Demystifying Relational Latent Representations

Sebastijan Dumančić and Hendrik Blockeel

Department of Computer Science, KU Leuven Belgium
{firstname.lastname}@cs.kuleuven.be

Abstract. Latent features learned by deep learning approaches have proven to be a powerful tool for machine learning. They serve as a data abstraction that makes learning easier by capturing regularities in data explicitly. Their benefits motivated their adaptation to relational learning context. In our previous work, we introduce an approach that learns *relational* latent features by means of clustering instances and their relations. The major drawback of latent representations is that they are often *black-box* and difficult to interpret. This work addresses these issues and shows that (1) latent features created by clustering are interpretable and capture interesting properties of data; (2) they identify local regions of instances that match well with the label, which partially explains their benefit; and (3) although the number of latent features generated by this approach is large, often many of them are highly redundant and can be removed without hurting performance much.

Keywords: relational learning, deep learning, unsupervised representation learning, clustering

1 Introduction

Latent representations created by deep learning approaches [1] have proven to be a powerful tool in machine learning. Traditional machine learning algorithms learn a function that directly maps data to the target concept. In contrast, deep learning creates several layers of latent features between the original data and the target concept. This results in a multi-step procedure that simplifies a given task before solving it.

The progress in learning such latent representations has predominantly focused on vectorized data representations. Likewise, their utility has been recognized in the relational learning community [2] in which models are learned not only from instances but from their relationships as well [3,4]. The prevalent latent representations paradigm in that direction are *embeddings to vector spaces* [5,6,7]. The core idea behind the embeddings is to replace symbols with numbers and logical reasoning with algebra. More precisely, relational entities are transformed to low-dimensional vectors and relations to matrices or functions. This way of learning latent features corresponds to learning the low-dimensional representations of relational entities and relations. There are many variations of formalizing this problem, but they all follow the same general principle. Assuming facts `p(a,b)` and `r(b,a)`, `a` and `b` are entities whereas `p` and `r` are existing

relations between them. The goal is to find corresponding vectorized representations of a and b, a and b respectively, together with matrix representations of p and r, P and R respectively. More precisely, the goal is to find vectorized representations such that products aPb and bRa have *high values*. In contrast, given a false fact q(a,b) product aQb should have a *low value*.

These embeddings approaches have several drawbacks. First, the latent features created that way have no inherent meaning – they are created to satisfy the aforementioned criteria. This is thus a major obstacle for interpretability of the approach, which is important in many aspects and one of the strengths of relational learning. Second, they need huge amount of data in order to extract useful latent features. Knowledge bases used for training often contain millions of facts. Third, it is not clear how these approaches can handle unseen entities (i.e., an entity not present in the training set and whose embedding is therefore not known) without re-training the entire model.

Recently, Dumančić and Blockeel [8] introduced a complementary approach, titled CUR^{2}LED, that takes a relational learning stance and focuses on learning relational latent representations in an unsupervised manner. Viewing relational data as a hypergraph in which instances form vertices and relationships among them form hyperedges, the authors rely on clustering to obtain latent features. The core component in this approach is a declarative and intuitive specification of the similarity measure used to cluster both instances and their relationships. This consequently makes entire approach more *transparent* with respect to the meaning of latent features, as the intuitive meaning of similarity is precisely specified.

The benefits of latent representations were clearly shown with respect to both performance and complexity. The complexity of models learned on latent features was consistently lower compared to the models learned on the original data representation. Moreover, the models learned with latent features often resulted in improved performance, by a large margin as well. These two results jointly show that latent representations capture more complex dependencies in a simple manner.

In this work we further investigate the properties of relational latent representations created by CUR^{2}LED. We start by asking the question: *what do latent features mean?* We introduce a simple method to extract the meaning of the latent features, and show that they capture interesting properties. We ask next: *what makes latent representation effective?* The initial work showed the benefits of the latent representations, however, no explanation is offered why that is the case. We hope to shed light behind the scene and offer (at least a partial) answer why that is the case.

