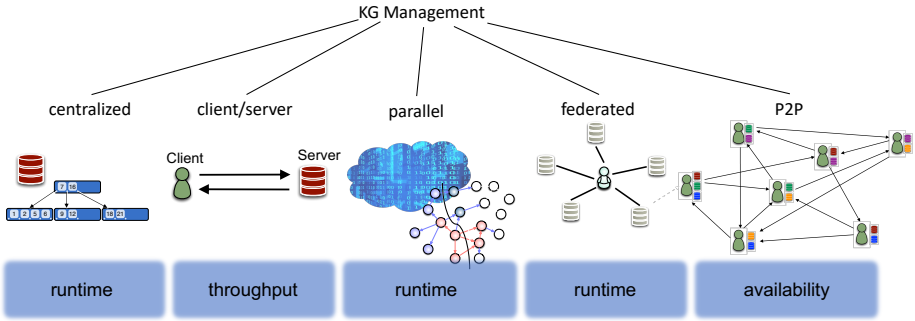


Fig. 4: Knowledge Graph Management Architectures and Optimization Goals

Each of these architectures is tailored for providing access to knowledge in a different use case. While in all setups, the query optimizer aims at answering queries efficiently, optimization strategies in particular systems might have slightly different objective functions. In case of single-server centralized systems, the goal is typically to minimize query execution time by exploiting local indexing structures, access paths, caching, precomputation, etc. [20, 20, 35, 37, 40, 51, 73]. This is similar in spirit to parallel systems, where the goal is to additionally exploit parallel computations in a cluster of machines [19, 34], e.g., by partitioning graphs to execute parts of a query independently and in parallel. Client/server architectures, on the other hand, also aim at answering queries efficiently. However, since such systems might suffer from times where more clients issue queries than the server can handle, the optimization goal is often not to focus on optimizing individual queries but instead the overall throughput, i.e., the number of queries that are successfully completed in a given period of time. This might entail that some expensive operations are "outsourced" from the server to the client, which might slow down the execution of an individual query but by freeing the resources on the server more queries can successfully complete within the same period of time [4, 12].

Another optimization goal can be witnessed in systems building upon unstructured P2P architectures [5, 8]. Since peers offering access to unique knowledge might leave the network at some point in time, the system stores copies of the data on other peers. Of course, the system tries to execute queries as efficiently as possible but more important than that is to keep the data available despite node failures. Due to the lack of central control, such systems then either resort to flooding the network with a particular query to find relevant data, i.e., sending messages to all known peers, or employing some kind of indexes that capture summaries of knowledge available at remote peers and query them directly.

Finally, federated setups [33, 36, 39, 52, 68, 69] are commonly used on the Semantic Web and Linked Open Data [27] , where hundreds of publicly accessibly servers provide query access to billions of facts covering a diverse range of topics. Such data sources typically offer access via SPARQL endpoints, i.e., they offer interfaces that given a SPARQL query provide corresponding answers executed over the local knowledge graph.

`http://cas.lod-cloud.net/`

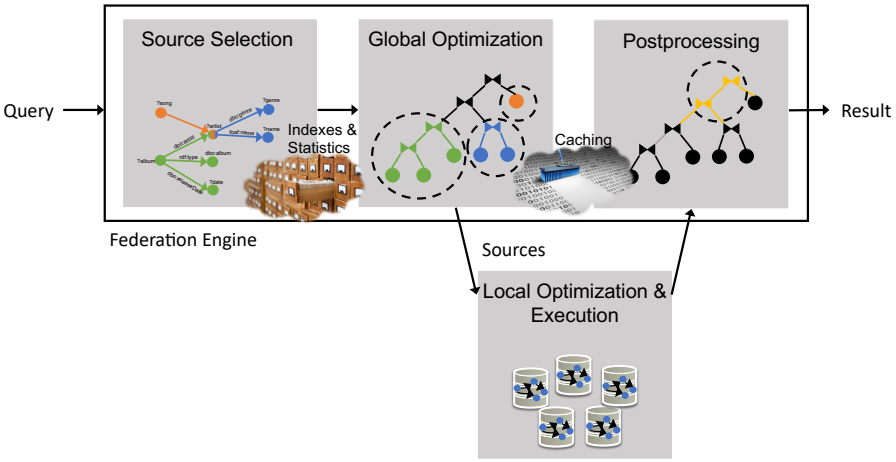


Fig. 5: Federated Query Processing

Then, given a SPARQL query spanning multiple knowledge graphs, hosted on multiple remote SPARQL endpoints, the task of a federation engine (illustrated in Figure 5) is to analyze the given query, identify which endpoints provide relevant data for the different parts of the query, decide which parts of the query should be executed on which endpoint and in which order, have the subqueries executed on the chosen endpoints, receive the partial results, and combine them into the final answer to the query.

There are many factors that influence which architecture is the most appropriate for a particular use case, e.g., how many computing resources do we have, who owns the data, which kind of access is allowed, etc. On the other hand, the types of queries and the information need of users plays a decisive role. Connectivity and path queries or logic reasoning, for instance, benefit from different optimization strategies, data models, and indexes than regular graph pattern matching queries. Finally, if knowledge graphs are to be integrated into machine learning pipelines, then a centralized or cluster-based architecture and ecosystem is often preferable.

While there is plenty of related work in each of these setups and scenarios, there are still many open challenges; some of them with respect to traditional issues such as cardinality estimation, cost functions, etc. Other challenges originate from heterogeneity and interoperability of data models, interfaces, use cases, etc. The OneGraph vision [44], for instance, sketches a scenario where the data model no longer determines the query languages and would allow formulating Cypher queries over an RDF store. On the other hand, heterogeneous federations [29, 50, 51] have been explored with the goal of bridging the gap between different query interfaces and architectures. Still, in many aspects research is just about to begin looking into efficient solutions to the wide variety of challenges in this context.

4 Knowledge Quality and Metadata

Intuitively, any system – using machine learning or not – will not be able to produce high-quality results ("garbage in, garbage out") if it is not provided with high-quality input data. So far research on aspects such as data cleaning [16] and data profiling [1] has mainly focused on relational data and is not straightforward to apply to knowledge graphs. This is amplified by the issue that – in contrast to relational data – knowledge graphs are valid without the need for a strict schema or compliance to an ontology. Hence, a first step to ensure quality is to extract schemas from knowledge graphs and check conformance. Whereas there exist standards, such as OWL and RDFS, to define ontologies and schema constraints for knowledge graphs modeled in RDF, similar work on property graphs has so far been hampered by the lack of a well-defined standard, which however might come soon [9].

Nevertheless, while OWL and RDFS have been developed for capturing the meaning of data by defining proper classes, hierarchies, and constraints, SHACL has been proposed more recently as a standard to define constraints on the structure of knowledge graphs – without the need to define a proper full-fledged ontology and capture the meaning of the data. SHACL allows to define graph patterns, referred to as shapes, along with constraints that subgraphs matching the patterns/shapes should fulfill. While SHACL is becoming more and more adopted by the community, it still remains a challenge to avoid having to define shapes manually [62] but instead being offered semi-automatic solutions for creating them given a knowledge graph as input. While mining shapes from large knowledge graphs meets scalability issues, it is also important to mine meaningful shapes [63] and avoid spurious ones, i.e., those that do not occur frequently or are fulfilled by only a small proportion of matching subgraphs. Once determined, such shapes can not only be used to create validation reports but they can also be used in a more interactive fashion in a similar way as mined association rules [22], e.g., to help experts find outliers and erroneous information so that the data can be corrected and the quality can be improved [64].

Another way of improving quality and trust in knowledge is to provide metadata. While metadata in property graphs can be expressed by adding attributes to nodes and edges, this is not straightforward for knowledge graphs. The latter require special constructs, such as reification, singleton properties [55], named graphs [15], or RDF-star. While reification leads to a large increase in the number of triples (because subject, predicate, and object of the original triple are separated into their own triples), singleton properties (instantiating a unique subproperty for each triple with metadata) and named graph solutions (in the worst case creating a separate named graphs for each single triple) typically also suffer from scalability issues and require verbose query constructs since existing engines are not designed to efficiently support such use cases. On the other hand, RDF-star is proposing to nest triples, i.e., to use a complete triple on subject or object position of another triple. While this is very elegant from a modeling perspective,

---

http://www.w3.org/TR/owl2-overview/
http://www.w3.org/TR/rdf-schema/
https://www.w3.org/TR/shacl/
https://w3c.github.io/rdf-star/

it poses several challenges on data organization and querying since nesting has not yet been a typical requirement. Still, many triple stores do already support RDF-star so that it can already be used in practice.

Provenance, in the sense of explaining the origin of data, is an important kind of metadata. In this sense it is often desired to capture information about who created the data, how and when it was obtained, how it was processed, etc. In RDF, such workflow provenance [21, 26] can for instance be encoded using the PROV-O ontology, which offers several classes with well defined meaning for this purpose. Another type of provenance, how-provenance [11, 23, 30], describes how an answer to a particular query was derived from a given input dataset. This approach allows to directly trace down the input tuples/triples/edges that were combined to derive a particular answer to a query – in addition, how-provenance also returns a polynomial describing how these tuples/triples/edges have been combined for a given query answer. In general, all flavors of provenance help explain answers to structured queries and in doing so increase the trust users can have in a system. To the best of our knowledge, however, there is currently no system for knowledge graphs combining workflow provenance with how-provenance.

Finally, it is also important to highlight that, although knowledge graphs are mostly considered to be static and not changing, in reality knowledge changes over time [32, 61] so that we can expect multiple versions of a knowledge graph. This is often referred to as knowledge evolution or dynamic knowledge graphs. However, in current practice, older versions of a knowledge graph typically disappear and only the latest version remains available. Instead of losing older versions, some systems attempt to capture previous versions and keep them retrievable [7, 58, 70]. However, existing engines have not been designed for this kind of usage and the current state of the art still has deficiencies in efficiency and coverage of use cases [59, 60].

While quality, provenance, and knowledge evolution are all very important topics discussed in the literature from different perspectives, most of today's engines do not yet come with efficient native support. Recent developments suggest that RDF-star might soon become the standard way of encoding and supporting metadata. Still, existing methods building upon it do not (yet) support the full spectrum and even though many triple stores already support RDF-star, they have limitations regarding full compliance and efficiency [2]. Hence, there are many open challenges with supporting quality on all levels incl. optimizing the data layout, encoding and computing metadata, quality assurance and validation, query optimization, etc.

5 Conclusion

Knowledge engineering in all its different aspects discussed in this paper is the foundation for making machine learning based systems more intelligent and enabling Data Science [28, 49]; it provides the structured and interconnected knowledge that enables intelligent systems to access, process, semantically integrate, interpret, discover, reason about knowledge and making informed decisions. In this sense, knowledge also builds the foundation for explainable systems that will help us get away from today's black

https://www.w3.org/TR/prov-o/

box systems that demand blind trust. In this vibrant area, research is currently moving very fast in knowledge engineering as well as machine learning. While today we are still witnessing big challenges and open issues – as sketched in this paper – it will be interesting to witness how research will progress in the next couple of years.

References

1. Abedjan, Z., Golab, L., Naumann, F., Papenbrock, T.: Data Profiling. Synthesis Lectures on Data Management, Morgan & Claypool Publishers (2018)
2. Abuoda, G., Aebeloe, C., Dell'Aglio, D., Keen, A., Hose, K.: StarBench: Benchmarking RDF-star Triplestores. In: QuWeDa/MEPDaW@ISWC. CEUR Workshop Proceedings, vol. 3565, pp. 34–49. CEUR-WS.org (2023)
3. Abuoda, G., Dell'Aglio, D., Keen, A., Hose, K.: Transforming RDF-star to Property Graphs: A Preliminary Analysis of Transformation Approaches. In: QuWeDa@ISWC. CEUR Workshop Proceedings, vol. 3279, pp. 17–32. CEUR-WS.org (2022)
4. Aebeloe, C., Keles, I., Montoya, G., Hose, K.: Star Pattern Fragments: Accessing Knowledge Graphs through Star Patterns. CoRR **abs/2002.09172** (2020)
5. Aebeloe, C., Montoya, G., Hose, K.: A Decentralized Architecture for Sharing and Querying Semantic Data. In: ESWC. Lecture Notes in Computer Science, vol. 11503, pp. 3–18. Springer (2019)
6. Aebeloe, C., Montoya, G., Hose, K.: Decentralized Indexing over a Network of RDF Peers. In: ISWC (1). Lecture Notes in Computer Science, vol. 11778, pp. 3–20. Springer (2019)
7. Aebeloe, C., Montoya, G., Hose, K.: ColChain: Collaborative Linked Data Networks. In: WWW. pp. 1385–1396. ACM / IW3C2 (2021)
8. Aebeloe, C., Montoya, G., Hose, K.: Optimizing SPARQL queries over decentralized knowledge graphs. Semantic Web **14**(6), 1121–1165 (2023)
9. Angles, R., Bonifati, A., Dumbrava, S., Fletcher, G., Green, A., Hidders, J., Li, B., Libkin, L., Marsault, V., Martens, W., Murlak, F., Plantikow, S., Savkovic, O., Schmidt, M., Sequeda, J., Staworko, S., Tomaszuk, D., Voigt, H., Vrgoc, D., Wu, M., Zivkovic, D.: PG-Schema: Schemas for Property Graphs. Proc. ACM Manag. Data **1**(2), 198:1–198:25 (2023)
10. Angles, R., Thakkar, H., Tomaszuk, D.: Mapping RDF Databases to Property Graph Databases. IEEE Access **8**, 86091–86110 (2020)
11. Asma, Z., Hernández, D., Galárraga, L., Flouris, G., Fundulaki, I., Hose, K.: NPCS: Native Provenance Computation for SPARQL. In: WWW. ACM (2024)
12. Azzam, A., Aebeloe, C., Montoya, G., Keles, I., Polleres, A., Hose, K.: WiseKG: Balanced Access to Web Knowledge Graphs. In: WWW. pp. 1422–1434 (2021)
13. Bang, Y., Cahyawijaya, S., Lee, N., Dai, W., Su, D., Wilie, B., Lovenia, H., Ji, Z., Yu, T., Chung, W., Do, Q.V., Xu, Y., Fung, P.: A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity. CoRR **abs/2302.04023** (2023)
14. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American **284**(5), 28–37 (2001)
15. Carroll, J.J., Bizer, C., Hayes, P.J., Stickler, P.: Named graphs. J. Web Semant. **3**(4), 247–267 (2005)
16. Chu, X., Ilyas, I.F., Krishnan, S., Wang, J.: Data Cleaning: Overview and Emerging Challenges. In: SIGMOD Conference. pp. 2201–2206. ACM (2016)
17. Deutsch, A., Francis, N., Green, A., Hare, K., Li, B., Libkin, L., Lindaaker, T., Marsault, V., Martens, W., Michels, J., Murlak, F., Plantikow, S., Selmer, P., van Rest, O., Voigt, H., Vrgoc, D., Wu, M., Zemke, F.: Graph pattern matching in GQL and SQL/PGQ. In: SIGMOD Conference. pp. 2246–2258. ACM (2022)

18. Dimou, A., Sande, M.V., Colpaert, P., Verborgh, R., Mannens, E., de Walle, R.V.: RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In: Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014. CEUR Workshop Proceedings, vol. 1184. CEUR-WS.org (2014), https://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf
19. Galárraga, L., Hose, K., Schenkel, R.: Partout: a distributed engine for efficient RDF processing. In: WWW (Companion Volume). pp. 267–268. ACM (2014)
20. Galárraga, L., Jakobsen, K.A., Hose, K., Pedersen, T.B.: Answering Provenance-Aware Queries on RDF Data Cubes Under Memory Budgets. In: ISWC. Lecture Notes in Computer Science, vol. 11136, pp. 547–565. Springer (2018)
21. Galárraga, L., Mathiassen, K.A.M., Hose, K.: QBOAirbase: The European Air Quality Database as an RDF Cube. In: ISWC (Posters, Demos & Industry Tracks). CEUR Workshop Proceedings, vol. 1963. CEUR-WS.org (2017)
22. Galárraga, L., Teflioudi, C., Hose, K., Suchanek, F.M.: Fast rule mining in ontological knowledge bases with AMIE+. VLDB J. **24**(6), 707–730 (2015)
23. Geerts, F., Unger, T., Karvounarakis, G., Fundulaki, I., Christophides, V.: Algebraic Structures for Capturing the Provenance of SPARQL Queries. J. ACM **63**(1), 7:1–7:63 (2016)
24. Group, R.W.: R2RML: RDB to RDF Mapping Language: http://www.w3.org/TR/r2rml/. http://www.w3.org/2001/sw/rdb2rdf/ (2014)
25. Gür, N., Pedersen, T.B., Zimányi, E., Hose, K.: A foundation for spatial data warehouses on the semantic web. Semantic Web **9**(5), 557–587 (2018)
26. Hansen, E.R., Lissandrini, M., Ghose, A., Løkke, S., Thomsen, C., Hose, K.: Transparent Integration and Sharing of Life Cycle Sustainability Data with Provenance. In: ISWC. pp. 378–394 (2020)
27. Harth, A., Hose, K., Schenkel, R. (eds.): Linked Data Management. Chapman and Hall/CRC (2014)
28. Helali, M., Vashisth, S., Carrier, P., Hose, K., Mansour, E.: Linked Data Science Powered by Knowledge Graphs. CoRR **abs/2303.02204** (2023)
29. Heling, L., Acosta, M.: Federated SPARQL query processing over heterogeneous linked data fragments. In: WWW. pp. 1047–1057. ACM (2022)
30. Hernández, D., Galárraga, L., Hose, K.: Computing How-Provenance for SPARQL Queries via Query Rewriting. Proc. VLDB Endow. **14**(13), 3389–3401 (2021)
31. Hoffart, J., Suchanek, F.M., Berberich, K., Lewis-Kelham, E., de Melo, G., Weikum, G.: YAGO2: exploring and querying world knowledge in time, space, context, and many languages. In: WWW (Companion Volume). pp. 229–232. ACM (2011)
32. Hose, K.: Knowledge Graph (R)Evolution and the Web of Data. In: MEPDaW@ISWC. CEUR Workshop Proceedings, vol. 3225, pp. 1–7. CEUR-WS.org (2021)
33. Hose, K., Schenkel, R.: Towards benefit-based RDF source selection for SPARQL queries. In: SWIM. p. 2. ACM (2012)
34. Hose, K., Schenkel, R.: WARP: workload-aware replication and partitioning for RDF. In: ICDE Workshops. pp. 1–6. IEEE Computer Society (2013)
35. Ibragimov, D., Hose, K., Pedersen, T.B., Zimányi, E.: Towards exploratory OLAP over linked open data - A case study. In: BIRTE. Lecture Notes in Business Information Processing, vol. 206, pp. 114–132. Springer (2014)
36. Ibragimov, D., Hose, K., Pedersen, T.B., Zimányi, E.: Processing aggregate queries in a federation of SPARQL endpoints. In: ESWC. Lecture Notes in Computer Science, vol. 9088, pp. 269–285. Springer (2015)
37. Ibragimov, D., Hose, K., Pedersen, T.B., Zimányi, E.: Optimizing Aggregate SPARQL Queries Using Materialized RDF Views. In: ISWC. Lecture Notes in Computer Science, vol. 9981, pp. 341–359 (2016)

