Embedding Formal Contexts Using Unordered Composition

Esteban Marquer, Ajinkya Kulkarni, Miguel Couceiro

Embedding Formal Contexts Using Unordered Composition[⋆]

Esteban Marquer, Ajinkya Kulkarni, and Miguel Couceiro

Université de Lorraine, CNRS, Inria N.G.E., LORIA, F-54000 Nancy, France
{esteban.marquer,ajinkya.kulkarni,miguel.couceiro}@inria.fr

Abstract. Despite their simplicity, formal contexts possess a complex latent structure that can be exploited by formal context analysis (FCA). In this paper, we address the problem of representing formal contexts using neural embeddings in order to facilitate knowledge discovery tasks. We propose *Bag of Attributes* (BoA), a dataset agnostic approach to capture the latent structure of formal contexts into embeddings. Our approach exploits the relation between objects and attributes to generate representations in the same embedding space. Our preliminary experiments on attribute clustering on the SPECT heart dataset, and on co-authorship prediction on the ICFCA dataset, show the feasibility of BoA with promising results.

Keywords: Formal Concept Analysis · Vector Space Embedding · Neural Networks · Complex Data · Link Prediction · Clustering.

1 Introduction

In recent years there has been an increasing interest in approaches to combine formal knowledge and artificial neural networks (NN). As mentioned in [1, p. 6], these approaches have a "generally hierarchical organization", with the "lowest-level network [taking] raw data as input and [producing] a model of the dataset". These networks represent data as real-valued vectors called *embeddings*.

Formal concept analysis (FCA) is another powerful tool for understanding complex data. Replicating its mechanisms using NNs could help processing complex and large datasets [5,17] by tackling FCA's scalability issues. Following this idea, we want to reproduce the general extraction process of FCA with NN architectures. This asks for a general embedding framework for contexts capable of handling data of arbitrary dimensions while encoding much of the contextual information. To our knowledge, there are only a few approaches in this direction [5,10,17,20]. Dürrschnabel *et al.* [5] propose FCA2VEC to embed formal contexts by encoding FCA's closure operators. It has three main components: *attribute2vec* and *object2vec*, (both based on *word2vec* [19]), and *closure2vec* that relies on a distance between closures of sets of attributes. In fact, *closure2vec*

⋆ We would like to thank the Inria Project Lab (IPL) HyAIAI ("Hybrid Approaches for Interpretable AI") for funding the research internship of E. Marquer.

is based on [20] that showed how to encode closure operators with simple feed-forward NNs. An advantage of this framework is that it produces embeddings of low dimensions (2 and 3) to facilitate interpretability. Yet FCA2VEC has several limitations: the embedding models need to be trained on each formal context (without guaranties of generalization) and the embeddings for objects and attributes are not defined in the same embedding space, which can be problematic when processing objects and attributes together.

To overcome these limitations, we propose *Bag of Attributes* (BoA) for providing a shared embedding space for objects and attributes by composing object embeddings from attribute embeddings. It is based on unordered composition and *long short-term memory neural network* (LSTM). Also, it predicts the number of concepts and a new measure of attribute similarity, called *co-intent similarity*, based on metric learning. This novel approach differs from the existing ones as it is agnostic to the data. Indeed, the model can accommodate any number of objects and attributes and it generalizes on real-world formal contexts despite being trained on randomly generated ones. We also explore the advantages and limits of our approach and provide experimental results on attribute clustering on the SPECT heart dataset[1], and on co-authorship prediction on the ICFCA dataset[2]. The comparison of BoA with FCA2VEC shows competitive performances on both tasks.

The paper is organized as follows. In Section 2, we briefly recall some basic background on FCA and NN. The architecture of BoA is defined in Section 3, whereas the corresponding training process and datasets are presented in Section 4. We discuss the impact of datasets characteristics on the performance in Section 5. Section 6 describes our experiments on attribute clustering and co-authorship prediction using real-world datasets[3]. Finally, we discuss potential developments for future work in Section 7.

2 Preliminaries And Basic Background

In this section we briefly recall some basic background on FCA (Subsection 2.1) and deep learning (Subsection 2.2 and 2.3), and define the new *co-intent similarity* for attributes. For further details on FCA see, *e.g.*, [8,12], and on NN architectures see, *e.g.*, [11].

2.1 Formal Concept Analysis

A *formal context* is a triple $\langle A, O, \mathbf{I} \rangle$, where A is a finite set of attributes, O is a finite set of objects, and $\mathbf{I} \subseteq A \times O$ is an incidence relation between A and O. A formal context can be represented by binary table C with objects as rows C_o and

[1] `https://archive.ics.uci.edu/ml/datasets/SPECT+Heart`

[2] `https://github.com/tomhanika/conexp-clj/tree/dev/testing-data/icfca_community`

[3] Compared to graph neural network approaches, FCA2VEC shows an improvement of at least 5% on all metrics for link prediction.

attributes as columns C_a, for $o \in O$ and $a \in A$. The entry of C corresponding to o and a is defined by $C_{o,a} = 1$ if $(o, a) \in \mathbf{I}$, and 0 otherwise.