In the following section we first briefly introduce neighbourhood trees – a central concept of CUR^{2}LED. We then describe an approach used in extracting the knowledge form the latent features, and investigating the properties of such latent representation. The results are presented and discussed next, followed by the conclusion.
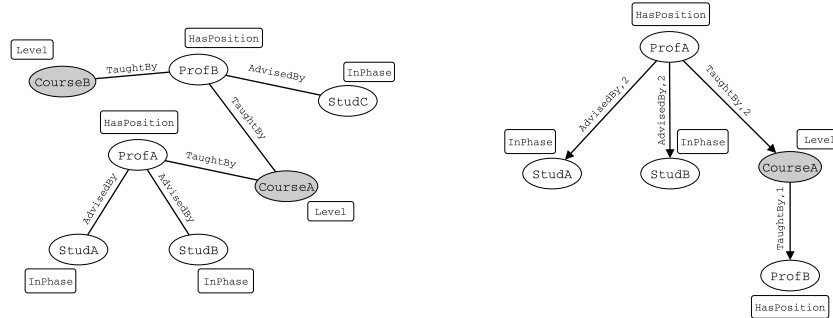
Fig. 1. A snapshot of a knowledge base (left) and the corresponding neighbourhood trees of `ProfA` entity (right). The knowledge base describes students, professors and courses they teach. Entities (people and courses) are represented with node, their attributes with rectangles and relationships with edges. Attribute values are left out for brevity.

2   Neighbourhood trees

The central concept of CUR^2LED is a neighbourhood tree. The neighbourhood tree is a rooted directed graph describing an instance, together with instances it relates to and their properties. Viewing relational data as a hypergraph, the neighbourhood tree provides a summary of all path of a pre-defined length that originate in a particular vertex (see Figure 1).

As instances are represented as neighbourhood trees, two instances are compared by comparing corresponding neighbourhood trees. The authors introduce a versatile and declarative similarity measure [9] that analyses neighbourhood trees over multiple aspects by introducing the following *core similarities*:

– attribute similarity of root vertices
– attribute similarity of neighbouring vertices
– connectivity between root vertices
– similarity of vertex identities in a neighbourhood
– similarity of edge types

Continuing the example in Figure 1, person instances can be clustered based on their own attributes, which yields clusters of professors and students. Clustering person instances based on the vertex identities in their neighbourhood yields clusters of *research groups* – a professor and his students.

These core similarities form basic building blocks for a variety of similarity measure over neighbourhood trees. The final similarity measure is a linear weighted combination of core similarities. Weights simply define a relative importance of core similarities in the chosen similarity measure. The value assignments to the weights defines a *similarity interpretation*. For the details of core similarities and the similarity measure itself see [9].

3 Opening the black box of latent features

Two ideas are central to CUR^2LED. First, it learns latent features by clustering instances and their relations. Second, it uses multiple similarity interpretations (i.e., combinations of core similarities) to obtain a variety of features. Both ideas are realised by means of neighbourhood trees. Instances and relations are represented as (collections of) neighbourhood trees, while similarity interpretation is a result of core similarities which consider only certain parts of neighbourhood trees.

Latent features are learned by CUR^2LED through *repeated clustering of instances and relations and alternating the similarity measure in each iteration*. Each latent feature, corresponding to a cluster of instances, is associated with one latent predicate. Truth instantiations of latent predicates reflect the cluster assignments, i.e., the instantiations of a latent predicate are true for instances that belong to the cluster. Therefore, latent features are defined extensionally and lack an interpretable definition. However, the intuitive specification of the similarity measure (and its core similarities) makes CUR^2LED a transparent method with a clear description which elements of neighbourhood trees make two instances similar. Consequently, discovering the meaning of latent features is substantially easier than with the embedding approaches (and deep learning in general).

3.1 Extracting the meaning of latent features

Each latent feature corresponds to a cluster and the meaning of the features is reflected in the *prototype* of the cluster. To approximate the *mean* or *prototypical* neighbourhood tree, we search for the elements common to all neighbourhood trees forming a cluster. These elements can be either attribute values, edge types or vertex identities. The similarity interpretations used to obtain the cluster limits which elements are considered to be a part of a definition. Moreover, neighbourhood trees [9] are compared by the relative frequencies of their elements, not the existence only. Therefore, to find a mean neighbourhood tree and the meaning of a latent feature, we search for *the elements with similar relative frequencies within each neighbourhood tree forming a cluster*.