38. Idreos, S., Dayan, N., Qin, W., Akmanalp, M., Hilgard, S., Ross, A., Lennon, J., Jain, V., Gupta, H., Li, D., Zhu, Z.: Design Continuums and the Path Toward Self-Designing Key-Value Stores that Know and Learn. In: CIDR. www.cidrdb.org (2019)
39. Jakobsen, A.L., Montoya, G., Hose, K.: How diverse are federated query execution plans really? In: ESWC (Satellite Events). Lecture Notes in Computer Science, vol. 11762, pp. 105–110. Springer (2019)
40. Jakobsen, K.A., Andersen, A.B., Hose, K., Pedersen, T.B.: Optimizing RDF data cubes for efficient processing of analytical queries. In: COLD. CEUR Workshop Proceedings, vol. 1426. CEUR-WS.org (2015)
41. Kaoudi, Z., Koubarakis, M., Kyzirakos, K., Miliaraki, I., Magiridou, M., Papadakis-Pesaresi, A.: Atlas: Storing, updating and querying RDF(S) data on top of DHTs. J. Web Semant. **8**(4), 271–277 (2010)
42. Keles, I., Hose, K.: Skyline queries over knowledge graphs. In: ISWC (1). Lecture Notes in Computer Science, vol. 11778, pp. 293–310. Springer (2019)
43. Khayatbashi, S., Ferrada, S., Hartig, O.: Converting property graphs to RDF: a preliminary study of the practical impact of different mappings. In: GRADES-NDA@SIGMOD. pp. 10:1–10:9. ACM (2022)
44. Lassila, O., Schmidt, M., Hartig, O., Bebee, B., Bechberger, D., Broekema, W., Khandelwal, A., Lawrence, K., López-Enríquez, C., Sharda, R., Thompson, B.B.: The OneGraph vision: Challenges of breaking the graph model lock-in. Semantic Web **14**(1), 125–134 (2023)
45. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web **6**(2), 167–195 (2015)
46. Lissandrini, M., Hose, K., Pedersen, T.B.: Example-driven exploratory analytics over knowledge graphs. In: EDBT. pp. 105–117. OpenProceedings.org (2023)
47. Lissandrini, M., Mottin, D., Hose, K., Pedersen, T.B.: Knowledge graph exploration systems: are we lost? In: CIDR. www.cidrdb.org (2022)
48. Lissandrini, M., Mottin, D., Palpanas, T., Velegrakis, Y.: Data Exploration Using Example-Based Methods. Synthesis Lectures on Data Management, Morgan & Claypool Publishers (2018)
49. Mansour, E., Srinivas, K., Hose, K.: Federated Data Science to Break Down Silos. SIGMOD Rec. **50**(4), 16–22 (2021)
50. Montoya, G., Aebeloe, C., Hose, K.: Towards efficient query processing over heterogeneous RDF interfaces. In: ISWC (Best Workshop Papers). Studies on the Semantic Web, vol. 36, pp. 39–53. IOS Press (2018)
51. Montoya, G., Keles, I., Hose, K.: Analysis of the effect of query shapes on performance over LDF interfaces. In: QuWeDa@ISWC. CEUR Workshop Proceedings, vol. 2496, pp. 51–66. CEUR-WS.org (2019)
52. Montoya, G., Skaf-Molli, H., Hose, K.: The Odyssey Approach for Optimizing Federated SPARQL Queries. In: ISWC. Lecture Notes in Computer Science, vol. 10587, pp. 471–489. Springer (2017)
53. Nargesian, F., Pu, K.Q., Bashardoost, B.G., Zhu, E., Miller, R.J.: Data lake organization. IEEE Trans. Knowl. Data Eng. **35**(1), 237–250 (2023)
54. Nath, R.P.D., Hose, K., Pedersen, T.B., Romero, O.: SETL: A programmable semantic extract-transform-load framework for semantic data warehouses. Inf. Syst. **68**, 17–43 (2017)
55. Nguyen, V., Bodenreider, O., Sheth, A.P.: Don't like RDF reification?: making statements about statements using singleton property. In: WWW (2014)
56. Noy, N.F., Gao, Y., Jain, A., Narayanan, A., Patterson, A., Taylor, J.: Industry-scale knowledge graphs: lessons and challenges. Commun. ACM **62**(8), 36–43 (2019)
57. Özsu, M.T., Valduriez, P.: Principles of Distributed Database Systems, 4th Edition. Springer (2020)

58. Pelgrin, O., Galárraga, L., Hose, K.: Towards fully-fledged archiving for RDF datasets. Semantic Web **12**(6), 903–925 (2021)
59. Pelgrin, O., Taelman, R., Galárraga, L., Hose, K.: Scaling Large RDF Archives To Very Long Histories. In: ICSC. pp. 41–48. IEEE (2023)
60. Pelgrin, O., Taelman, R., Galárraga, L., Hose, K.: The Need for Better RDF Archiving Benchmarks. In: QuWeDa/MEPDaW@ISWC. CEUR Workshop Proceedings, vol. 3565, pp. 50–54. CEUR-WS.org (2023)
61. Polleres, A., Pernisch, R., Bonifati, A., Dell'Aglio, D., Dobriy, D., Dumbrava, S., Etcheverry, L., Ferranti, N., Hose, K., Jiménez-Ruiz, E., Lissandrini, M., Scherp, A., Tommasini, R., Wachs, J.: How Does Knowledge Evolve in Open Knowledge Graphs? TGDK **1**(1), 11:1–11:59 (2023)
62. Rabbani, K., Lissandrini, M., Hose, K.: SHACL and ShEx in the Wild: A Community Survey on Validating Shapes Generation and Adoption. In: WWW (Companion Volume). pp. 260–263. ACM (2022)
63. Rabbani, K., Lissandrini, M., Hose, K.: Extraction of Validating Shapes from very large Knowledge Graphs. Proc. VLDB Endow. **16**(5), 1023–1032 (2023)
64. Rabbani, K., Lissandrini, M., Hose, K.: SHACTOR: Improving the Quality of Large-Scale Knowledge Graphs with Validating Shapes. In: SIGMOD Conference Companion. pp. 151–154. ACM (2023)
65. Sagi, T., Lissandrini, M., Pedersen, T.B., Hose, K.: A design space for RDF data representations. VLDB J. **31**(2), 347–373 (2022)
66. Sakr, S., Bonifati, A., Voigt, H., Iosup, A., Ammar, K., Angles, R., Aref, W.G., Arenas, M., Besta, M., Boncz, P.A., Daudjee, K., Valle, E.D., Dumbrava, S., Hartig, O., Haslhofer, B., Hegeman, T., Hidders, J., Hose, K., Iamnitchi, A., Kalavri, V., Kapp, H., Martens, W., Özsu, M.T., Peukert, E., Plantikow, S., Ragab, M., Ripeanu, M., Salihoglu, S., Schulz, C., Selmer, P., Sequeda, J.F., Shinavier, J., Szárnyas, G., Tommasini, R., Tumeo, A., Uta, A., Varbanescu, A.L., Wu, H., Yakovets, N., Yan, D., Yoneki, E.: The future is big graphs: a community view on graph processing systems. Commun. ACM **64**(9), 62–71 (2021)
67. Scherp, A., Groener, G., Skoda, P., Hose, K., Vidal, M.E.: Semantic Web: Past, Present, and Future. TGDK (2024)
68. Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: FedX: A Federation Layer for Distributed Query Processing on Linked Open Data. In: ESWC. Lecture Notes in Computer Science, vol. 6644, pp. 481–486. Springer (2011)
69. Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: FedX: Optimization Techniques for Federated Query Processing on Linked Data. In: ISWC. Lecture Notes in Computer Science, vol. 7031, pp. 601–616. Springer (2011)
70. Taelman, R., Mahieu, T., Vanbrabant, M., Verborgh, R.: Optimizing storage of RDF archives using bidirectional delta chains. Semantic Web (2021)
71. Varadarajan, R., Bharathan, V., Cary, A., Dave, J., Bodagala, S.: DBDesigner: A customizable physical design tool for Vertica Analytic Database. In: ICDE. pp. 1084–1095. IEEE Computer Society (2014)
72. Vrandecic, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10), 78–85 (2014)
73. Zervakis, L., Setty, V., Tryfonopoulos, C., Hose, K.: Efficient Continuous Multi-Query Processing over Graph Streams. In: EDBT. pp. 13–24. OpenProceedings.org (2020)
74. Zhang, C., Bonifati, A., Özsu, M.T.: An overview of reachability indexes on graphs. In: SIGMOD Conference Companion. pp. 61–68. ACM (2023)

An exercise-based introduction to ethical issues of AI

Erich Prem[0000-0002-4380-3722]

University of Vienna, Institute of Philosophy, A-1010 Vienna, Austria and
eutema GmbH, Lindengasse 43/13, A-1070 Vienna, Austria
erich.prem@univie.ac.at

Abstract. This paper introduces into ethical issues raised by AI systems and proposed solutions to address them. It uses introductory exercises that present an ethical issue and questions that provide guidance for study and debate. The included topics address fairness and bias, user information, trolley problems, and freedom of speech. The paper concludes with a brief introduction into ethical frameworks and proposed tools to address ethical issues of AI systems.

Keywords: First Keyword, Second Keyword, Third Keyword.

1 A Short Introduction

1.1 AI Ethics

If it is true that – as Arthur Schopenhauer suggested already in 1840 – compassion is the basis of morality, then AI is clearly facing a challenge that is too big [1]. Given the current state-of-the-art, compassion simply is not a stronghold of artificial systems. However, as AI systems increasingly participate in processes of decision-making about humans, it becomes increasingly necessary to ensure that such decisions are taken responsibly and that they are beneficial for individuals and society. This has created an enormous interest in the ethics of AI. This paper provides a short introduction into some key concerns, approaches, and future challenges using a set of six exercises relevant for AI ethics. Let us first briefly introduce ethics.

The philosophical discipline of ethics is concerned with questions about what a person should do from the point of view of morality. The US American moral philosopher Bernard Gert defined morality as "*an informal public system applying to all rational persons, governing behaviour that affects others, and includes what are commonly known as the moral rules, ideals and virtues and has the lessening of evil and harm as its goal." [7]* In this short characterization, Gert makes several important distinctions that are worth pointing out:

- As an informal public system, moral judgments are usually those of a social group such as a tribe, village, or a nation state. They arise from often tacit expectations about how people should act. This system may include widely held believes (e.g., it is bad not to help people in need) or religious rules (e.g., 'thou shalt not steal') that are somehow formalised but may require contextualisation and are often not directly enforced. Legal regulation partially overlaps with

widely help moral beliefs. However, there are laws that are not usually considered morally relevant, e.g., regulations concerning the shapes and symbols used for traffic signs. And vice versa, the law does not address everything morally relevant, such as the question when we should tell the truth to our friends or, perhaps more relevant, what precisely constitutes bad business practice from a moral point of view.

- Morality according to Gert concerns rational persons. It is therefore not directly applicable to children, the mentally disabled, to animals, robots (unless we think they are rational persons), and perhaps intoxicated people. In ethics, the focus is on people with the ability to make judgements about the consequences of their actions for others.
- Common virtues include truthfulness, courage, honesty, impartiality, reliability, and other character traits that we would often classify as positive or beneficial. Ideals are generally accepted ideas – often about our society and its institution – such as fairness or justice. Some common harms include death, pain, disability, loss of freedom, loss of pleasure, loss of rights and many more. What precisely constitutes a moral harm may drastically change over time. For example, the degree to which a person should be covered when going for a swim on a public beach has kept changing over the centuries and shows great regional variation.

1.2 Modelling people

The art and science of modelling lies at the heart of computer science. Different from other fields of science and technology that are often focused on a particular type of system (e.g., living systems in biology), computer science is extremely broad in the type of models that it uses and in with what these models are concerned. Models can be explicitly formulated, e.g., a linear model of a specific part of a physical or economic system. Often, they are more implicit, e.g., when a computer program operates a device and only implicitly is based on a model of the device's environment or its interactions with users. Computer programmers may have to model cars, homes, financial markets, power plants, airplanes as well as cows, trees, or novels. Most ethical issues, however, arise when modelling people.

An important reason for modelling people is to systematically organize knowledge about them. Such knowledge may form the basis of categorising people into similar groups. Another reason is to predict something about them, for example the language in which they would like to be given information or the style of music that they are likely to appreciate. Thirdly, it can be useful to control the behaviour of people, for example in guiding them to a desired location or improving their fitness.

The straightforward fact of modelling people often already creates ethical issues. At the simplest level, even striving to know something about a person can be considered unethical. There may be situations where it is morally more appropriate not to know a person's gender, income, sexuality, political opinion etc. In fact, such aspects are often considered private and are debated in discussions about privacy protection. A good reason for keeping such information private is the fact that knowing certain facts about a

person can have disadvantageous consequences for the person. Knowing a person's sexuality can have severe consequences for how that person is treated in certain situations or in parts of the world. Perhaps even simpler, modelling people's gender as only binary can lead to undesired classifications, misaligned expectations about the person, or unsuitable functionality, for example for transgender people. Generally, categorising people may make people feel unduly classified or reduced to 'just a mere category'.

AI models are often used for deciding on people, e.g., in a bank loan application, a job search, or an insurance application. Consequently, AI decisions based on models of people and on categorisation can lead to denied loans, not getting a job, on an increased insurance premium. Finally, controlling people's behaviour can easily be abused and become immoral, for example when exposing people to unwanted advertising or limiting people's autonomy.

2 Bias and Choice

2.1 Fairness

A. Assume that a data-driven creditworthiness rating system has been trained on past data. In the past, significantly fewer women asked for a loan, also they proved less likely to pay it back.

a. Which ethical issues may arise when using the system?

b. Assume that you would like to correct for potential bias with the aim to "treat men and women equally". Would it be better to look for a "nearest neighbour" in the data so that we treat a woman like a "similar" man, or to ensure similar approval rates between men and women?

c. *Will the aim of ensuring similar approval rates lead to the same results as ensuring similar disapproval rates for men and women?*

Let us start with the observation that (machine-learning based) AI models are usually trained on historic data. As the future (or in fact the present) may be different from the past, there will always be the issue of model validity, i.e., is the AI model still a good model for the situation or have the underlying characteristics of whatever is modelled changed? In fact, we can never be absolutely sure that an AI model is fully correct, especially when modelling complex systems or humans. We can only try to evaluate model correctness to ensure that the model is still applicable.

But even when the model is correct in terms of the historical data and it is, in principle, still a valid model of the current situation, it may deliver results that are societally or ethically inappropriate. The following figure describes a common situation where the AI model correctly grasps past data, but applying the model directly leads to undesirable consequences. Just because in the past most engineers were men and this may still be the case, it does not follow that we should only select men or an open position in engineering. In fact, it may be a good idea to specifically select more women as

engineers if we would like to realize a balanced representation of women in engineering in the future.

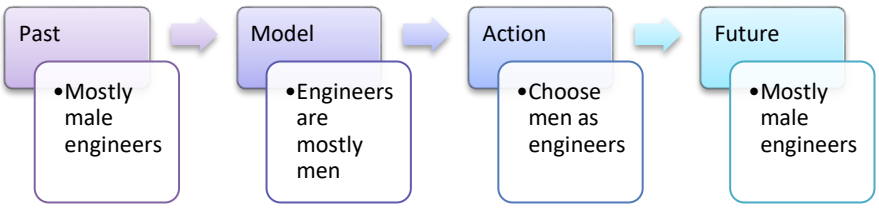


Figure 1 The design of AI models often relies on historic data. Decisions based on such models will tend to continue the past into the future. In this example, a system that supports hiring decisions for engineers based on historic data may choose men over women given the historic fact that more women than men were engineers in the past.