It is well known [8] that every formal context $\langle A, O, \mathbf{I} \rangle$ induces a Galois connection between objects and attributes: for $X \subseteq O$ and $Y \subseteq A$, defined by: $X' = \{y \in A \mid (x, y) \in \mathbf{I} \text{ for all } x \in X\}$ and $Y' = \{x \in O \mid (x, y) \in \mathbf{I} \text{ for all } y \in Y\}$. A *formal concept* is then a pair (X, Y) such that $X' = Y$ and $Y' = X$, called respectively the *intent* and the *extent*. It should be noticed that both X and Y are closed sets, i.e., $X = X''$ and $Y = Y''$. The set of all formal concepts can be ordered by inclusion of the extents or, dually, by the reversed inclusion of the intents. We denote by $I \subseteq 2^A$ the set of intents and $E \subseteq 2^O$ the set of extents.

2.2 Auto-Encoders, Embeddings And Metric Learning

Auto-encoders are a class of deep learning models composed of *(i)* an encoder, that takes some x as an input and produces a latent representation z, and *(ii)* a decoder, that takes z as an input and reconstructs $\hat{x}$ a prediction of x. The model learns to compress x into z by training the the model to match x and $\hat{x}$. The training objective matching x and $\hat{x}$ is the *reconstruction loss*. Auto-encoders are one of the methods to generate representation of data as vectors. In that case z is called the *embedding* of x, and the real-valued space in which z is defined is called the *embedding space*.

Unlike traditional auto-encoders, *variational auto-encoders* (VAEs) [14] encode a distribution for each value of z instead of the value itself. In practice, for each component of z, the encoder produces two values: a mean μ and a standard deviation σ. When training the model, z is sampled from the normal distribution defined by μ and σ. Finally, the distribution defined by μ and σ is normalized by adding the *Kullback–Leibler (KL) divergence* loss term. To make this process differentiable and be able to train the model, a method called *reparametrization* [14] is used. VAEs are known to provide better generalization capabilities and are easier to use to decode arbitrary embeddings, compared to classic auto-encoders. This property is useful for generation since we can train a model generating embeddings and then decode them with a pre-trained VAE. In fact, VAEs are used in a wide variety of applications to improve the quality of embedding spaces, *e.g.*, for image [14], for speech [16] and for graph generation [15].

Metric learning [18] is a training process used ensure that embedding spaces have the properties of metric spaces. To achieve this, a loss is used to reduce the distance between the embeddings of equal elements and increase the distance between embeddings of different elements. Multiple losses can achieve this, such as the *pairwise loss* and the *triplet loss*. Triplet loss considers the embeddings of three elements: an input x_1, some x_2 judged equal to x_1 and some y different from x_1. In some approaches [16] a predictor (typically a *multi-layer perceptron* (MLP)) is used to predict a distance between the embeddings instead of applying a standard distance directly on the embeddings. It is possible to learn distances on different properties of the embedded elements, by splitting the embedding into segments and learn a different distance on each one [16]. Metric learning is usually used to approximate actual distances. However, this process can be applied to learn other kinds of measures not fitting the definition of a distance, which is what we do in this paper.

We need both "equal" and "different" attributes to use metric learning losses on attribute embeddings. Nonetheless, even if we consider equivalent attributes (*i.e.* with the same extent) as "equal", they are usually rare within a given context. We define co-intent similarity to compare attributes and avoid this issue. Given two attributes a_1 and a_2 we define their *pairwise co-intent similarity* as:

$$\text{co-intent}(a_1, a_2) = \begin{cases} 1 \quad \text{if } |\{i \in I | a_1 \in i\}| + |\{i \in I | a_2 \in i\}| = 0 \\ \dfrac{2 \times |\{i \in I | a_1 \in i, a_2 \in i\}|}{|\{i \in I | a_1 \in i\}| + |\{i \in I | a_2 \in i\}|} \text{ otherwise.} \end{cases} \tag{1}$$

In other words, it is the ratio of intents containing both attributes over the intents containing a_1 or a_2[4]. In cases where no intent contain the attributes (both attributes are empty or padding columns), the similarity is set to 1. Co-intent similarity ranges from 0, for attributes never appearing in the same intents, to 1, for attributes always appearing together or for identical attributes.

2.3 Unordered Composition

By *unordered composition functions* we mean operations that do not take into account the order of the input elements and that can accommodate any number of input elements. Typical examples are the componentwise min, max, and average (also called respectively min-, max-, and average-pooling). Unordered composition-based models have proven their effectiveness in a variety of tasks, for instance, sentence embedding [13], sentiment classification [4] and feature classification [9]. On the one hand, this family of methods allows inputs of varying sizes to be processed at a relatively low computational cost, by opposition to recurrent models like LSTM [11]. On the other hand, we lose the information related to the order of the input elements.