To identify such elements, we proceed in three steps illustrated in Figure 2.

1. **Calculate the relative frequencies of all elements within each neighbourhood tree, per level and vertex type**. In case of discrete attributes, that corresponds to a distribution of its values. In case of numerical attributes, we consider its mean value. In case of vertex identities and edge types, we simply look into their frequencies with respect to the depth in a neighbourhood tree. In the example in Figure 2, the neighbourhood tree for `profA` contains two `advisedBy` relations, thus its frequency is $\frac{2}{3}$.
2. **Calculate the mean and standard deviation of relative frequency for each element within a cluster**. In Figure 2, the frequencies of the `advisedBy` elements in individual neighbourhood trees are $\frac{2}{3}, \frac{2}{3}$ and $\frac{1}{3}$. Thus, its mean is 0.55 with a standard deviation of 0.15.
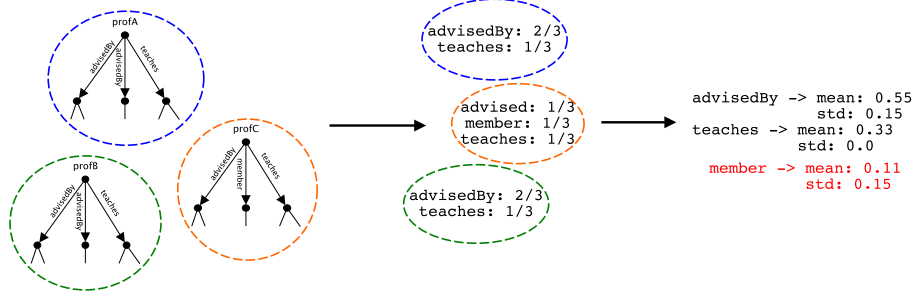
Fig. 2. Discovering the meaning of latent features by analysing their relations. Properties that describe latent features are the ones that have similar relative frequency in all neighbourhood trees. Starting from a cluster of instances viewed as neighbourhood trees (left), the relative frequencies of elements are calculated for each neighbourhood tree (middle). Next, the mean and standard deviation of relative frequencies are calculated for each individual element within the cluster (right). Which elements *explain* the latent features is decided with θ-confidence. Setting θ to 0.3 identifies `advisedBy` and `teaches` as relevant elements (in black).

3. **Select relevant elements.** The final step involves a decision which elements should form a definition of a latent feature. Relevant elements are identified by a notion of θ-*confidence* which captures the allowed amount of variance in order to element to be relevant.

Definition 1. *(θ-confidence) An element with mean value μ and standard deviation σ in a cluster, is said to be θ-confident if $\sigma \in [0, \theta \cdot \mu]$.*

In Figure 2, setting θ to 0.3 makes `advisedBy` a 0.3-confident element, because its standard deviation of 0.15 is within the range $[0, 0.3 \cdot 0.55] = [0, 0.165]$ specified by θ. In contrast, `member` is not a 0.3-confident elements as its standard deviation is outside the range $[0, 0.3 \cdot 0.11] = [0, 0.0363]$.

The above-described procedure explains the latent features in terms of distribution of the elements. This type of explanation does not conform to the standard first-order logic syntax common within relational learning. Despite this reduced readability, these explanations are substantially more transparent and interpretable than the ones produced by the embeddings approaches. Moreover, an additional benefit of this approach is that it increases the expressivity of a relational learner by extensionally defining properties otherwise inexpressible in the first-order logic.

3.2 Properties of latent spaces

Latent features produced by CUR^{2}LED have proven useful in reducing the complexity of models and improving their performance. However, no explanation was offered why that is the case. In the second part of this work, we look into the properties of these latent representations and offer a partial explanation for their usefulness. To answer this question we introduce the following properties: label entropy, sparsity and redundancy.

Entropy and sparsity. Label entropy and sparsity serve as a proxy to a quantification of learning difficulty – i.e., how difficult is it to learn a definition of the target concept. Considering a particular predicate, label entropy reflects a *purity* of its true groundings with respect to the provided labels. Intuitively, if true groundings of predicates tend to predominantly focus on one particular label, we expect model learning to be easier.