Note that such a situation is usually called a *bias*, because the model is skewed towards men as engineers. This does not mean that it is wrong, it just means that it is ethically, socially, or politically questionable. On a positive note, AI models may help to make certain societal issues explicit. For example, it is not uncommon that certain types of bias in data are detected or at least debated following the implementation of an AI model. Decisions to 'correct' such biases are often political or societal agreements to overcome existing and unfair tendencies with the help of explicit policies, e.g., affirmative action. This is especially important as AI models will be used for the future and without such correction may lead to the perpetuation of trends that are considered unethical or societally unwanted.

2.2 Model Cards

B. Discuss the advantages of model cards (i.e. information about how a model was trained, training data, and potential limitations). Where are the disadvantages and limits?

How do you think such information would help the following users of an AI system:

- *A medical doctor using an X-ray classification system?*
- *A human resource manager classifying job applicants?*
- *A consumer using a music recommendation system?*

A suggestion that has been made to increase the transparency of AI systems is to provide so-called *model cards* that contain details about the data used for training or characteristics of the AI model including, for example, potential bias, benchmarks across different cultural or demographic groups. Model cards can also provide information on application contexts and procedures used for the evaluation of the AI model. The OECD

includes model cards in its catalogue of tools and metrics for trustworthy AI.[1] Model cards can be useful for users, policy makers, or AI and machine learning practitioners.

While this may provide relevant information in important situations, it also means that there can be a shift of responsibility to the user of the system. The information provided for an AI model may also be quite difficult to interpret for laymen. While an expert, e.g., medical doctor, may find it useful and is perhaps capable to understand potential biases or overall quality assessments regarding model correctness, such information may be of little help to consumers, for example.

2.3 Trolley Problem

C. *The Trolley problem is often formulated as follows:*

A tram is running out of control and threatens to kill a group of people located on the track. There is a person near a lever who could flip the switch and divert the train on a different track so that only one person would be killed. Should the person near the lever pull the lever to divert the runaway trolley onto the side track to save the five people from being killed at the price of killing just one?

Now consider a hospital emergency unit. A young, otherwise healthy man arrives after an accident with broken legs. At the same time, there are five transplant patients in the hospital of which two need new kidneys, one needs a lung, one a heart, and one a liver - all of which would match the young man. Do you think the young man should be killed to provide the organs?

- *Discuss the similarities and the differences between the two cases!*
- *Discuss the similarities and differences of the Trolley problem in modelling the situation of autonomous vehicles!*

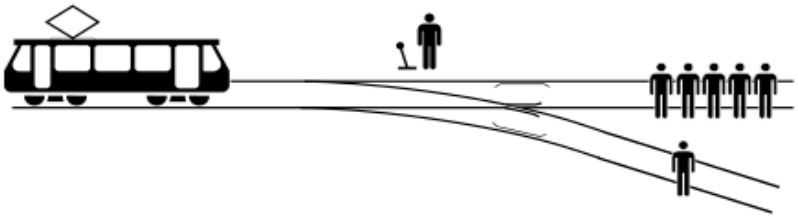


Figure 2 A person is depicted at a switch that can divert an out-of-control tram to killing only one person instead of five people if no action is taken. (Original figure: McGeddon Vector: Zapyon, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons)

[1] https://oecd.ai/en/catalogue/tools/model-cards

The Trolley problem has become very popular in discussing AI-based decision making. It is often referred to in discussions about autonomous driving or in medical decision-making. The problem is also used in ethics to explain different ethical schools of thinking, e.g., consequentialist versus deontological ethics, or the moral obligation to act versus inaction.

One possible approach to answering the Trolley dilemma is to take a utilitarian perspective and ask, what would be the greatest benefit for the largest number of people? In such a view, the focus is entirely on the consequences of the action. Neither does it consider certain rules that may govern operating switches or general rules about killing people, nor does it consider any character traits of the person at the switch. The focus is solely on the result. From this point of view, the only thing that counts is to limit the damage done as much as possible and to accept a single person dying rather than five. In addition, we may want to regard the flipping of the switching as a duty to act rather than not to act.

An entirely different perspective is what is called "deontological" where we assume that killing is intrinsically bad. Hence, the person at the switch has no duty to change the track as this would lead to killing a person. In this view, we must not count up lives against other lives, i.e., killing one is as bad as killing five. This view of ethics relates to a line of thinking that is usually associated with Immanuel Kant and his idea that we should only follow rules that we could consider as general laws of action.

While many people may consider the consequentialist position as straightforward, this changes as soon as we start adding information and changing the situation only slightly. A common variation is the introduction of a "fat man" instead of the switch. We now assume that there is big man on a bridge that could be pushed onto the track and that this would also keep the five people from being killed. Interestingly, now some people would no longer regard pushing the man on the track to stop the tram and save the five people as either morally good nor a duty of a person. The fact that in this situation we would have to directly interact with the person seem to change the moral characteristic, at least for many people.

Another even more drastic change is the hospital situation mentioned above. Only very few strict consequentialists would suggest killing a young man with a broken leg and to use him as an organ donor to save five others. We may assume that this is not just morally inacceptable for many, but it is also undermining the idea of a hospital as a safe place to go to in case of injuries. These few examples already demonstrate how much our human moral decision-making depends on subtleties of the situation. Changing just small aspects such as the age of the people, family relations among them or between the person at the switch and the potential victims will add more complications and changes in moral decision-making.

Finally, there is the question of whether the Trolley problem can be used of a model for autonomous driving [2]. The obvious analogy would be a self-driving car that faces a situation where it could choose to kill a group of pedestrians or the driver of the car. It is important to point out that this is hardly a realistic scenario. Firstly, such situations will be extremely rare. Secondly it is unlikely that a car's predictions would be sufficiently reliable to predict deaths and offer any choice in real-world situations. Thirdly,

it may be entirely impossible to sell a car that does not protect the driver in all such situations. All of this means that there are serious doubts about the practical applicability of the Trolley problem to real-world situations. A possible answer to the question posed in the Trolley problem is therefore to reject the question. Not only is it unrealistic, but it also suggests that the problem has a technical solution and it therefore implies that we should aim at a technical solution.[2]

2.4 Freedom of Speech

D. Discuss the following image from the point of view of an AI-based image classification system.
   - What is this image about? Provide a range of perspectives!
   - Do you think this image is pornographic? If so, what are the reasons for such a judgement? If you think it is artistic, why?

Figure 3 Michelangelo Marisi da Caravaggio: Amor vincit omnia, ca. 1602. (Public domain, via Wikimedia Commons,)

[2] Another way to answer the question has been proposed by Tim Henning. We should (ceteris paribus) save a larger number of people if / because this is what those concerned would choose given a fair voting procedure. x

There have been intensive debates about wrongful, illegal, or inappropriate content in social networks and its damaging impact, e.g., on democratic values or the results of elections. Online comments can also have negative impacts on individuals when they are insulting, wrongful or contribute to bullying. As a technical approach to solving this problem, AI has been suggested for the identification and classification of problematic online content [5]. It is widely used in online networks for the classification of texts, videos, and even music with the aim to delete content that is considered wrongful, illegal, or inappropriate.

There is a broad debate about inappropriate content and its removal with many different and interrelated aspects making this a complex problem. The first point to consider is whether private platforms should be considered public spaces. This is relevant for discussing to what extent such platforms should be held responsible for the type of content that is published. It has been suggested, for example, to consider differences between small platforms, e.g., from a local news web site, that do not reach millions of people in the way that some of the very large platforms such as Facebook YouTube, or LinkedIn do.

Another question relates to what type of content should be concerned. In many countries, there are laws that regulate certain types of content making its publication clearly illegal, e.g., advertising terrorist acts, denying the holocaust, public insult, infringement of intellectual property etc. However, there have also been calls for limiting so-called "harmful" content. This is a much more problematic content as it is often not clear what should be considered harmful and who should have the power to define it. What some may consider strong but necessary language to identify political wrongdoing, other may consider inappropriate and harmful for society. Obviously, these are not only moral but also political questions.

AI can be used to identify unwanted key words and certain types of content. But it is important to note that the technology used for the identification of illegal or inappropriate content is far from perfect. It will wrongly identify wrongful content as appropriate and miss the identification of illegal content. Consequently, this raises significant questions about complaint and re-instantiation procedures as well as questions about potentially negative consequences for people who are falsely accused of having published bad content. While many states are introducing regulation for the deletion of content, the question of complaints and rights to publish in large networks has received less attention.

Deletion of content always raises concerns regarding censorship, both in democratic and undemocratic states. Using AI for content moderation therefore poses important questions about the power of online platforms to explicitly or tacitly delete, annotate, or rank content. This includes questions of publishing information about deleted content, responsibilities to monitor content, check the factuality of content, facilitate complaints etc. An important question from a democratic perspective regards the collaboration of large online platforms with dictatorships and parties with specific political interests.

In democratic states, the right to publish online relates to Freedom of Speech. This is often considered a core and important basic right that must be fundamentally protected. For example, the European Court of Human Rights has ruled that language used

in public opinion sometimes needs to be strong. This might be considered against the current trend where there are strong calls to also regulate "harmful" content in large networks.

Finally, these issues are very similar to more recent developments regarding generative AI such as ChatGPT. Again, the question is to what extent should we regulate the utterances of chatbots as concerns content that can be considered illegal. Potential issues range from infringing personal rights of people (e.g. insult) to plagiarism, misinformation, illegal content etc. The difference to social network content moderation is that in chatbots the texts are automatically generated for a user. This poses the question to what extent users should be held responsible as well.

3 Ethical Frameworks

Given the many ethical challenges of AI systems, it is unsurprising that there have been many proposals how to address them [3]. A large number of publications therefore addresses ethical principles and proposes frameworks for addressing them. These frameworks include aspects such as the following:

- **Transparency** (including explainability, understandability, disclosure etc.)
- **Justice** and fairness (including consistency, inclusion, equality, bias, diversity, remedy, redress etc.)
- **Non-maleficence** (security, safety, precaution, prevention, integrity etc.)
- **Responsibility** (accountability, liability)
- **Privacy**
- **Beneficence** (well-being, peace, social good, common good)
- **Freedom** & autonomy (consent, choice, self-determination, liberty, empowerment)
- **Trust**
- **Sustainability** (environment, energy)
- **Dignity**
- **Solidarity** (social security, cohesion)

More than 100 ethical frameworks for AI have been developed and many of them are very similar. While they are useful for structuring the issues, they do not provide clear recommendations on how to design systems. Indeed, several aspects of such frameworks provide significant research challenges. As an example, consider the case of "explainability": Explainable AI (or "XAI") has turned into a whole subfield of machine learning with its own conferences and a growing number of publications. This clearly indicates that the issue is complex and may require significant research on its own.

There are many proposed techniques for addressing ethical issues in the literature ranging from algorithms to labels, data bases, communities, standards etc. [4]. Many

of these are in fact useful, but their application depends on the application, the type of model, data, and training algorithms etc. There is no automatic or even straightforward way of implementing a "fair AI" system and it is questionable what this precisely means. Just consider aspects such as "justice" or "fairness". These are concepts that require not just algorithmic or mathematical considerations. They really are societal or political concepts that include decisions about what a society considers to be "fair". The scientific AI literature sometimes seems to be overlooking this when focusing on mostly algorithmic solutions to "fair AI". But questions such as what an acceptable bias of a system is and what is unacceptable or which actions should be taken to overcome bias in an AI system require decisions that go far beyond just mathematical considerations [6].

References

1. Schopenhauer, A.: Über die Grundlagen der Moral. (On the Basis of Morality, 1995) Berghahn Books, Providence (1840).
2. Nyholm, S., Smids, J. The Ethics of Accident-Algorithms for Self-Driving Cars: an Applied Trolley Problem?. Ethic Theory Moral Prac 19, 1275–1289 (2016). https://doi.org/10.1007/s10677-016-9745-2
3. Morley, J., Floridi, L., Kinsey, L., Elhalal, A.; From What to How: An Initial Review of Publicly Available AI Ethics Tools, Methods and Research to Translate Principles into Practices, http://dx.doi.org/10.2139/ssrn.3830348 (2019).
4. Prem, E.: From Ethical AI Frameworks to Tools: A review of approaches. In: AI and Ethics, Vol. 3, pp. 699-716, (2023).
5. Erich Prem, Brigitte Krenn (2024) On algorithmic content moderation. In Werthner, H. et al. (eds.) Introduction to Digital Humanism, Springer, Heidelberg (2024).
6. Seng Ah Lee, M., Floridi, L., Singh, J.: Formalising trade-offs beyond algorithmic fairness: lessons from ethical philosophy and welfare economics, https://ssrn.com/abstract=3679975 (2021).
7. Gert, B.: Morality. Its Nature and Justification. Oxford University Press, Oxford https://doi.org/10.1093/0195176898.001.0001 (2005).

Natural Language Processing. An Overview*

Brigitte Krenn and Johann Petrak

Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, Austria
{brigitte.krenn,johann.petrak}@ofai.at
https://www.ofai.at

1 Natural Language Processing (NLP)

This contribution provides an overview of computational methods for "understanding" or generating natural language. By putting understanding under quotes, we want to stress that what is called understanding in Natural Language Processing (NLP) is qualitatively very different from what a human commonly would interpret as understanding. The overall goal of NLP is to process/analyse and to generate human language with computers. The goal in natural language analysis is to build useful representations from natural language input, and in natural language synthesis or generation, the goal is to produce natural language output from (structured) representations. The massive challenge we still face in NLP is to make the meaning (semantics, pragmatics) of human language accessible to computers.

1.1 Ambiguities

A major challenge in NLP is ambiguity, i.e., the same text can have very different meanings, and different texts can have the same meaning. As regards the latter, think of different wordings of summaries of the same key facts or bits of information of a document. Differences in language use may also arise from socio-linguistic differences, from colloquial and informal language. This is specifically the case in social media (and in emails), where code and language switching, i.e., mixing standard language and dialect, and also mixing languages is prevailing. In the following, we will give a number of examples for natural language ambiguities:

Lexical ambiguities The sentence *He went to the bank* may be interpreted in different ways, depending on the reading of the word *bank* which either may refer to a *savings bank* or a *river bank*. Moreover, a reference to *savings bank* may relate to the office building of the bank, but also to the bank as an institution, and we understand a sentence such as *She talked to the bank.* in a way such that she talked to a person with an official function in the bank.

* Supported by the Austrian Research Institute for Artificial Intelligence (OFAI).

Part-of-speech ambiguities There are also ambiguities due to the syntactic categories a word can have In NLP, syntactic categories of words are called parts-of-speech (pos). As an example, see the potential headline *Teacher Strikes Idle Kids* (cred. Dan Klein) where we have two words *strikes* and *idle* for which we find different pos when we look up a dictionary: *strikes* can be a noun or a verb and *idle* can be a verb or an adjective. Accordingly, *Teacher Strikes Idle Kids* may mean the due to strikes of teachers kids idle with *strikes* being interpreted as a noun and *idle* as a verb. When we interpret *strikes* as a verb and *idle* as an adjective, the meaning changes to 'a teacher hits who idle'. You may also have noticed that *strikes* is also lexically ambiguous: *strike* meaning work stoppage or hitting.

Structural ambiguities Depending on how you parse the sentence *If you love money problems show up*, you will end up with different readings: 'if you are a money lover, problems will occur' (in this case the noun *money* functions as object to the verb *love* and *problems* is the subject to the verb *show up*) or 'if you are a lover of money problems, you should come here' (in this case *money problems* is interpreted as a compound noun and is the object of the verb *love*, whereas the verb *show up* is interpreted as an imperative). Another important structural ambiguity is related to PP-attachment. Consider for instance the sentence *She wrote a report on Mars.* Is the report written while being on Mars or is it a report about Mars? Similarly, a headline such as *Cop kills man with knife* can mean that the cop has the knife and uses it to kill the man, or the cop kills the man who has the knife.

Anaphoric reference Consider the following two sentences *Mark saw the men with the telescope. He had brown hair.* In the first sentence, we have a structural ambiguity due to PP-attachment of the prepositional phrase (PP) *with the telescope*, i.e., who had the telescope – Marc or the man? For the second sentence we need to find the referent for the pronoun *he* in order to know who had brown hair – Marc or the man?

Multi-word expressions See the sentence *they wanted to do it but they got cold feet which made them lose their face* where *got cold feet* is a multi-word expression and means 'become frightened' as opposed to the literal meaning of getting cold feet as in *They got cold feet because their shoes were too thin.* Distinguishing whether something should be literally interpreted or in a figurative sense (as a multi-word expression) is hard for computers.

Pragmatics an utterance such as *It is cold.* can be both interpreted as a factual statement or as a request for action, e.g., to close the window, to turn on the heating, to pass on a plaid, etc. The intended meaning is often clear from the (situational) context or with background knowledge. For a system to deal with it, this extra knowledge needs to be somehow accessible or available.