Sparse representations, one of the cornerstones of deep learning [10], refer to a notion in which concepts are explained based on local (instead of global) properties of instance space. Even though many properties might exist for a particular problem, sparse representations describe instances using only a small subset of those properties. Intuitively, a concept spread across a small number of local regions is expected to be easier to capture than a concept spread globally over an entire instance space.

Quantifying sparsity in relational data is a challenging task which can be approached from multiple directions – either by analysing the number of true groundings or interaction between entities, for instance. We adopt a simple definition: the number of true groundings of a predicate. Label entropy and sparsity jointly describe a compelling property of data representation – instances space is divided in many local regions that match labels well and consequently make learning substantially easier.

Redundancy. A downside of CUR^{2}LED is the high number of created features. Despite their proven usefulness, a high number of latent features enlarges the search space of a relational model and increases the difficulty of learning. As similarity interpretations are provided by the user, it is possible that almost identical clusterings are obtained with different similarity interpretations. Thus, if many of the features are redundant, removing them simplifies learning. We measure the redundancy with the *adjusted Rand index* (ARI) [11], a standard measure for overlap between clusterings, and study its impact on the performance.

To evaluate the influence of redundant features, we modify CUR^{2}LED by adding an additional *overlap parameter* α. Every time a new clustering is obtained, we check its overlap with the previously discovered clusterings using the ARI. If the calculated value is bigger than α, the clustering is rejected.

4 Experiments and results

We devise the experiments to answer the following questions:

(Q1) *Are latent features created by CUR^2LED interpretable and do they capture sensible information?*

(Q2) *Do latent features that result in models of lower complexity and/or improved performance exhibit a lower label entropy compared to the original data representation?*

(Q3) *Are latent representation that improve the performance of a model sparser than the original data representations?*

(Q4) *To which extent are latent features redundant?*

4.1 Datasets and setup

The results in [8] can be divided in three categories. The first category contains the IMDB and UWCSE datasets. These datasets present easy relational learning tasks in which the original data representation is sufficient for almost perfect performance. The main benefit of latent representations for these tasks was the reduction of model complexity. The second category includes the TerroristAttack dataset. The main benefit of latent representation here was the reduction of complexity, but not the performance. The third category involves the Hepatitis, Mutagenesis and WebKB datasets. These tasks benefited from latent representations in both performance and reduced model complexity. That is especially true for the Hepatitis and WebKB datasets on which the performance was improved by a large margin.

We take a representative task from each of the categories. Precisely, we use IMDB, UWCSE, Hepatitis and TerroristAttack datasets in our experiments. Both IMDB and UWCSE datasets were included as they are easy to understand without the domain knowledge, and thus useful for analysing the interpretability of relational latent features. As for the parameters of latent representation, we take the best parameters on individual datasets selected by the model selection procedure in [8]. When analysing the interpretability, we set θ to 0.3.

When evaluating the redundancy, we create latent representations by setting the α to the following values: $\{0.9, 0.8, 0.7, 0.6, 0.5\}$. We then learn a relational decision tree TILDE [12] on the obtained representation and compare accuracies, the number of created features and the number of facts.

4.2 Interpretability

To illustrate the interpretability of relational features, we show examples of latent features created for different datasets. We show that the relational decision trees learned on both original and latent representations. The explanations of latent features are provided as well.

Figure 3 shows the decision trees learned on the IMDB dataset. The task is to distinguish between actors and directors – this is a simple relational learning task and both original and latent decision tree achieve the perfect performance with only a single node. Even though latent representation does not seem beneficial in this particular case, it is interesting to see that the selected latent feature captures the same information as the decision tree learned on the original data –
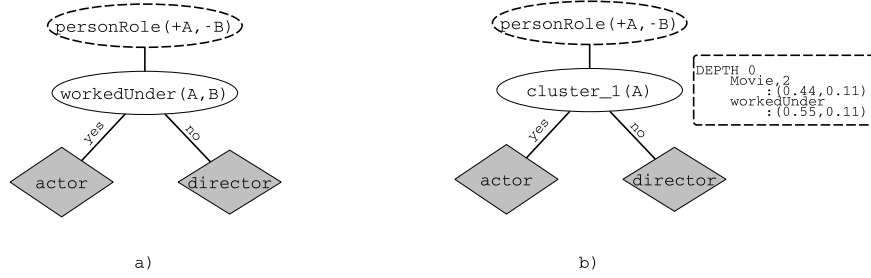
Fig. 3. Relational decision trees learned on the original (left) and latent (right) data representation of the IMDB dataset. The dashed ellipse indicates the target predicate and its arguments. The first argument, marked `A` and declared as input (`+`), denotes a person. The second argument, marked `B` and declared as output (`-`), states the label of the instance given by `A`. The values in the leaves of the decision trees are assignments to `B`. The dashed rectangle describes the latent feature – for each level of the *mean neighbourhood tree*, θ-confident elements are listed with the mean and standard deviation.