1.2 Two Views on Natural Language

The Linguistics View The linguistics view sees natural language with a stratified view from speech signal to text along the following steps:

- Phonetic form is the visual representation of sounds, including including phonemes as units of sound (/p/, /t/, /k/, ...), and syllables as units of pronunciation (/wa-ter/, /un-for-giv-ing/). For a notation system, cf. the International Phonetic Alphabet (IPA).[1]
- The morphological structure: Morphemes as building blocks for representing the relation between words (word stems or types) and their inflected forms (tokens): e.g., build (stem/type) – build-ing, build-er, built, build, build-s (forms/tokens); forgiv (stem/type) – forgiv-e, un-forgiv-ing, forgave, forgiv-en, forgiv-ing (forms/tokems).
- The syntactic structure of sentences and phrases as their building blocks, e.g., *an unforgiving environment* (noun phrase, NP), *is unforgiving* (verb phrase, VP), *in the deep see* (prepositional phrase, PP), *The deep sea is an unforgiving environment.* (sentence, S).
- Text as a sequence of (related) sentences: see for instance the following two sentences *The deep sea is an unforgiving environment. This is why we should not stay in it.*. The first sentence introduces the topic or theme (deep sea) and the rheme (unforgiving environment),[2] The second sentence formulates as an entailment not to stay in the deep sea. Both sentences together express a causal relation between deep sea as unforgiving environment and the conclusion not to stay in the deep sea, with *it* in the second sentence being an anaphoric reference to *deep sea* in the first sentence. Explicitly modelling these kinds of relations on a computer, e.g., with rule-based approaches, is very hard. Machine learning models which take large contexts into account, however, stand a good chance to implicitly modelling such relations.

A Computational, Practical View From a more computational, practical perspective, text is viewed as a sequence of (Unicode) characters. Words or (sub-word) tokens are (usually) the relevant parts of the text. A document is a unit of text that somehow belongs together. An article, a book, a page in a book, a Tweet, etc. can have associated meta-data sch as author, publishing date, keywords and many more. A corpus is a collection of documents that belong together. With the advancement of machine learning (ML), linguistics has increasingly taken a backseat in natural language processing.

[1] https://www.cambridge.org/features/IPAchart/
[2] Note, theme and rheme are terms from information structure, [48].

2 Approaches to NLP and NLP Tasks

2.1 Approaches

There are two major approaches to NLP, the classical pipeline-based divide and conquer approach and the end-to-end approach. Furthermore NLP applications can be differentiated into applications for analysis and applications for synthesis.

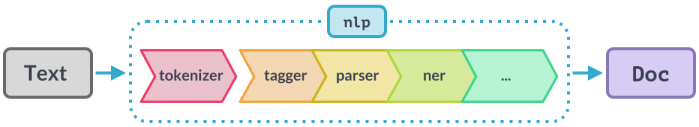


Fig. 1. NLP Pipeline: Figure taken from https://spacy.io/usage/linguistic-features, "Hooking a custom tokenizer into the pipeline".

Classical divide and conquer breaks up a complex task into smaller ones. Specialised methods are developed to solve the individual tasks, whereby there can be many different possible approaches (rule-based, machine learning based or hybrid) to solve the overall task. The methods for solving the individual sub-tasks are combined into a "pipeline" to solve the superordinate task.

End-to-end A single (deep-learning based approach) does the whole task from input to output all in one go. However, some NLP tasks such as tokenization and some machine learning tasks, including data-loaders and batching, are still needed.

Ananlysis and Synthesis Typical examples for text analysis applications are text classification (e.g. spam detection), and Information Extraction (IE) in order to extract structured information from text, to find mentions of names, organizations, relationships, events. Natural language synthesis or generation refers to content generation from structured information (e.g. weather forecast writing), and more recently to large language models (LLMs) which predict what comes next given a preceding text. Moreover, question answering (QA) and machine translation (MT) comprise both, methods for analysis and generation. In QA a natural language query is analysed and an answer in natural language is generated. In MT the textual input in source language is analysed and it pendant in a target language is generated.

2.2 Common NLP Tasks: Overview List

The following list is meant to give a brief overview of the most NLP tasks. In the next sections, we will look at some of those tasks in more detail.

- Tokenization: A machine-readable test is noting else than a stream of characters. Tokenization is required to identify individual words which then can be used for further processing.
- Sentence splitting: Even though, the text is already split into words, sentence splitting is required to identify individual sentences in the sequence of representing a tokenized text. Sentence splitting is a precondition for syntactic parsing, as the largest unit for a parse is a sentence.
- PoS tagging: Part-of-speech tagging identifies for each word its unambiguous syntactic category or part-of-speech in the context of the given sequence of words the word of interest occurs in. PoS tagging requires tokenization as a precondition.
- Parsing: groups the words within a sentence into syntactic units or phrases. PoS tagging supports parsing as the parser can already operate on a sequence of words with disambiguated syntactic categories. However, there are also approaches to parsing where PoS tagging is integrated in the parsing process. Knowing the syntactic structure of words helps to identify the meaning relations, e.g., what is the verb, what is the subject, what are the objects, which word modifies another one, and so forth.
- Lemmatization/stemming: is particularly interesting in highly inflected languages such as German where different full forms relate to the same stem, see for instance *Haus* (house), *Hauses* (house), *Häuser* (houses), *Häusern* (houses).
- Keyword/term extraction: Some words are more relevant for a document than others. Term extraction is the task to identify these words.
- Entity recognition and disambiguation/linking: Entity recognition refers to the identification of person, company, product, country, place names. Entity disambiguation or linking is the task to unambiguously map named entities identified in a text to real people, companies, etc.
- Reference resolution: is the more general task of identifying who or what is referenced by a linguistic expression such as a name (e.g., *George Bush* – which George Bush?), a pronoun (e.g., *she made her choice* – who is referred to by *she*), a definite NP or PP (e.g., *the second husband of the president* – who is the husband, who the president?).
- Topic detection: also called topic analysis, topic extraction aims at assigning topic or theme category labels to individual documents/text of large collections of text data based on the content of the individual text to be labelled.
- Relation extraction: aims at identifying relations between entities in a text, e.g., what is the person's name, their occupation, to whom is the person married, and so forth.
- Text classification: aims at assigning pre-defined categories to text.

2.3 NLP Task: Tokenization

Tokenization is the task to divide a sequence of characters into larger meaningful units (tokens) for further processing. When we want to split a text into words, we

need to ask ourselves: What is a word? See for instance strings such as *it's, don't, data-mining*. You also need to consider punctuation, e.g., does a dot belong to an abbreviation (*etc.*) or is it a punctuation mark or both (*They bought fruit, vegetables, drinks etc.*)?

The most simple, however, not sufficient way to do this is to split on white space or punctuation. White space does not work for each language, see for instance Chinese or Turkish or any other language which has sub-word units. Moreover, languages can have multi-word tokens, e.g. *it's, suntan lotion, New York Times, kick the bucket*, where more than one words are merged into a token (*it's* → *it is*, or one token comprises several words separated by blanks as it is the case in English compound nouns (*suntan lotion*, names (*New York Times*), or multi-word expressions (*kick the bucket* meaning 'die'). Languages can have punctuation tokens (punctuation marks) and other special tokens, including abbreviations (*e.g., UNESCO*, URLs, @names, email addresses, #multiwordhashtags. As a result of tokenization a text is represented as a sequence of tokens instead of a string of characters including white spaces.

2.4 NLP Task: Sentence Splitting

Sentence Splitting is important in NLP because sentences are a good unit of analysis: A sentence is a self-contained unit of meaning relations, it is syntactically complete, and typically contains subject, predicate, object(s), modifiers. See for instance the following sample sentences and their internal structure represented by labelled brackets:

Sentence: The house is green.
Syntactic structure: ((the house)_subject (is green)_predicate)_sentence

Sentence: The green house belongs to my cousin.
Syntactic structure: ((the green house)_subject ((belongs)_predicate (to my cousin)_object)_sentence

Splitting text into 1 or more sentences is sometimes good for for technical reasons, e.g., for parallelization or if there is limited input size. The most simplistic way to split a text into sentences in to split on on double new lines and on punctuation, e.g., full stop, exclamation mark, question mark, however, you need to take care of abbreviations with dot. Therefore there are also more sophisticated methods which make use of some of the sub-tasks and approaches we discuss later in the text.

After sentence splitting, the text may get represented as a sequence of sentences. Each sentence being a sequence of tokens: (((After) (sentence) (splitting) (,) (the) (text) (may) (get) (represented) (as) (a) (sequence) (of) (sentences) (.)) ((Each) (sentence) (being) (a) (sequence) (of) (tokens) (.))) In comparison the the structured examples above, there is no information about the syntactic structure of the individual sentences yet.

2.5 NLP Task: Stand-off Annotations

From the above examples you already see how clumsy it is to use nested lists. Therefore stand-off annotations have been introduced as an alternative comprising several sets or lists of annotations which are linked to the original text via start, end offsets in the original text document. Some variations use a token index instead of char offsets. Apart from the start-end offset annotation, there are type-specific annotations with features providing information about the given text, e.g. a specific token is a word of type noun, a specific sequence of tokens is a noun phrase or a person name. There may be arbitrary many and arbitrarily overlapping stand-off annotations grouped into sets for a given text. Stand-off annotations can be linked to form a tree or graph, and the ones useful for specific processing tasks can be deliberately selected. Details differ between implementations, for examples see SpacY, GateNLP, BRAT.[3] For an example of an annotated document with several types of stand-off annotations from the GateNLP framework see https://tinyurl.com/stoffann.

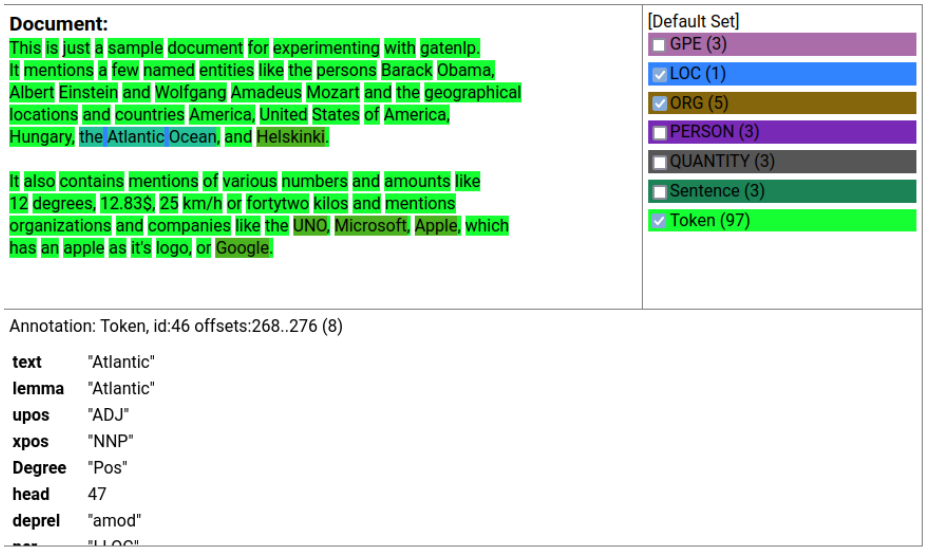


Fig. 2. Stand-off annotations GUI.

2.6 NLP Task: Part of Speech (PoS) Tagging

For further processing, We need more information about each word/token, such as what morpho-syntactic category (part-of-speech, PoS) it has, is it a verb or a

[3] https://spacy.io/usage/linguistic-features, https://gatenlp.github.io/python-gatenlp/annotations, https://gatenlp.github.io/python-gatenlp/annotationsets, https://brat.nlplab.org/standoff.html

noun, etc. Ideally, we also want to have additional morpho-syntactic information such as gender, time, singular/plural, case. Mere lookup in a dictionary does not work well because of ambiguities, e.g., *duck* can be a noun referring to the animal or a verb meaning crouch down. The aim of PoS Tagging is to disambiguate a word's syntactic category according to the given textual context.

Machine learning is used for developing PoS Taggers. To train a tagger, first of all a pre-annotated corpus is required where each word is annotated with those features relevant for training a machine learning (ML) algorithm. Second, a suitable ML algorithm for training the Tagger model needs to be selected, the model needs to be trained and tested.

```
1    Seriously   seriously  ADV
2    :           :    PUNCT
3    do   do   AUX
4    not  not  PART
5    waste    waste    VERB
6    your you    PRON
7    time time NOUN
8    .    .    PUNCT
```

Fig. 3. CoNLL-U format example.

PoS Tagging: Corpus and Tagset Examples Several Treebanks with different tag-sets and for different languages exist. Treebanks are collections of sentences annotated with PoS and syntactic structure, see for instance the Penn Treebank for English ([35], [52]) or the TIGER Treebank for German [8] and its conversion into universal dependencies annotation.[4] A universal dependencies (UD) annotation is useful because of the harmonised PoS tagset applicable to many languages.[5] Corpora in UD format exist for many languages, with the CoNLL-U annotation format being the most widespread. [6] In Figures 5 and 4, you see examples for the Universal and and the Penn PoS Tagsets. Apart from the core part-of-speech categories shown in Figure 5 the UD tagset comprises additional universal features to distinguish further lexical and grammatical properties of words.[7]

POS Tagging: Features A number of features is used in PoS Tagging. This ranges from features representing information related to the current token to

[4] https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger/, https://github.com/UniversalDependencies/UD_German-GSD
[5] https://universaldependencies.org/
[6] https://universaldependencies.org/format.html
[7] https://universaldependencies.org/u/pos/

1. CC	Coordinating conjunction	25. TO	*to*
2. CD	Cardinal number	26. UH	Interjection
3. DT	Determiner	27. VB	Verb, base form
4. EX	Existential *there*	28. VBD	Verb, past tense
5. FW	Foreign word	29. VBG	Verb, gerund/present participle
6. IN	Preposition/subordinating conjunction	30. VBN	Verb, past participle
7. JJ	Adjective	31. VBP	Verb, non-3rd ps. sing. present
8. JJR	Adjective, comparative	32. VBZ	Verb, 3rd ps. sing. present
9. JJS	Adjective, superlative	33. WDT	*wh*-determiner
10. LS	List item marker	34. WP	*wh*-pronoun
11. MD	Modal	35. WP$	Possessive *wh*-pronoun
12. NN	Noun, singular or mass	36. WRB	*wh*-adverb
13. NNS	Noun, plural	37. #	Pound sign
14. NNP	Proper noun, singular	38. $	Dollar sign
15. NNPS	Proper noun, plural	39. .	Sentence-final punctuation
16. PDT	Predeterminer	40. ,	Comma
17. POS	Possessive ending	41. :	Colon, semi-colon
18. PRP	Personal pronoun	42. (	Left bracket character
19. PP$	Possessive pronoun	43.)	Right bracket character
20. RB	Adverb	44. "	Straight double quote
21. RBR	Adverb, comparative	45. ‘	Left open single quote
22. RBS	Adverb, superlative	46. “	Left open double quote
23. RP	Particle	47. ’	Right close single quote
24. SYM	Symbol (mathematical or scientific)	48. ”	Right close double quote

Fig. 4. Penn Treebank POS tagset (from [35]).

Open class words	Closed class words	Other
ADJ	ADP	PUNCT
ADV	AUX	SYM
INTJ	CCONJ	X
NOUN	DET	
PROPN	NUM	
VERB	PART	
	PRON	
	SCONJ	

Fig. 5. Universal POS tags (https://universaldependencies.org/u/pos/).

information about the left and right context of the current word (token). Information at token level comprises whether the token is the start or end of a sentence, whether it is a punctuation mark, whether the token contains a number, starts with an upper-case character, is all upper case, contains a hyphen, and so forth. In addition, features from previous and succeeding n tokens (context window) are taken into account. Moreover, [55] showed a positive effect on tagging accuracy when adding predictions for PoS-tags of previous word(s) as additional features.

With these features we have a broad range of information about a token and its context, however, the word itself, i.e., its lexical representation is not yet in our feature set. Languages comprise tens of thousands of lemmata (stems or base forms) and many more full forms. How can this information be added as a feature in an efficient way? E.g.: Give each word a number? This would require tens of thousands of numbers, in other words we would define a feature with a huge value set. An alternative would be one-hot features, which, however, would require tens of thousands of these features. So there is actually no gain, yet. An alternative idea is to group words into k clusters and then use one-hot encodings for the clusters, leading to k one-hot features instead one one-hot feature per word. An example for a clustering algorithm is Brown Clustering which we will introduce in the following.

Brown Clustering is a hierarchical, agglomerative clustering algorithm [9]:

$$\sum_{c_1=1}^{k} \sum_{c_2=1}^{k} p(c_1 c_2) \log \frac{p(c_1 c_2}{p(c_1)p(c_2)}$$

The idea behind is to maximise the mutual information of cluster bigrams. As a consequence, Words that appear in the same context tend to get clustered together such as *'Friday, Monday, Thursday, weekends, Sundays'*, *'people, guys, folks, chaps, doubters, blokes'*, *'down, backwards, ashore, sideways'*, etc. This idea works, because the contexts of words give important information about the word itself! Each cluster represents a "word type/class" or "context class", and the cluster number can be Used as a feature, represented as number or one-hot vector.

POS Tagging: Machine Learning Use the best ML algorithm of the time, e.g., logistic regression, decisions trees, random forests; support vector machines and other large margin algorithms, i.e., algorithms that are able to identify the hyperplane that represents the largest gap with the fewest exception (also called margin) between two classes. Different algorithms exist, handling numerical versus categorical features, handling very many and sparse features, with different model complexity, etc. Models are created that map from the features of the current, preceding and following tokens to a PoS tag for the current token, whereby each token gets classified separately.

Sequence labelling or sequence tagging algorithms are used to include dependencies how likely each PoS tag follows another one, in order to cover properties

of natural language such as a determiner (DET), pronoun (PRON) or adjective (ADJ) most often precedes a noun (NOUN), e.g., *the dog* (DET NOUN), *my dog* (PRON NOUN),*black dog* (ADJ NOUN). Which is a reflection of the syntactic structure of the (here the English) language. Lafferty et al. 2001 [32], for instance, use (linear-chain) Conditional Random Fields to cover these surface relations for PoS tagging.

Find $\text{argmax}_y\, p(\mathbf{y}|\mathbf{x})$ (using Viterbi algorithm). Whole sequence to be PoS tagged gets labeled in one go!