person instances in `cluster_1` are the ones that have a relationship with `movie` instances, and have worked under another person (a director).

Figure 4 shows the decision trees for the UWCSE dataset, which benefit from the latent features. Despite the simplicity of distinguishing students from professors, the decision tree learned on the latent features is more compact and has only a single node whereas the decision tree learned on the original features consists of three nodes. The latent feature here again captures similar knowledge as the original decision tree but expressed in a simpler manner – professor is someone who either has a position at the faculty, or is connected to people who are currently in a certain phase of a study program and have been in the program for a certain number of years.

What is particularly interesting about the examples above is that, even though the latent features are created in an unsupervised manner, they match the provided label very well. Moreover, they seem to almost perfectly capture the labelled information as only a few features are needed to outperform the decision tree learned on the original data representation. This observation shows that CUR^{2}LED is indeed capturing sensible knowledge in the latent space.

Both aforementioned examples are easy to understand and interpret without an extensive domain knowledge. The other tasks that have benefited more from the latent features are substantially more difficult to understand. For instance, the latent features created from the Mutagenesis dataset reduce the complexity of the relational decision tree from 27 to only 3 nodse, while improving the accuracy for 4 %. Similarly, on the Hepatitis dataset the latent features reduced the complexity of a decision tree from 22 nodes down to 5, improving the accuracy for 11 %. Because these examples require an extensive knowledge to interpreted them, we leave them out from this work.
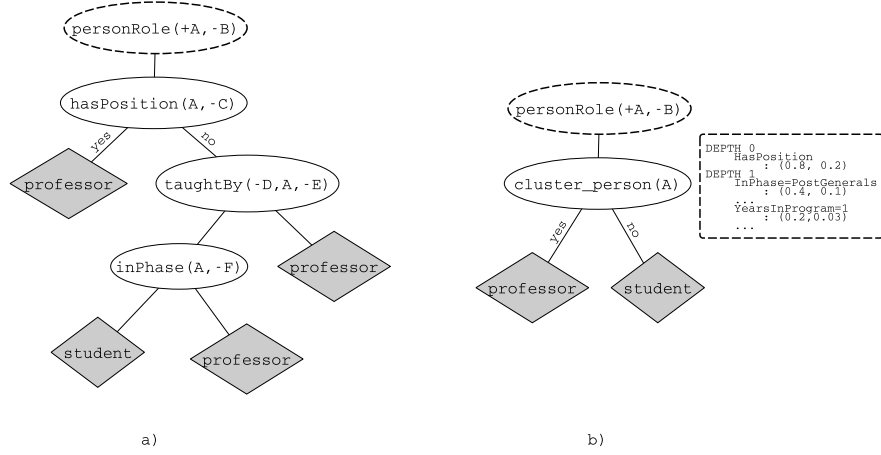
Fig. 4. Relational decision trees learned on the original (left) and latent (right) representations of the UWCSE dataset. The elements have the same meanings as in Figure 3.
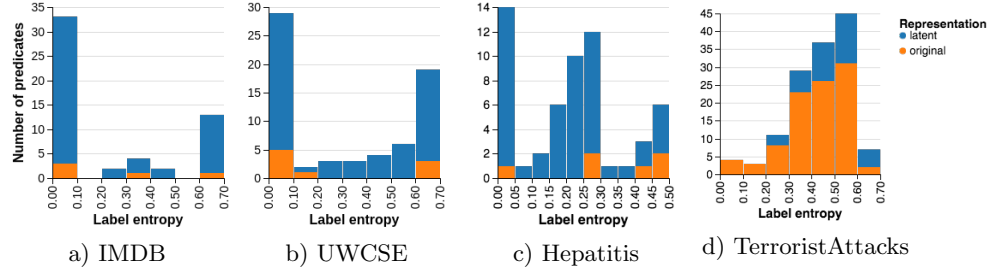
Fig. 5. The histogram of label entropy values for latent and original representations (indicated by the colour).