2.7 NLP Task: Word representations

Finding good word representations which make it easier to deal with the meaning of words (e.g. make it easier to group words with similar meanings is crucial in NLP. As we already discussed, Brown clusters are helpful as those clusters provide a grouping that is related to the meaning of words, but can we represent words in a better way where even more aspects of meaning can be captured?

Bag of Words (BoW) Inspiration comes from Information Retrieval (IR): Represent a document by a vector with n elements, n = number of different words (word forms) in the corpus the document is part of. For each word in the document, the document vector contains at the ith element of the vector representing all words of the corpus some numerical information about the word, for instance, an indicator (0, 1) whether the word occurs in the document, or the term frequency (= number of occurrences of the word in the document). For illustration see Table 1 depicting the vectorization of a toy corpus comprising 3 documents D1 to D3. Note: The BoW vector of a single word is a one-hot vector, e.g, the vector for mouse in our sample corpus is 0 1 0 0 0 0.

Documents	Words in corpus:	the	mouse	jumped	table	on	under
D1: the mouse jumped	D1 vector:	1	1	1	0	0	0
D2: the mouse jumped on the table	D2 vector:	1	1	1	1	1	0
D3: the mouse jumped under the table	D3 vector:	1	1	1	1	0	1

Table 1. Vectorization of a toy corpus comprising 3 documents D1 to D3; words occurring in the document are represented with 1.

However, we still have problems with this kind of representation: 1. in a real example we have tens of thousands of elements per vector; 2. there is no information about how the words occur in sequence, which means we are missing out crucial information, see for instance the two texts *bad, not good at all* and **good, not bad at all** which have the same BoW representation but an opposite meaning. A remedy for problem 2 is to use bag of n-grams instead of BoW: use one feature per e.g. bigram *good not*, *not bad*, etc. which, however, increases our problem 1. So we need to do better!

Again we use inspiration from IR: Whereas we initially represented each document of a corpus by a BoW vector, the corpus can as well be represented by a matrix with words in one dimension and documents in the other dimension where the value shows how often word i occurs in document j. Each document is represented by a vector of words (BoW), each word is represented by a vector of documents. Documents can be compared by finding the similarity between their word vectors, and words can be compared by finding the similarity of their document vectors. To do this usually cosine similarity is used: $\frac{v_1 v_2^\top}{||v_1|| \, ||v_2||}$

Unfortunately, these term/document matrices are huge, mostly sparse and very redundant. Therefore, rank reduction is called for! Singular Value Decomposition (SVD) is a widely used means in this context: Use SVD and choose the top k singular values. Latent Semantic Indexing (LSI) then employs the SVD to reduce the dimensionality of the original vector and, thus, represents documents as k-dimensional vectors of "concepts", i.e., words represented by k-dimensional vectors. This leads to vectors of dimensionality in the hundreds instead of hundred thousands to represent words. These vectors are called "dense" word vectors or "word embeddings". In the following, we look more closely into word embeddings.

Word Embeddings A problem is that the dense vectors described above produce word embedding from word occurrences over a whole (possibly large document), while the local context is particularly important! So the question is, can we create word embeddings based on the co-occurrence of words in their local contexts within a corpus?

Mikolov et al. 2013 introduced with Word2Vec an engineering approach to the problem. They used a neural network to create embeddings (Mikolov2013). They introduce Continuous Bag of Words (CBOW) predicting the center word of a context window from its neighboring words, and Skip-Gram: predict the neighboring words from the center word. For both approaches, we can choose the dimensionality we want, e.g. 200. A simple 1 hidden layer network is used, with symmetric windows (e.g. 5 word to the right and left). In addition, negative samples are generated with random selected words which do not occur in the vicinity of a certain word and which therefore should not be predicted from that word. Depending on the size of the training set, 2 to 20 words are randomly sampled which do not occur in the context of the word in question. The learned network parameters are then used as embeddings.

There is also a mathematical approach: generate a matrix and do dimensionality reduction. Many different solutions have been proposed, with Glove [44] being the most widely used one. Glove works as follows:

Define for a word a context window size and a symmetric (left+right) or asymmetric (left only) context. This gives us a $n * n$ matrix X where n is the size of the vocabulary. The Values in the matrix are the co-occurrence counts of the word and its contexts. What we want embeddings w such that $F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$ where the P_{ik} are co-occurrence probabilities. Do a simplification and conversion to least squares problem:

$$\sum_{i,j=1}^{n} f(X_{ij})(w_i^\top \tilde{w}_j + b_i + \tilde{b}_j - logX_{ij})^2$$

Use AdaGrad and stochastic sampling of X_{ij} to train. Use $W + \tilde{W}$ as final embeddings.

Many other approaches exist, e.g. FastText [4] which makes use of internal structure, i.e., use subword information (character n-grams). This is very helpful for highly inflecting languages such as German where different inflected forms as well as compound nouns share subwords. Moreover subwords are useful to infer the meaning of unseen words .

Word embeddings are useful for solving down-stream tasks (e.g. POS tagging). Similarities in embeddings (closeness in the vector space) follow to some extent human intuition.

Analogies can be found in embeddings, for instance, the vector for "king" minus the vector for "man" plus the vector for "woman" is closest to the vector for "queen" [37]. However, embeddings also reflect bias in the data: v("woman") + v("doctor") - v("man") = closest to v("nurse") [5].

2.8 NLP Task: Named Entity Recognition (NER)

The goal of Named Entity Recognition is to find and classify names in text: person, location, organization, genre name, species name, or other "chunks" of text like movie names, book titles, etc.

A supporting task is chunking, i.e., splitting sentences in phrases. This is not only important for IE but also for subsequently finding relationships between entities. An entity can consist of 1 to k tokens, see *Vienna* versus *People's Republic of China*. The need to find named entities anywhere in a text calls for converting the task to a manageable classification task, see [40, 34] for surveys.

BIO Coding BIO coding is a means to annotate multi-word strings: For each token, assign a label that shows if the token is outside (O) a NE, is the beginning (B), inside (I), the end word of an NE (E), a single word NE (S). Many similar formats, many names: BIO, BIOES=BILOU=BMEWO, etc. See [29] for a list of BIO codings. See an example for IOB2 coding in Table 2. One can use per token classification of the code similar to PoS tagging. Chunk boundaries can be derived from assigned codes. There is also a need to correct invalid sequences, as the ML classifier is not aware of what label sequences are actually possible and could e.g. assign an end label without a beginning label or two end labels in a row. So the output of the classifier has to be cleaned. As with PoS tagging, CRF can be used to consider the dependencies between labels.

Other Chunking tasks are to find short token sequences of some interesting type, e.g., noun phrases (NP), verb phrases (VP), prepositional phrases (PP). Chunking is also useful for more complex tasks, for example in biological event extraction, e.g. *apoptosis induced by the p53 tumor suppressor*. Actually in all

Alex	is	going	to	Los	Angeles	in	California
B-PER	O	O	O	B-LOC	I-LOC	O	B-LOC

Table 2. BIO coding: example for IOB2 format, with B-PER referring to begin person name, B-LOC begin location name, I-LOC inside location name, and O outside a named entity.

cases where it is important to know Which is the head word, i.e., the most important word, and which other words are attached to it. This is a task for (dependency-)parsing.

2.9 NLP Task: Parsing

Natural language has has structure (syntax) which follows construction principles (grammar) regulating how tokens (words from the lexicon) can be combined. The task in parsing is to analyse a sequence of tokens according to the rules of a formal grammar. Note: natural languages, formal languages, data structures follow different formal grammars.

In parsing a tree structure is used (i) to derive structure and meaningful relations (who is doing what to whom ?), (ii) to disambiguate word categories (parts-of-speech, e.g., noun or verb), (iii) to disambiguate the nesting of constituents (e.g., PP-attachment).

Natural language parsing uses linguistic knowledge for the structural analysis of a sentence, i.e., to parse a sentence into a tree revealing its phrasal structure. Phrase structure is the hierarchical structure of phrasal constituents and words. It express the linear order of a sentence. Dependency structure, in contrast, links word pairs by grammatical relations.

Dependency Parsing The aim of dependency parsing is to identify related words and the type of their relation: head word – relation type – dependent word (modifies head). Dependency treebanks are used to train respective parsers. Widely used are Universal Dependency (UD) Treebanks where text is annotated with Universal Dependency Relations[8] and PoS tags from the UD Tagset. Tools and models for parsing are for instance spaCy, NLTK, Stanza, CoreNLP, SyntaxNet ([1], Parsey Mc Parseface).

CoNLL-U[9] is a widely used annotation format for dependency relations: Each word is represented by 10 fields: ID, FORM, LEMMA, UPOS (universal PoS), lang.specific PoS (XPOS), morphological features (FEATS), the HEAD word, the dependency relation (DEPREL), head-dependency pairs (DEPS), any other annotation (MISC). See the annotation example below.

```
# sent_id = 1
```

[8] https://universaldependencies.org/u/dep/, e.g., the English UD Treebanks https://universaldependencies.org/treebanks/en-comparison.html

[9] https://universaldependencies.org/format.html

```
# text = They buy and sell books.
1 They  they    PRON    PRP     Case=Nom|Number=Plur            2 nsubj 2:nsubj|4:nsubj _
2 buy   buy     VERB    VBP     Number=Plur|Person=3|Tense=Pres 0 root  0:root          _
3 and   and     CONJ    CC      _                               4 cc    4:cc            _
4 sell  sell    VERB    VBP     Number=Plur|Person=3|Tense=Pres 2 conj  0:root|2:conj   _
5 books book    NOUN    NNS     Number=Plur                     2 obj   2:obj|4:obj     SpaceAfter=No
6 .     .       PUNCT   .       _                               2 punct 2:punct         _
```

2.10 NLP Task: Keyword/Key Phrase Extraction

Keyword or key phrase extraction has two Subtasks: candidate identification and candidate ranking. While the former identifies a keyword or key phrase in a document, the latter ranks the keywords/key phrases based on some statistics or probabilities. Different approaches exist: TextRank [36] and Topicrank [7] are graph-based approaches. Another example is YAKE! [12], a heuristic-/statistics-based model which computes a combined heuristic score based on various statistics on each word, and combines high-score adjacent words to phrases.

2.11 NLP Task: Named Entity Disambiguation (NERD)

Finding NEs is not enough! We also want to know what the NE refers to. For instance, does the NE *Vienna* refer to Vienna in Austria, in Ontario, or in Alabama, or is it the river Vienna running though Vienna in Austria, and so forth. A an obvious possibility to distinguish NEs is by linking to some knowledge base, e.g. Wikimedia, Wikidata.[10] However, some NEs may refer to an entity not in the KB, or several NEs may refer to different entities not in the KB. As a remedy new unlinked concepts are clustered.

There are many different approaches to NERD, but with frequent commonalities: Usually Entity Recognition (ER) is followed by Entity Disambiguation (ED). There are also methods for doing ER and ED jointly [28]. To desambiguate NEs you need to find semantic similarity between information about the entity (e.g. a respective Wikipedia article text) and the context of the mention of the NE in question. Moreover such features as a-priory likelihood or popularity of each link are taken into account. Multiple entities in the proximity of an NE should be compatible, i.e., belong to the same similarity space.

2.12 Evaluation

Evaluation is crucial in order to find out how good an NLP model is. Different NLP tasks require different kinds of evaluation. In the following, we will look into evaluation in NER, and discuss general issues in ML and NLP evaluation.

NER Evaluation NER evaluation basically comes in two variants: exact-match evaluation and relaxed or lenient-match evaluation.

[10] https://www.wikimedia.org/, https://www.wikidata.org/wiki/Wikidata:Main_Page

Exact-match evaluation In exact-match evaluation, both NE boundary and NE type must be correct to be correct to be counted. The following measures are used:

Precision P covers how many of the found NEs are correct:
$P = TP/(TP + FP)$
Recall R covers how many of the actual NEs have been found:
$R = TP/(TP + FN)$
F1-score F1 is the harmonic mean of precision and recall:
$F1 = 2(PR)/(P + R)$
where TP:true positives, FP: false positives, FN: false negatives.

P, R and F1 are calculated by micro and macro averaging. Micro-averaged means if there are several NE classes, average over all using TP, FN, FP. In macro averaging TP, FN, FP are calculated per class, and the per class results are averaged.

Relaxed/lenient-match evaluation NE type and (token-/character-) boundary are Considered separately. Different evaluation approaches exist, for instance: In MUC-6 evaluation [23] the type of the retrieved NE is considered as correct if the predicted type overlaps with the type of the target NE (i.e., the NE type in the test data). Independent of the NE type, the boundary of the retrieved NE is considered as correct, if it matches the boundaries of the target NE. Doddington et al. 2004 [18] propose ACE, complex scheme that is more flexible and powerful, but less intuitive and more difficult to use for error analysis.

Area under curve (AUC) [6] is another method to evaluate true positives TP versus false positives FP (AUC-ROC) or preision P versus recall R (AUC-PR) for different probability thresholds and calculate the area under the receiver operating characteristic (ROC) curve. Depending on threshold more correct examples may appear but also more wrong ones.

ML/NLP Evaluation What we really want to know in evaluation is the following: If we train a method to solve a certain task (e.g. NER) on a specific training set, how well will it do on new data, i.e., what it the generalization performance of a trained model. A single evaluation on some train/test split does not tell very much: maybe we were lucky, maybe we used the right random seed. To minimise this risk, we repeat our training and testing on several different train/test splits and use e.g. (class-stratified) k-fold cross-validation instead to get an average over the train/test runs and the related standard deviation. Note: To properly compare different approaches or settings the standard deviation needs to be checked, and significance tests should be performed in order to make reliable statements about whether different approaches differ in their performance. Unfortunately, many papers and leaderboards still neglect this and just show bare numbers from a single train/test split. In addition to the quantitative evaluation, a (qualitative error analysis) should be made, assessing which

errors have been made, and how "severe" or "surprising" these errors for the given NLP task are.

2.13 NLP Task: Text Classification

Text classification is an NLP task where predefined classes are assigned to open ended free text, e.g.: sentiment or polarity classification where text is assigned positive, negative or neutral sentiment. A typical application area for polarity classification are movie reviews, see for instance the IMDB Movie Review Dataset with 50K texts and related polarity information.[11].

The following is needed for text classificaton: a sufficiently large training corpus (e.g. IMDB Movie Review Dataset) annotated with the categories that shall be assigned to new text, an ML algorithm and features to represent the whole text corpus for the ML algorithm.

In order to represent a whole document as features for the ML algorithm the following approaches are possible:

- use a single document vector: BoW vectors, LSI (Latent Semantic Indexing) document vectors
- Use a single vector which is the average of all word vectors of the document → DAN
- Use list of word vectors: this requires an ML algorithm which can deal with variable length imputs: CNN, RNN, LSTM

3 Machine Learning Algorithms

In this section, we introduce a number of ML algorithms that were/are relevant for NLP. In doing so, we try to give an overview of how the field developed over time.

3.1 Deep Averaging Network (DAN)

DAN are Simple deep networks that worked surprisingly well at the time [25]. DAN take the averaged sum of embeddings with optional word dropout (randomly remove a certain number of words from the input) as their input. The model has several (usually 2 or 3) hidden layers with dropout and a non-linearity, like a Rectified Linear Unit (RELU) [22], or an Exponentional Linear Unit (ELU) [15], and uses linear classification with the *softmax* function for the final layer. The softmax function takes the values z_i of the k-dimensional input vector and normalizes it into a probability distribution of k probabilities so that those probabilities are proportional to the exponentionals of the input values:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

[11] https://paperswithcode.com/dataset/imdb-movie-reviews

DANs have an obvious disadvantage: they are negligent of word order, as a consequence syntax does not influence the input, e.g. negation cannot be handled!

3.2 Convolutional Neural Network (CNN)

Kim 2014 [26] presents experiments with CNN for sentence classification built on top of word2vec embeddings, which advanced the state-of-the-art amongst others in sentiment and in question classification. The input to CNN are n embeddings of dimension k (padded if text is shorter than n), whereby n depends on the maximum size of the training or expected test set. The CNN model is simple, comprising a convolutional layer with multiple filters, followed by max-pooling, and a fully connected layer with dropout and softmax output.

3.3 Recurrent Neural Network (RNN)

RNN apply a (multi-layer) neural network (NN) to each input of a sequence, e.g. from left to right. As RNN have loops, Figure 6 shows an unrolled RNN for better illustration. In addition to the input, at each time step, we get the hidden layer activations from the previous step and pass them on to the next step. back-propagation "through time" is applied. Advantages of RNN are that they have no fixed sequence length, and they can be used for both classification (use last output activations) and sequence labeling (use output activations for each input). Disadvantages are, they have local optima and exploding/vanishing gradients. Their inability to learn long distance dependencies, their effectiveness for NLP is restricted, a drawback which was overcome by LSTMs.

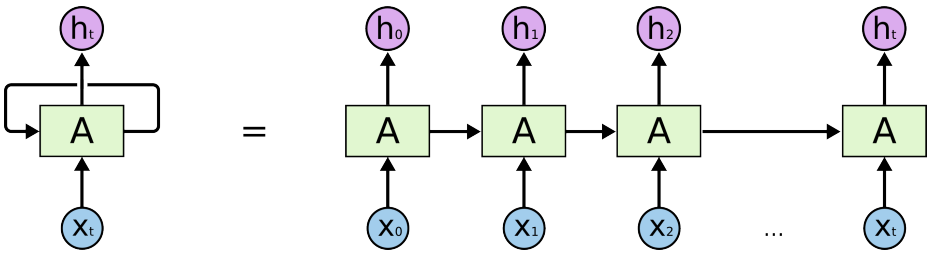


Fig. 6. Recurrent network (unrolled); image from http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

3.4 Long short-term memory (LSTM)

LSTM networks add input/forget/ouput gates to control the information flow [24]. See Figure 7 for illustration of an LSTM cell. Advantages of LSTMs are,

they learn more complex and long-range patterns, and are less overfitting. Disadvantages are, they are more complex, and slower to train. A compatitive alternative are GRU (Gated Recurrent Units, [13]). They have fewer gates and fewer parameters, but often their performance equals LSTMs.

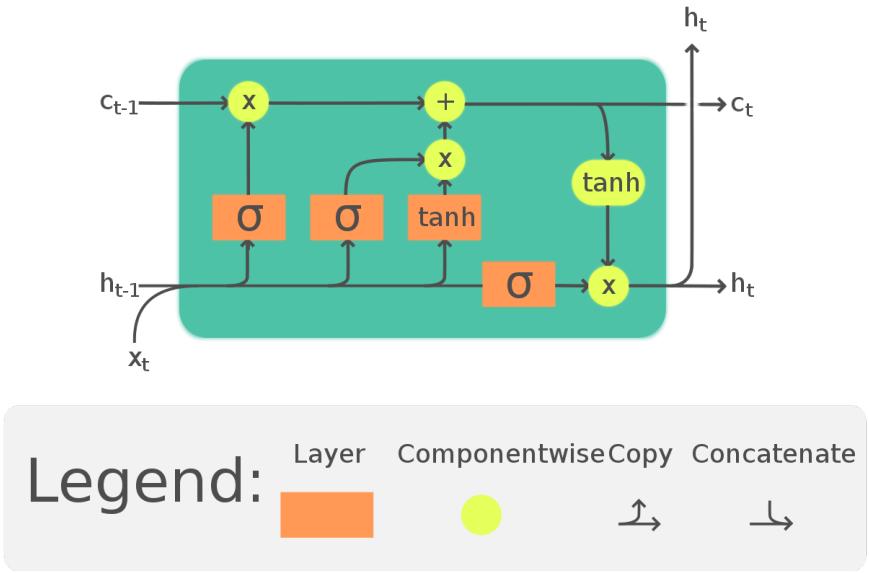


Fig. 7. LSTM cell; image from Wikipedia https://en.m.wikipedia.org/wiki/File:LSTM_Cell.svg.

3.5 ELMO

ELMO stands for "Embeddings from Language Models" [45]. The idea behind is that embedding should depend on context, i.e., one and the same word should have different embeddings depending on its context. The approach uses forward+backward multi-layer LSTM to predict tokens simultaneously from left and right context, thus, creating contextualized word embeddings. Lower layers capture more syntactic properties, and upper layers more semantic properties. Instead of just using the upper layer embeddings, create a weighted sum of the embeddings from each layer and learn the weights that work best for a specific task. Figure 8 illustrates the ELMO architecture. The language model (from left) looks as follows: $p(w_k|w_{k-n}w_{k-n+1}\dots w_{k-2}w_{k-1}$

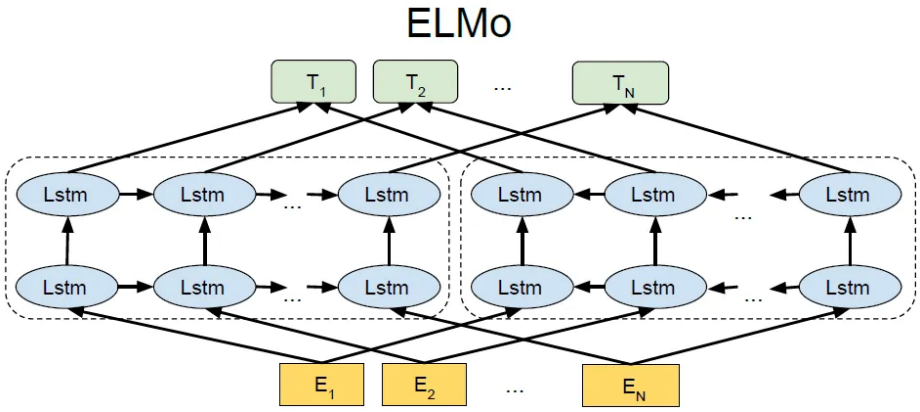


Fig. 8. ELMO architecture; image from [17].

3.6 Transformers

Transformers are encoder-decoder sequence-to-sequence models and were first introduced by Vasvani et al. 2017 [58]. Instead of RNN/CNN use a feed-forward architecture with "multi-head self attention". A Transformer model comprises One feed-forward "column" per token, horizontally interconnected via self-attention. See Figure 9 for the model architecture representing a single column.

Transformers: Tokenization and Inputs What is a technically good input? Per-wordform tokens are large in number, some wordforms very rare, and internal word structure is ignored. Per-byte tokens are small in number, but it is hard for the model to learn meaningful representations for this kind of tokens. Therefore, sub-word tokenization has been introduced as a compromise.

Subword Tokenization A number of approaches for subword tokenization have been proposed: Sennrich et al. 2026 [51] introduced Byte-Pair Encoding BPE, i.e., start with a base vocabulary (unicode, byte) from a list of unique words, merge the tokens according to frequency to create new tokens from the base tokens, add new tokens and repeat. BPE also learns merge rules for splitting unseen words. The tokenizer must be trained based on the full training set vocabulary. WordPiece [50] is another approach. It is similar to BPE, but uses a different merge criterion.

Kudo 2018 introduces Unigram [30] which is initialized with a large vocabulary of words, substrings and characters, and is progressively trimmed to reduce the vocabulary size. Unigram is not used on its own but together with SentencePiece. SentencePiece is introduced by Kudo and Richardson 2018 [31] as a method for subword tokenization which is better suited for languages that do not separate words by spaces. SentencePiece treats the text input as a stream,

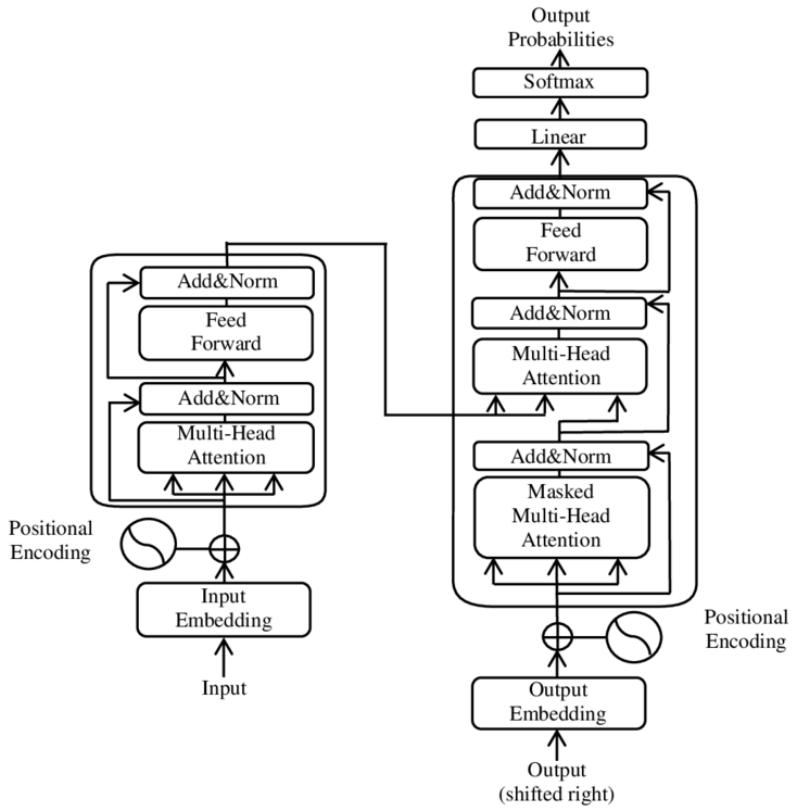


Fig. 9. Transformer model architecture (single column); image from https://de.wikipedia.org/wiki/Transformer_%28Maschinelles_Lernen%29.

whitespace is seen as an ordinary character. Detokenization is used to reconstruct the original string including whitespace from subwords.

Transformers: Self-Attention and Multi-head Self Attention Attention was used before Transformers e.g. in seq2seq LSTMs: depending on input, use representation from different states differently by learning weights for combining them. With the introduction of the Transformer architecture [58] attention became everything you needed. In each layer of the Transformer the representation is a weighted sum of all value vectors in all columns. The weight is calculated from the similarity between query vector of the current column, and the key vectors of all columns. Query, key, value vectors are calculated from the current layer input via trainable matrices. See Figure 10 for an illustration of Transformer self-attention.

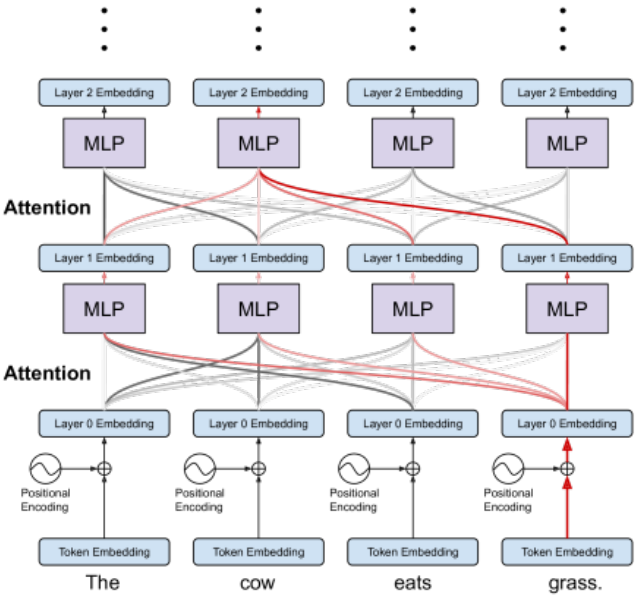


Fig. 10. Transformer self-attention; image from [11].

Given the inputs x_i (each input is an embedding of some dimension d, three kinds of vectors are calculated by multiplying the input embeddings with a matrix of learned weights: a query vector using a matrix Q: $q_i = x_i Q$, a key vector using a matrix K: $k_i = x_i K$ and a value vector using a matrix V: $v_j = x_i V$. For each combination of input locations i and j, the an attention weight a_{ij} is calculated as the dot product between q_i and k_j: $a_{ij} = q_i k_j^\top$. These weights

are then scaled by dividing through the square root of the vector dimensionality producing w_{ij} and normalized by passing them through a softmax function producing s_{ij}

$$w_i j = a_{ij}/\sqrt{(d)}$$
$$s_i j = \frac{\exp(w_{ij})}{\sum_{j'=1}^{n} \exp(w_{ij'})}$$

These values are then used to calculate the output value as the weighted sum of all value vectors using the normalized dot products as weights.

$$o_i = \sum_j s_{ij} v_j$$

The contribution of each value vector to the output at some position i thus depends on the similarity between the query vector of position i and the key vector corresponding to the position of the value vector.

The actual calculation is performed using more efficient matrix multiplication and in transformer models several attention heads (*multi-headed attention*) are calculated in parallel with different learned weight matrices, in order to provide more flexibility to focus on different kinds of influencing positions at the same time.

Transformers: Encoding

Positional Encoding Since the result of self-attention is just a weighted sum, there is no way to find out where each value comes from. Therefore, position embeddings PE are added to the input embeddings, where each dimension of the position embedding contains the value of a sine / cosine function that depends on position and dimension. For offset k, PE_{pos+k} can be calculated as a linear function of PE_{pos}

Full Encoder is characterised as follows: Each column has input embeddings+positional embeddings, followed by multiple layers, each of which has: multi-head attention, layer normalization per column (norm), residual connection adding the input of the layer to the output (add), feed-forward dense layer with add+norm, per-column nonlinearity (ReLU) and dropout per sub-layer.

3.7 Bidirectional Encoder Representations from Transformers (BERT)

BERT [16] uses the Transformer encoder only. BERT is designed to solve many different NLP tasks. The first token is always a special classification token CLS,

i.e., the first column is especially reserved for classification. Segment embeddings are added in addition to positional embeddings which allows to deal with several segments/sentences of text. The model gets pre-trained on large amounts of text. There are different BERT models, see for instance BERT Base cased/uncased with 12 layers, 768 hidden, 12 attention heads, 110M parms, or BERT Large cased/uncased with 24 layers, 1024 hidden, 12 attention heads, 340M parms, just to mention the two fundamental ones.

3.8 BERT: Pre-training and Fine-tuning / Downstream Tasks

Pre-training BERT pre-training has two tasks: creating a masked language model LM and doing next sentence prediction.

Masked Language Model (MLM) The goal is to mask a word/token in the input and try to predict it from the context (actually use MASK 80%, random token 10%, unchanged token 10%), only use masked tokens for the loss function. For this a classification head with a dense layer and softmax is added on top of all columns except the CLS column.

Next Sentence Prediction Input are two sentences separated by a special SEP token. Additional segment embeddings encode which tokens belong to which sentence. A binary classification tasks decides whether the 2nd sentence would follow the 1st one or not. For this a classification head with a dense layer and softmax is added on top of the CLS column.

Fine-tuning Fine-tuning means to load a pre-trained model, add appropriate head(s) and train on new data. The LM layers can be "frozen", i.e., their weights are NOT adapted to the new data, have adapted learning rate or just get fully trained. Fine-tuning is used ot adjust the Transformer model to downstream tasks, e.g., specific classification tasks such as sentiment identification or the identification of sexist utterances, hate speech etc. See Figure 11 for a comparison of pre-training and fine-tunig. See the next paragraph for a few more words on downstream tasks.

Downstream Tasks We distinguish different classes of downstream tasks. Many subtasks can be based on these:

- Sequence tagging: per-column classification head over label set (POS tags, IOB codes), optionally followed by a CRF layer.
- Classification: add a single dense layer after the output of the CLS column, followed by softmax.
- Question answer identification: find start/end token of answer for a question in context: softmax over all context token column outputs after dot product with start / end vectors.
- Get contextual word vectors: just forward the text and use the column outputs.

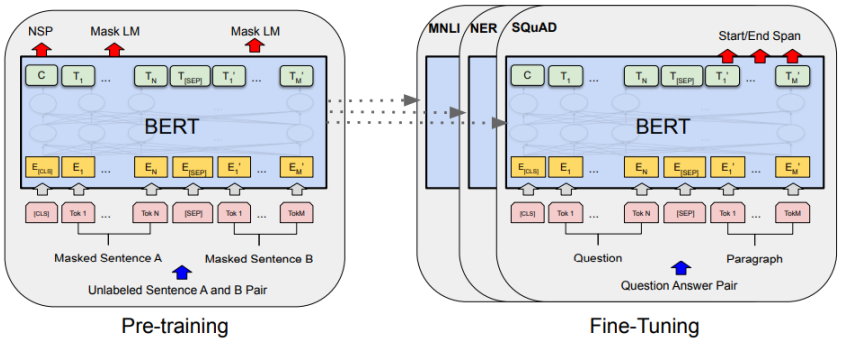


Fig. 11. BERT pre-training versus fine-tuning; image from [17].

3.9 BERT: Zero Shot Classification

Zero shot classification is an approach where the knowledge stored in the BERT language model from being pre-trained on a large text corpus is used to solve specific classification tasks without any additional training on a trainingset specific to that task. For a given task, the model should be used to assign one of several classification labels to some text. For example, given a comment, assign a label to describe the sentiment of the comment as "positive", "negative" or "neutral". There are several possible approaches to do this, one that is commonly used is to treat the classification as a natural language inference problem where the model is presented with two sentences and has to decide if the second sentence follows from the first. The text can then be presented as the first sentence and each of the candidate labels, possibly as part of a template like "this text expresses a [LABEL] sentiment" as the second sentence and the classification is done by picking the label where the probability for the second sentence to follow from the first one is highest.

Alternately the text and label can be combined to a sentence forming a hypothesis sentence (e.g. "[TEXT] expresses a [LABEL] sentiment" and the language model probability of each of the labels in the context of the template can be used to determine the most likely classification.

3.10 Transformer Encoder-Decoder

The original Transformer has both encoder and decoder, and was intended for Machine Translation MT. For MT, a seq2seq model is required which takes an input sequence of tokens from the source language and generate a corresponding output sequence of tokens from the target language. Before the introduction of the Transformer architecture, this has been done using e.g. LSTMs. The decoder is similar to the encoder, but adds another masked multi-head attention where only previous positions get attended to. The decoder outputs one token at at time. To generate the next output token, the token sequence generated so far

(or the special start of sentence token SOS) gets fed to the input right shifted which is called causal autoregressive text generation. The output embedding goes through softmax to generate the token probabilities. Rather than always using the most probable token, use beam search to find better (higher overall probability) combinations of tokens. Stop when the end of sentence (EOS) token is the most probable output. Figure 12 illustrates the encoder-decoder process.

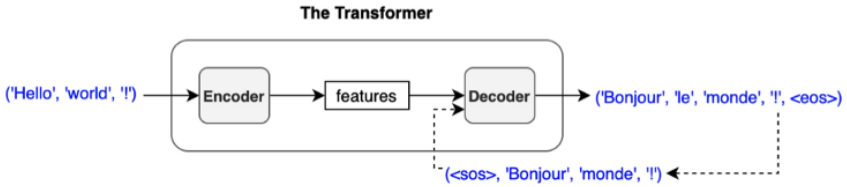


Fig. 12. Transformer encoder-decoder; image from https://kikaben.com/transformers-encoder-decoder/.