4.3 Properties of latent spaces

Label entropy. Figure 5 summarizes the label entropy for each dataset. In all cases where representation learning proved helpful (i.e., IMDB, UWCSE, Hepatitis), latent representations have a substantially larger number of predicates with low label entropy compared to the original data representation. The latent representation for the TerroristAttack datasets, however, shows a different behaviour in which latent features with high entropy dominate the representation. These results agree with the expectation that a high number of low entropy features makes learning easier. However, not all latent features have low label entropy. This is expected, as the labels are not considered during learning of latent features. It also does not pose a problem – these latent features are less consistent with the provided labels, but it easily might be the case that those features are useful for a different task.
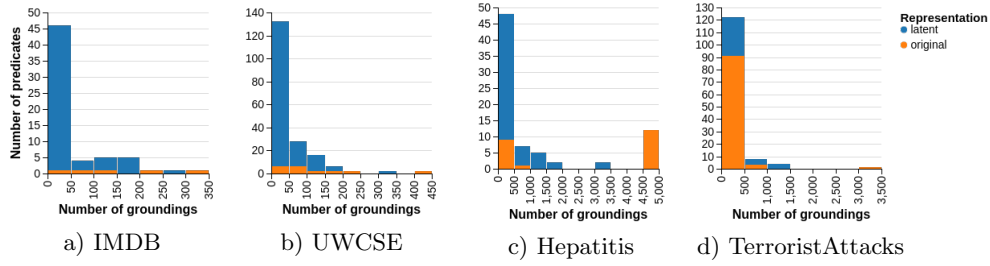
Fig. 6. Histogram of the number of true groundings of predicate for latent on original representations (indicated by the colour).

Sparsity. Figure 6 summarizes the number of groundings, i.e., the sparsity. The distribution of the number of true groundings in the latent representations (where latent features are beneficial) is heavily skewed towards a small number of groundings, in contrast with the original representation. That is especially the case in the Hepatitis dataset, which profits the most from the latent features. The exception to this behaviour is again the TerroristAttack dataset in which the original representation already is very sparse. These results indicates that latent features indeed describe smaller groups of instances and their local properties, instead of global properties of all instances.

Connecting label entropy and sparsity. A potential explanation of the above discussed results might be that many latent features capture a very small number of instances (e.g., 1 or 2) which consequently leads to a large number of features with low label entropy. Such features would largely be useless as they make generalization very difficult. To verify that this is not the case, Figure 7 plots the label entropy versus the number of groundings of a predicate. If latent features of low label entropy would indeed capture only a small number of instances, many points would be condensed in the bottom left corner of the plot. However, that is not the case – many latent predicates with low label entropy actually have a number of groundings comparable to the predicates in the original representation. The exception to this is again the TerroristAttacks dataset.

These results jointly point to the following conclusion: *latent features successfully identify local regions in the instance space that match well with the provided labels.* As a consequence, these local regions are easier to capture and represent.