3.11 Decoder only transformer: GPT-x

Radford et al. 2018 [46] propose generative pre-training GPT. They use the Transformer decoder to pre-train a language model, and then use fine-tuning on textual entailment, similarity, question answering, commonsense reasoning. Their proposed model had 12 layers, 768 dimensions, 12 attention heads, 3072 hidden nodes for feed forward, and a context size of 512 tokens. See Figure 13 for an illustration of the GPT architecture. BPE is used for tokenization. Since its introduction GPT has made tremendous progress in terms of model size and capabilities, and at the same time has become more and more sealed off. See for instance:

- GPT-2 [47] is comparable to GPT with small modifications comprising the layer norm at beginning of a sub-block, the final normalization after attention, and the context size is now 1024 tokens.
- GTP-3 [10] is like GPT-2, but with alternating dense and sparse attention patterns in the layers. The model has 96 layers, 12288 dimensions, 96 attention heads, and a context window of 2048 tokens. A 3.2M batch size, 175B parameters, no fine-tuning of the base model, and is pre-trained on 500B tokens of text (93% English). It shows remarkable zero-shot capabilities. It is a Large Language Model (LLM), meaning the model itself is huge and it is trained on a huge dataset.
- GPT-3.5 is the model where no accessible scientific papers exist. Overall, fewer information is provided from OpenAI on this model. GPT-3.5 is based on GPT-3, fine-tuned and improved using Reinforcement Learning from Human Feedback (RLHF). Several versions with different properties exist (con-

text size 4k-16k, 1.3B-175B parmeters not publicised at the time of model publication.[12]

– GPT-4 [42] is a multimodal (text and image inputs) model. It is trained on both publicly available and licensed data, with a context size of 32k tokens. The model is fine-tuned on multiple tasks, again RLHF is used to improve the model's output behaviour. No further details are given about the model architecture, including the model size. Parameters are unknown, however, estimated to be around 1T.

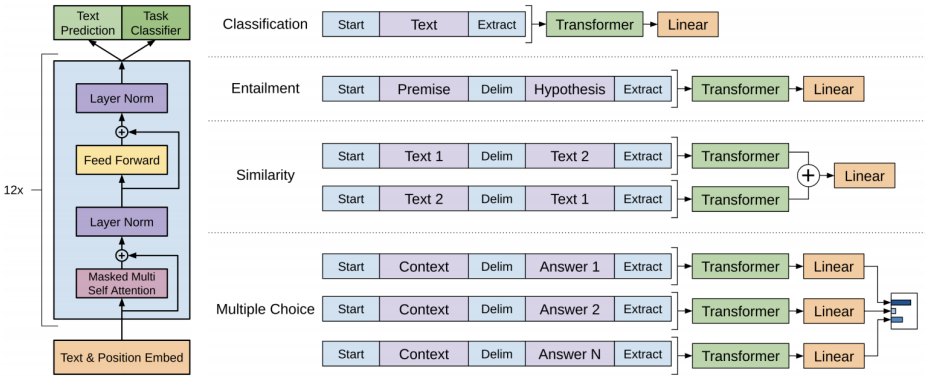


Fig. 13. GPT Decoder-only model; image from [46].

3.12 Reinforcement Learning from Human Feedback (RLHF)

RLHF is important for LLMs as it adjusts the model according to human feedback and, thus, makes it more relevant. The RLHF process is as folows: Start with a pre-trained language model and optionally fine-tune it on a variety of tasks. Then gather many (task-specific) prompts (e.g. from deployed LM users) and generate responses from current and possibly also other LMs. Human annotators order responses in order of preference, and generate a score using methods that are similar to the Elo [19] or similar rating systems used in tournament chess (the method calculates an overall quality score based on pairwise comparisons). Next, a scoring model is trained on the prompt/response + score data. Use e.g. Proximal Policy Optimization (PPO) (or NLPO, A2C, ..) with a copy of the LM to tune some or all of its parameters, see https://huggingface.co/blog/deep-rl-ppo.

[12] In the meantime there is an extended list with newer models, cf. https://platform.openai.com/docs/models/gpt-3-5-turbo.

3.13 Other LLMs

The number of pre-trained LLMs, both with and without fine-tuning (FT) or RLHF grows rapidly. While recent models from OpenAI or Google are closed and restricted (CS/R), open source (OS) models start to appear on the scene (it is open to debate though what conditions must be satisfied to make a model open source: is it sufficient if the model weights and the code to use the model is available with an open-source license, should all the code that was used for training be available as well, should all the data used for training and tuning be available etc.)

In the following, we list a number of models, being aware that such lists age very quickly:

- PaLM/PaLM2 [14], [2]: (8B-)540B parameters, 118 Layers, 48 heads, 18432 dimensions, trained on 780B token corpus (123 languages, 78% EN, 24 programming languages). PROPRIETORY
- LLaMA [56] and many others derived from it, 7B-65B parmeters, 64 heads, 8192 dimensions, 80 layers, trained on 1.4T tokens (unknown language distribution). OS
- BLOOM [49]: 176B parmeters, Responsible AI License (RAIL), ROOTS corpus (1.6TB, 46 languages, 13 programming languages, 70% non-English). OS
- GALACTICA [53]: Intended for scientific knowledge. Public demo showed hallucinations (i.e., inventing competent sounding stuff which is) got pulled off. The modle is trained on papers, code, KBs, .. 106B tokens. 125M-120B parmeters, 96 layers, 10240 dimensions, 80 heads.
- Falcon [43]: improves the pre-training data by filtering and deduplicating, a 600B token subset of the 5000B token set for training is open-sourced. The model comes in two versions, a larger one with 40B parameters and a smaller one with 7B parmeters. FT versions are available, OS
- Sparrow [21]: RLHF for different aspects of being capable of "good dialogue". The modle has 70B parmeters, and can lookup information. CS/R
- LaMDA [54]: The model has up to 137B parameters, and 64 layers. It is trained on documents, dialogs, 1.5T tokens (¿ 90% English). It allows to query knowledge sources, a translator, and a calculator.
- Apart from the models listed above, there are many more and their numbe ris rapidly growing.

3.14 LLMs and LLM-based NLP: (some) Advantages and Issues

In the following, we will list some major advantages of and issues with LLMs and LLM-based NLP. The pros are marked with +, the cons with -.

+ Huge pre-training improves overall performance on many tasks.
+ Capable of solving many tasks with the same or just slightly modified model (zero/few-shot).

+ Capable of resolving common ambiguities, e.g. Winograd Schema Challenge [27].

- "Hallucinations": Today's LLMs are prone to producing competent-sounding nonsense. This is a variation of previously often-seen (catastrophic) failure modes with ML, especially with DNNs.
- Hard to control bias, performance and bias depends on selection of TB of data.
- As with most ML-based approaches: The modle does not know what it does not know.
- As with most DNN-based ML approaches: missing model explainability.
- Currently: not easily accessible (effort in hardware, data, know-how). We depend on large-player models, APIs and solutions.

Pre-training Corpora Another issue relates to the corpora used for pre-training LLMs. Mostly "accessible" data is used such as web crawls, wikipedia, fan-fiction, etc. Most good data, however, is restricted as it is copyrighted, licensed, hidden, not digital, etc. There is an imbalance on topics which have freely available text, e.g., computer science versus social sciences, which obviously influences the knowledge of an LLM. Moreover, there is a huge imbalance on the availability and usage of data in non-English languages. which is even worse for low-resource languages, local idioms and dialects. As Falcon has shown, cleaning and filtering data already improves the LLM (cf. the description if the Falcon model above). There is also a huge problem of opinionated texts, propaganda, religions, ideologies which enter LLMs through their training data, and there is a problem of time: Facts often relate to a time span (e.g., who is the US president), new facts need to get constantly added, and data may become obsolete, out-dated and wrong over time. All this leads the the need for models being capable both of life-long learning and of steerable forgetting (as opposed to current uncontrollable catastrophic forgetting).

4 Corpus Annotation

Almost all the tasks we discussed in the course of this chapter need training material. For supervised learning, human annotation is needed, e.g., What is the POS tag of a word? What is a Named Entity in a sentence and which entry in the knowledge base does it refer to? Does sentence A entail sentence B? Is that text misogynist? Does this tweet contain hate speech, directed at whom? Is this a good and factually correct answer to a chat question?, and so forth. For unsupervised pre-training, on the contrary, we need large amounts of quality documents across many different languages, domains and writing styles.

4.1 Numerical Evaluation of Human Annotation Quality

Quality annotations are created by at least 2 independent (expert) annotators who are trained on task-specific annotation guidelines. Cases of disagreement

are resolved by mutual agreement, a third annotator, or by majority voting if there are more than 2 annotators. This concept of quality annotation requires a single truth which, however, does not exist in many real-world cases, e.g., what is considered misogynist, in which context, by whom, to which extent.

Measuring Inter-annotator Agreement/Reliability Inter-annotator agreement (IAA) assesses the agreement among individual annotators on an annotation task. IAA reflects how likely the annotators are to achieve the same annotation results using the same guidelines. Both the proficiency of the annotators and the quality of annotation guidelines have an effect on IAA. There is a number of measures for assessing IAA, ranging from simple (e.g., percent-agreement) to rather complex measures (e.g., Krippendorff' Alpha).

Percent agreement is the number of annotations agreed upon divided by the number of annotations in total. With 2 annotators annotating all data of the same dataset. As percent agreement does not account for chance agreement, the amount of true agreement may be overestimated. Moreover, there is no consensus on the significance of the numeric result in the research community, so the numbers are hard to interpret.

Krippendorff's Alpha is a complex measure, allowing for any number of annotators. There is no precondition that all data must be annotated by each annotator, and the measure minimizes the effect of chance. This leads to a better general interpretability of the numeric results, e.g., a Krippendorff's Alpha of > 0.9 is generally acceptable, > 0.8 shows fair reliability, and > 0.7 is tolerable (cf. [41].

More Reasons for Differing Data Quality Data quality heavily effects model quality: The model my be noisy because of bad data quality, e.g., low-paid workers annotating huge amounts of data, non-expert annotators do the annotations. Do the data reflect a bandwidth of (task-relevant) views, opinions, conceptions? Different groups of annotators may introduce very different biases, e.g. with culturally differing concepts as obscenity, offensiveness, sexism. There is a trade-off between employing diverse groups of annotators covering different opinions, however, at the cost low IAA versus employing more homogenous groups of annotators with a fair chance to reach higher levels of IAA but with a risk of potentially putting a too restricted lens on the data. There is also an ethical component in data annotation, i.e., the expected model output defines which data are collected and how they are annotated. Therefore, transparency of the data acquisition, data cleaning and annotation processes is required, e.g., datasheets for datasets [20], model cards [39].

Another issue is, for which languages is effort of extensive data collection and annotation invested? Here English surpasses all other languages. For which topics and tasks are datasets created? What about the life of a datset after annotation? On the one hand we want static corpora for ML evaluation and comparison. On the other hand we want to correct mistakes, improve and update the dataset, not to forget the obligation to change the dataset, e.g. to remove deleted tweets.

Dealing with Annotator Disagreement As already said, the usual view on annotated data is that there is one true label, and annotator disagreement is a sign of error (noise). The usual disambiguation strategy is to use the majority label, to let an expert meta annotator decide, or to get additional annotations and chose the majority label. For many complex concepts, however, there is no definite single true label. Therefore, we need to think about alternatives to the current one-true-label concept.

An alternative is to work with soft-labels, i.e., get a label distribution $p(y|x)$ which the model should learn. Uma et al. 2021 or Leonardelli et al. 2023 [57], [33] propose learning with disagreements. Others perform model calibration [38] to human uncertainty [3]. These approaches also need different evaluation scores, e.g., cross-entropy, correlation, KL-divergence. There are similar issues with machine translation, i.e., which of the many translations for a text are acceptable/correct, and what is a good amount of variation between them? A major problem is the low number of annotator labels which may not reflect the population distribution.

5 Opinionated Short List of Related NLP Tools

The following is a list of NLP tools we consider useful for everybody which plans to build NLP applications. (P, J and C stand for the following programming languages P=Python, J=Java, C=C/C++):

- Huggingface[13] (P): transformer-related code, model-zoo, data
- Pytorch[14] (P): Deep Learning Library
- Spacy[15] (P), Stanza[16] (P), CoreNLP[17] (J): pipeline-based NLP, models for several languages
- Online services: GoogleNLP[18], Perspective[19], ELG[20]
- GateNLP[21] (P): integrates most of the above, good abstractions
- Gensim[22] (P), fastText[23] (C+P), AllenNLP[24] (P), FLAIR[25] (P)
- Label-studio[26], Teamware[27], Amazon Mechanical Turk[28]: human annotation

[13] https://huggingface.co/
[14] https://pytorch.org/
[15] https://spacy.io/
[16] https://stanfordnlp.github.io/stanza/
[17] https://stanfordnlp.github.io/CoreNLP/
[18] https://cloud.google.com/natural-language
[19] https://perspectiveapi.com/
[20] https://live.european-language-grid.eu/
[21] https://gatenlp.github.io/python-gatenlp/
[22] https://radimrehurek.com/gensim/
[23] https://fasttext.cc/
[24] https://allenai.org/allennlp
[25] https://flairnlp.github.io/
[26] https://labelstud.io/
[27] https://gatenlp.github.io/gate-teamware/development/
[28] https://www.mturk.com/

– Scipy[29], Scikit-Learn[30]: lower level Python packages

6 Outlook

The main progress in NLP obviously lies in highly contextualized representations through massive pre-training, leading to large improvements in the performance on traditional tasks and impressive new systems like conversational AI. These developments make understanding and researching the following aspects more and more important:

– Knowing not to know
– Explanation of results on both the domain and model level
– How to evaluate models trained on a huge area of topics, facts, knowledge?
– How to make models and research more democratic: who can afford to train a 1T parameters model?
– How to deal with opinions, ideologies, disagreement, emotions represented in those models?
– Representation grounding: link concepts to images, videos, sensory data, robot-actions

We hope this chapter has helped to give you an idea of how NLP/ML research has developed over time and how NLP/ML research has become more interesting than ever! Enjoy!

References

1. Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., Collins, M.: Globally normalized transition-based neural networks. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (2016)
2. Anil, R., Dai, A.M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., Chu, E., Clark, J.H., Shafey, L.E., Huang, Y., Meier-Hellstern, K., Mishra, G., Moreira, E., Omernick, M., Robinson, K., Ruder, S., Tay, Y., Xiao, K., Xu, Y., Zhang, Y., Abrego, G.H., Ahn, J., Austin, J., Barham, P., Botha, J., Bradbury, J., Brahma, S., Brooks, K., Catasta, M., Cheng, Y., Cherry, C., Choquette-Choo, C.A., Chowdhery, A., Crepy, C., Dave, S., Dehghani, M., Dev, S., Devlin, J., Díaz, M., Du, N., Dyer, E., Feinberg, V., Feng, F., Fienber, V., Freitag, M., Garcia, X., Gehrmann, S., Gonzalez, L., Gur-Ari, G., Hand, S., Hashemi, H., Hou, L., Howland, J., Hu, A., Hui, J., Hurwitz, J., Isard, M., Ittycheriah, A., Jagielski, M., Jia, W., Kenealy, K., Krikun, M., Kudugunta, S., Lan, C., Lee, K., Lee, B., Li, E., Li, M., Li, W., Li, Y., Li, J., Lim, H., Lin, H., Liu, Z., Liu, F., Maggioni, M., Mahendru, A., Maynez, J., Misra, V., Moussalem, M., Nado, Z., Nham, J., Ni, E., Nystrom, A., Parrish, A., Pellat, M., Polacek, M., Polozov, A., Pope, R., Qiao, S., Reif, E., Richter, B., Riley, P.,

[29] https://scipy.org/
[30] https://scikit-learn.org/stable/

Ros, A.C., Roy, A., Saeta, B., Samuel, R., Shelby, R., Slone, A., Smilkov, D., So, D.R., Sohn, D., Tokumine, S., Valter, D., Vasudevan, V., Vodrahalli, K., Wang, X., Wang, P., Wang, Z., Wang, T., Wieting, J., Wu, Y., Xu, K., Xu, Y., Xue, L., Yin, P., Yu, J., Zhang, Q., Zheng, S., Zheng, C., Zhou, W., Zhou, D., Petrov, S., Wu, Y.: Palm 2 technical report (2023)

3. Baan, J., Aziz, W., Plank, B., Fernández, R.: Stop measuring calibration when humans disagree. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (2022)
4. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017)
5. Bolukbasi, T., Chang, K.W., Zou, J.Y., Saligrama, V., Kalai, A.T.: Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 29. Curran Associates, Inc. (2016), https://proceedings.neurips.cc/paper_files/paper/2016/file/a486cd07e4ac3d270571622f4f316ec5-Paper.pdf
6. Bouckaert, R.R.: Efficient auc learning curve calculation. In: Australasian joint conference on artificial intelligence. pp. 181–191. Springer (2006)
7. Bougouin, A., Boudin, F., Daille, B.: Topicrank: Graph-based topic ranking for keyphrase extraction. In: International joint conference on natural language processing (IJCNLP). pp. 543–551 (2013)
8. Brants, S., Dipper, S., Eisenberg, P., Hansen-Schirra, S., König, E., Lezius, W., Rohrer, C., Smith, G., Uszkoreit, H.: Tiger: Linguistic interpretation of a german corpus. Research on language and computation **2**, 597–620 (2004)
9. Brown, P.F., Della Pietra, V.J., deSouza, P.V., Lai, J.C., Mercer, R.L.: Class-based n-gram models of natural language. Computational Linguistics **18**(4), 467–480 (1992), https://aclanthology.org/J92-4003
10. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: Advances in Neural Information Processing Systems 33. pp. 1877–1901 (2020)
11. Brunner, G., Liu, Y., Pascual, D., Richter, O., Wattenhofer, R.: On the validity of self-attention as explanation in transformer models. arXiv preprint arXiv **1908.04211**(40) (2019)
12. Campos, R., Mangaravite, V., Pasquali, A., Jorge, A., Nunes, C., Jatowt, A.: Yake! keyword extraction from single documents using multiple local features. Information Sciences **509**, 257–289 (2020)
13. Cho, K., Van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259 (2014)
14. Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito,