Redundancy. Figure 8 summarizes the influence of α on the accuracy and the number of latent features. These results show that the performance of the classifier is not affected by removing features based on the overlap of clusterings they define. The performance of TILDE remains approximately the same, whereas the number of latent features is reduced by 20 to 30 %. As the number of features is directly related to the size of the search space of relational model (and thus the complexity of learning), this is an encouraging result indicating that the size of the search space can be naively reduced without sacrificing the performance.
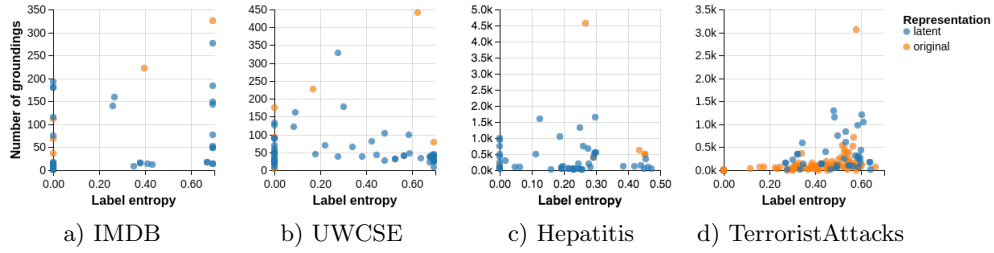
Fig. 7. Contrasting the label entropy of predicates (horizontal axis) and the number of true groundings (vertical axis). Whether a predicate comes from the original or latent representation is indicated by the colour.
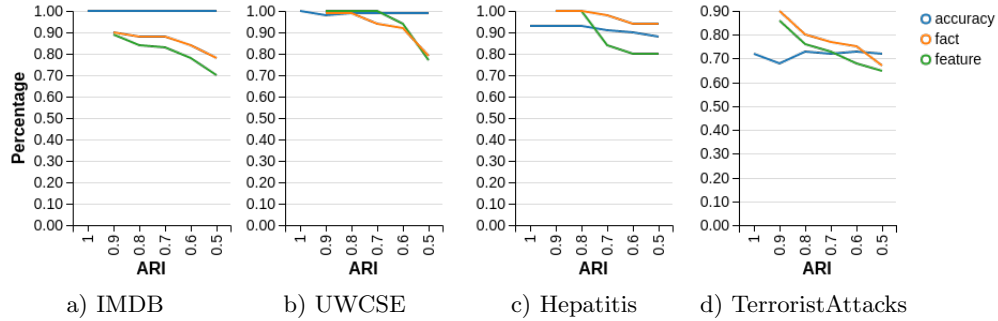
Fig. 8. Redundancy of features in latent representations. The accuracy (blue line), the number of latent features (green line) and the number of facts (= a sum of true groundings of all predicates, orange line) are reported for varying values of α. For the accuracy, the percentage of correctly classified examples is reported. For the number of features and facts, we report the ratio between the number of features/facts in the latent representation obtained with a specific value for α and the number of features/facts in the latent representation with $\alpha = 1.0$.

5 Conclusion

In this work we closely inspect the properties of latent representations for relational data. We focus on relational latent representations created by clustering both instances and relations among them, introduced by CUR^{2}LED [8]. The first property we analyse is the interpretability of latent features. We introduce a simple method to explain the meaning of latent features, and show that they capture interesting and sensible properties. Second, we identify two properties of these latent representation that partially explain their usefulness – namely, the label entropy and sparsity. Using these two properties, we show that obtained latent features identify local regions in instance space that match well with the labels. Consequently, this explains why predictive model learned from latent features are less complex and often perform better than the model learned from the original features. Third, we show that that latent features tend to be redundant,

and that 20 to 30 % of latent features can be discarded without sacrificing the performance of the classifier. This consequently reduces the search space for the relational models, and simplifies learning.

References

1. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.
2. Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.
3. Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.
4. Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *JOURNAL OF LOGIC PROGRAMMING*, 19(20):629–679, 1994.
5. Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In Wolfram Burgard and Dan Roth, editors, *AAAI*. AAAI Press, 2011.
6. Maximilian Nickel, Volker Tresp, and Hans peter Kriegel. A three-way model for collective learning on multi-relational data. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 809–816, New York, NY, USA, 2011. ACM.
7. Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc., 2013.
8. Sebastijan Dumančić and Hendrik Blockeel. Clustering-based relational unsupervised representation learning with an explicit distributed representation. In *International Joint Conference on Artificial Intelligence*, 2017.
9. Sebastijan Dumančić and Hendrik Blockeel. An expressive dissimilarity measure for relational clustering using neighbourhood trees. *Machine Learning*, 2017.
10. Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, August 2013.
11. Leslie C. Morey and Alan Agresti. The measurement of classification agreement: An adjustment to the rand statistic for chance agreement. *Educational and Psychological Measurement*, 44(1):33–37, 1984.
12. Hendrik Blockeel and Luc De Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1–2):285 – 297, 1998.