D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A.M., Pillai, T.S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., Fiedel, N.: Palm: Scaling language modeling with pathways. arXiv preprint arXiv **2204.02311** (2022)

15. Clevert, D., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus). In: Bengio, Y., LeCun, Y. (eds.) 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings (2016), http://arxiv.org/abs/1511.07289
16. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language (NAACL2019) (2019)
17. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-1423, https://aclanthology.org/N19-1423
18. Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L., Strassel, S., Weischedel, R.: The automatic content extraction (ace) program-tasks, data, and evaluation. In: LREC (2004)
19. Elo, A.E.: The proposed uscf rating system, its development, theory, and applications. Chess Life **22**(8), 242–247 (1967)
20. Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J.W., Wallach, H.M., III, H.D., Crawford, K.: Datasheets for datasets. CoRR **abs/1803.09010** (2018), http://arxiv.org/abs/1803.09010
21. Glaese, A., McAleese, N., Trebacz, M., Aslanides, J., Firoiu, V., Ewalds, T., Rauh, M., Weidinger, L., Chadwick, M., Thacker, P., Campbell-Gillingham, L., Uesato, J., Huang, P.S., Comanescu, R., Yang, F., See, A., Dathathri, S., Greig, R., Chen, C., Fritz, D., Elias, J.S., Green, R., Mokrá, S., Fernando, N., Wu, B., Foley, R., Young, S., Gabriel, I., Isaac, W., Mellor, J., Hassabis, D., Kavukcuoglu, K., Hendricks, L.A., Irving, G.: Improving alignment of dialogue agents via targeted human judgements. arXiv preprint arXiv **2209.14375** (2022)
22. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Gordon, G., Dunson, D., Dudík, M. (eds.) Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 15, pp. 315–323. PMLR, Fort Lauderdale, FL, USA (11–13 Apr 2011), https://proceedings.mlr.press/v15/glorot11a.html
23. Grishman, R., Sundheim, B.M.: Message understanding conference-6: A brief history. In: COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics (1996)
24. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
25. Iyyer, M., Manjunatha, V., Boyd-Graber, J., Daumé III, H.: Deep unordered composition rivals syntactic methods for text classification. In: Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th inter-

national joint conference on natural language processing (volume 1: Long papers) (2015)
26. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1746–1751 (2014)
27. Kocijan, V., Davis, E., Lukasiewicz, T., Marcus, G., Morgenstern, L.: The defeat of the winograd schema challenge. arXiv preprint arXiv **2201.02387** (2022)
28. Kolitsas, N., Ganea, O.E., Hofmann, T.: End-to-end neural entity linking. In: Proceedings of the 22nd Conference on Computational Natural Language Learning (2018)
29. Krishnan, V., Ganapathy, V.: Named entity recognition. Stanford Lecture CS229 (2005)
30. Kudo, T.: Subword regularization: Improving neural network translation models with multiple subword candidates. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (2018)
31. Kudo, T., Richardson, J.: Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In: EMNLP 2018. p. 66 (2018)
32. Lafferty, J., McCallum, A., Pereira, F.C.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning. p. 282–289. ICML '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)
33. Leonardelli, E., Abercrombie, G., Almanea, D., Basile, V., Fornaciari, T., Plank, B., Rieser, V., Uma, A., Poesio, M.: Semeval-2023 task 11: Learning with disagreements (lewidi). arXiv preprint arXiv **2304.14803** (2023)
34. Li, J., Sun, A., Han, J., Li, C.: A survey on deep learning for named entity recognition. IEEE Transactions on Knowledge and Data Engineering **34**(1), 50–70 (2020)
35. Marcus, M., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of english: The penn treebank. Computational Linguistics **19**(2), 313–330 (1993), https://aclanthology.org/J93-2004
36. Mihalcea, R., Tarau, P.: Textrank: Bringing order into text. In: Proceedings of the 2004 conference on empirical methods in natural language processing (2004)
37. Mikolov, T., Yih, W.t., Zweig, G.: Linguistic regularities in continuous space word representations. In: Vanderwende, L., Daumé III, H., Kirchhoff, K. (eds.) Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 746–751. Association for Computational Linguistics, Atlanta, Georgia (Jun 2013), https://aclanthology.org/N13-1090
38. Minderer, M., Djolonga, J., Romijnders, R., Hubis, F., Zhai, X., Houlsby, N., Tran, D., Lucic, M.: Revisiting the calibration of modern neural networks. Advances in Neural Information Processing Systems **34**, 15682–15694 (2021)
39. Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., Spitzer, E., Raji, I.D., Gebru, T.: Model cards for model reporting. CoRR **abs/1810.03993** (2018), http://arxiv.org/abs/1810.03993
40. Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. Lingvisticae Investigationes **30**(1), 3–26 (2007)
41. Nili, A., Tate, M., Barros, A., Johnstone, D.: An approach for selecting and using a method of inter-coder reliability in information management research. International Journal of Information Management **54**, 102154 (2020)
42. OpenAI: GPT-4 Technical Report. ArXiv **abs/2303.08774** (2023)

43. Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, R., Cappelli, A., Alobeidli, H., Pannier, B., Almazrouei, E., Launay, J.: The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only. arXiv preprint arXiv **2306.01116** (2023)
44. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (2014)
45. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep Contextualized Word Representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 2227–2237 (2018)
46. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (2018), https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf
47. Radford, A., Wu, J., Rewon, C., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners (2019), https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf
48. Roberts, C.: Information structure: Towards an integrated formal theory of pragmatics. Semantics and pragmatics **5**, 6–1 (2012)
49. Scao, T.L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A.S., Yvon, F., Gallé, M., Tow, J., Rush, A.M., Biderman, S., Webson, A., Ammanamanchi, P.S., Wang, T., Sagot, B., Muennighoff, N., del Moral, A.V., Ruwase, O., Bawden, R., Bekman, S., McMillan-Major, A., Beltagy, I., Nguyen, H., Saulnier, L., Tan, S., Suarez, P.O., Sanh, V., Laurençon, H., Jernite, Y., Launay, J., Mitchell, M., Raffel, C., Gokaslan, A., Simhi, A., Soroa, A., Aji, A.F., Alfassy, A., Rogers, A., Nitzav, A.K., Xu, C., Mou, C., Emezue, C., Klamm, C., Leong, C., van Strien, D., Adelani, D.I., Radev, D., Ponferrada, E.G., Levkovizh, E., Kim, E., Natan, E.B., Toni, F.D., Dupont, G., Kruszewski, G., Pistilli, G., Elsahar, H., Benyamina, H., Tran, H., Yu, I., Abdulmumin, I., Johnson, I., Gonzalez-Dios, I., de la Rosa, J., Chim, J., Dodge, J., Zhu, J., Chang, J., Frohberg, J., Tobing, J., Bhattacharjee, J., Almubarak, K., Chen, K., Lo, K., Werra, L.V., Weber, L., Phan, L., allal, L.B., Tanguy, L., Dey, M., Muñoz, M.R., Masoud, M., Grandury, M., Šaško, M., Huang, M., Coavoux, M., Singh, M., Jiang, M.T.J., Vu, M.C., Jauhar, M.A., Ghaleb, M., Subramani, N., Kassner, N., Khamis, N., Nguyen, O., Espejel, O., de Gibert, O., Villegas, P., Henderson, P., Colombo, P., Amuok, P., Lhoest, Q., Harliman, R., Bommasani, R., López, R.L., Ribeiro, R., Osei, S., Pyysalo, S., Nagel, S., Bose, S., Muhammad, S.H., Sharma, S., Longpre, S., Nikpoor, S., Silberberg, S., Pai, S., Zink, S., Torrent, T.T., Schick, T., Thrush, T., Danchev, V., Nikoulina, V., Laippala, V., Lepercq, V., Prabhu, V., Alyafeai, Z., Talat, Z., Raja, A., Heinzerling, B., Si, C., Taşar, D.E., Salesky, E., Mielke, S.J., Lee, W.Y., Sharma, A., Santilli, A., Chaffin, A., Stiegler, A., Datta, D., Szczechla, E., Chhablani, G., Wang, H., Pandey, H., Strobelt, H., Fries, J.A., Rozen, J., Gao, L., Sutawika, L., Bari, M.S., Al-shaibani, M.S., Manica, M., Nayak, N., Teehan, R., Albanie, S., Shen, S., Ben-David, S., Bach, S.H., Kim, T., Bers, T., Fevry, T., Neeraj, T., Thakker, U., Raunak, V., Tang, X., Yong, Z.X., Sun, Z., Brody, S., Uri, Y., Tojarieh, H., Roberts, A., Chung, H.W., Tae, J., Phang, J., Press, O., Li, C., Narayanan, D., Bourfoune, H., Casper, J., Rasley, J., Ryabinin, M., Mishra, M., Zhang, M., Shoeybi, M., Peyrounette, M., Patry, N., Tazi, N., Sanse-

viero, O., von Platen, P., Cornette, P., Lavallée, P.F., Lacroix, R., Rajbhandari, S., Gandhi, S., Smith, S., Requena, S., Patil, S., Dettmers, T., Baruwa, A., Singh, A., Cheveleva, A., Ligozat, A.L., Subramonian, A., Névéol, A., Lovering, C., Garrette, D., Tunuguntla, D., Reiter, E., Taktasheva, E., Voloshina, E., Bogdanov, E., Winata, G.I., Schoelkopf, H., Kalo, J.C., Novikova, J., Forde, J.Z., Clive, J., Kasai, J., Kawamura, K., Hazan, L., Carpuat, M., Clinciu, M., Kim, N., Cheng, N., Serikov, O., Antverg, O., van der Wal, O., Zhang, R., Zhang, R., Gehrmann, S., Mirkin, S., Pais, S., Shavrina, T., Scialom, T., Yun, T., Limisiewicz, T., Rieser, V., Protasov, V., Mikhailov, V., Pruksachatkun, Y., Belinkov, Y., Bamberger, Z., Kasner, Z., Rueda, A., Pestana, A., Feizpour, A., Khan, A., Faranak, A., Santos, A., Hevia, A., Unldreaj, A., Aghagol, A., Abdollahi, A., Tammour, A., Haji-Hosseini, A., Behroozi, B., Ajibade, B., Saxena, B., Ferrandis, C.M., McDuff, D., Contractor, D., Lansky, D., David, D., Kiela, D., Nguyen, D.A., Tan, E., Baylor, E., Ozoani, E., Mirza, F., Ononiwu, F., Rezanejad, H., Jones, H., Bhattacharya, I., Solaiman, I., Sedenko, I., Nejadgholi, I., Passmore, J., Seltzer, J., Sanz, J.B., Dutra, L., Samagaio, M., Elbadri, M., Mieskes, M., Gerchick, M., Akinlolu, M., McKenna, M., Qiu, M., Ghauri, M., Burynok, M., Abrar, N., Rajani, N., Elkott, N., Fahmy, N., Samuel, O., An, R., Kromann, R., Hao, R., Alizadeh, S., Shubber, S., Wang, S., Roy, S., Viguier, S., Le, T., Oyebade, T., Le, T., Yang, Y., Nguyen, Z., Kashyap, A.R., Palasciano, A., Callahan, A., Shukla, A., Miranda-Escalada, A., Singh, A., Beilharz, B., Wang, B., Brito, C., Zhou, C., Jain, C., Xu, C., Fourrier, C., Periñán, D.L., Molano, D., Yu, D., Manjavacas, E., Barth, F., Fuhrimann, F., Altay, G., Bayrak, G., Burns, G., Vrabec, H.U., Bello, I., Dash, I., Kang, J., Giorgi, J., Golde, J., Posada, J.D., Sivaraman, K.R., Bulchandani, L., Liu, L., Shinzato, L., de Bykhovetz, M.H., Takeuchi, M., Pàmies, M., Castillo, M.A., Nezhurina, M., Sänger, M., Samwald, M., Cullan, M., Weinberg, M., Wolf, M.D., Mihaljcic, M., Liu, M., Freidank, M., Kang, M., Seelam, N., Dahlberg, N., Broad, N.M., Muellner, N., Fung, P., Haller, P., Chandrasekhar, R., Eisenberg, R., Martin, R., Canalli, R., Su, R., Su, R., Cahyawijaya, S., Garda, S., Deshmukh, S.S., Mishra, S., Kiblawi, S., Ott, S., Sang-aroonsiri, S., Kumar, S., Schweter, S., Bharati, S., Laud, T., Gigant, T., Kainuma, T., Kusa, W., Labrak, Y., Bajaj, Y.S., Venkatraman, Y., Xu, Y., Xu, Y., Xu, Y., Tan, Z., Xie, Z., Ye, Z., Bras, M., Belkada, Y., Wolf, T.: Bloom: A 176b-parameter open-access multilingual language model. arXiv preprint arXiv **2211.05100** (2022)

50. Schuster, M., Nakajima, K.: Japanese and korean voice search. In: 2012 IEEE international conference on acoustics, speech and signal processing (ICASSP) (2012)
51. Sennrich, R., Haddow, B., Birch, A.: Neural Machine Translation of Rare Words with Subword Units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (2016)
52. Taylor, A., Marcus, M., Santorini, B.: The penn treebank: an overview. Treebanks: Building and using parsed corpora pp. 5–22 (2003)
53. Taylor, R., Kardas, M., Cucurull, G., Scialom, T., Hartshorn, A., Saravia, E., Poulton, A., Kerkez, V., Stojnic, R.: Galactica: A large language model for science. arXiv preprint arXiv **2211.09085** (2022)
54. Thoppilan, R., Freitas, D.D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H., Jin, A., Bos, T., Baker, L., Du, Y., Li, Y., Lee, H., Zheng, H.S., Ghafouri, A., Menegali, M., Huang, Y., Krikun, M., Lepikhin, D., Qin, J., Chen, D., Xu, Y., Chen, Z., Roberts, A., Bosma, M., Zhou, Y., Chang, C., Krivokon, I., Rusch, W., Pickett, M., Meier-Hellstern, K.S., Morris, M.R., Doshi, T., Santos, R.D., Duke, T., Soraker, J., Zevenbergen, B., Prabhakaran, V., Diaz, M., Hutchinson, B., Olson,

K., Molina, A., Hoffman-John, E., Lee, J., Aroyo, L., Rajakumar, R., Butryna, A., Lamm, M., Kuzmina, V., Fenton, J., Cohen, A., Bernstein, R., Kurzweil, R., y Arcas, B.A., Cui, C., Croak, M., Chi, E.H., Le, Q.: Lamda: Language models for dialog applications. arXiv preprint arXiv **2201.08239** (2022)
55. Toutanova, K., Manning, C.D.: Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In: Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000) (2000)
56. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G.: Llama: Open and efficient foundation language models. arXiv preprint arXiv **2302.13971** (2023)
57. Uma, A., Fornaciari, T., Dumitrache, A., Miller, T., Chamberlain, J., Plank, B., Simpson, E., Poesio, M.: Semeval-2021 task 12: Learning with disagreements. In: Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021) (2021)
58. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems 30 (2017)

Deep Learning

Günter Klambauer

Abstract. Over the last decade, machine learning and Deep Learning methods have paved their way into all kinds of computational task for molecules. The molecular machine learning research community has made strong progress in a) activity and property prediction, b) representation learning and molecular modeling, c) chemical synthesis and reaction prediction, and d) generative models for molecules. In this talk, we provide an overview over the main deep architectures, such as fully-connected, convolutional, RNN and Transformer architectures. We also provide a perspective of the recent progress in molecular machine learning, on the essential properties that our AIs should have to make a difference, and steps towards such broad AIs.

Reinforcement Learning

Clemens Heitzinger

Abstract. Reinforcement learning is the type of machine learning where an agent interacts with an environment, receives rewards, and learns policies that maximize the sum of all rewards. It is a general concept with many applications, e.g. in decision making, robotics, gaming, and medicine. Reinforcement-learning methods solved Go and power ChatGPT. The lecture starts with an introduction to reinforcement learning and its classical algorithms. Then recent techniques such as deep reinforcement learning and distributional reinforcement learning are discussed, as well as the use of reinforcement learning in large language models. Finally, applications are presented.

Brain-Inspired Computation and Learning

Robert Legenstein

Abstract. The quest of Artificial Intelligence research is to build artefacts that mimic the cognitive capabilities of the human brain. Starting with the McCulloch-Pitts neuron exactly 80 years ago, the brain has always provided inspiration for systems and algorithms in AI research. In this tutorial, I will ask the question how AI research we can make use of our knowledge about the brain. I will discuss the current knowledge about the organisation of computations in brain. Next, I will present efforts that attempt to model such computations and how these models give rise to new machine learning approaches. One immediate application of this research is 'neuromorphic' hardware, that is, hardware that implements computation and learning based on principles borrowed from the brain. I will discuss the advantages and challenges of such neuromorphic systems.