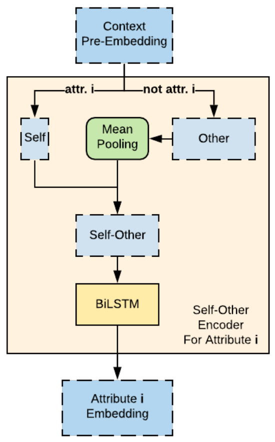
3 Proposed approach

In this section we first define the proposed approach and explain the objectives used to train the model. The training process is detailed in Section 4.

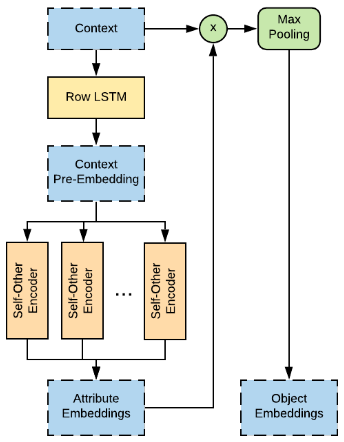
3.1 Bag of Attributes

Bag of Attributes (BoA) takes a formal context as input, and produces embeddings for its attributes. Then, object embeddings are computed using the embeddings of the attributes and the formal context. BoA has four main components: a pre-embedding generator, an attribute encoder called *self-other encoder* to compute the attribute embedding, an object encoder and a decoder. It considers the

[4] Observe that this is essentially the Jaccard index on the set of intents.



(a) Self-other attribute encoder for an attribute.



(b) BoA encoder architecture.

Fig. 1: Schematic representation of the BoA architecture. Blue blocks correspond to tensors, the orange to neural components and green blocks to non-neural computations. Arrows joining blocks represent concatenation of tensors.

attributes as an unordered set to produce the object and attribute embeddings. The name is inspired by *Bag of Words* (BoW) [19]. The structure of the encoder is schematized in Figure 1. The decoder itself is an MLP predicting if an object has an attribute or not (1 or 0, respectively). Its input is the concatenation of the object and the attribute embeddings. A sigmoid function applied on the output ensures it is in $[0, 1]$. BoA is trained as a VAE on formal contexts, so a μ and σ vector is produced for each attribute. The sampling of the attribute embeddings is done before the generation of object embeddings.

The order of the attributes in the dataset does not matter for FCA, therefore in BoA each attribute is processed in a similar manner to capture this property. Each attribute is compared to all the other attributes, for each object of the dataset. In practice, the column of an attribute (*self*) is compared to an unordered composition (average-pooling) of all the other attributes (*other*). *Self* and *other* are then processed by a *bidirectional LSTM* (BLSTM) [11], with the object dimension as the sequence dimension. The last hidden state of the BLSTM is processed with a feed-forward layer into an embedding that represents the attribute. The structure of the attribute encoder is presented on Figure 1a. Finally, the object embeddings are computed by applying max-pooling on the embeddings of the attributes present in the object's intent. We apply a LSTM on each row of C before the *self-other encoder*, as it produces different embeddings for each attribute despite the same input, to allow the model to determine which attribute is involved by avoiding the use unordered composition directly on C.

3.2 Training Objective

We train BoA using KL divergence on the attribute embeddings exclusively as the sampling happens before the computation of the object embeddings. We use the *binary cross entropy* loss for reconstruction because the model predicts between two classes (1 and 0). On top of that, we use metric learning with the new *co-intent similarity* (see Equation 1) and the number of concept [8], with *mean square error* (MSE) as the loss function. Predicting the number of concepts from the context without actually computing the intents, helps when generating the set of concepts using neural models. Indeed, knowing how many elements to generate beforehand facilitates the generation process. We use MLPs to predict the co-intent similarity and number of concepts. For co-intent similarity between two attributes a_1 and a_2, the input is the concatenated embeddings of a_1 and a_2 and a sigmoid output function is added to ensure the predicted similarity is in $[0, 1]$. We apply a max-pooling over the attribute embeddings before predicting the number of concepts, which corresponds to a *deep averaging network* (DAN) [13].

4 Training Setup

In this section we present our training process (Subsection 4.1), dataset (Subsection 4.2) and we describe the data augmentation process (Subsection 4.3).

4.1 Training Process

We train BoA in two phases of 5000 epochs each. In the first phase, we apply the reconstruction loss and the KL divergence only. Then, we gradually introduce the prediction of the co-intent similarity and of the number of concepts. When using metric learning with multiple distances, a common approach is to split the embedding space and to learn one distance per sub-part of the embedding space [16]. We apply the same principle and use 50% of the embedding space to predict the co-intent similarity and 25% for the number of concepts. The exact embedding dimension of BoA is 128, with a pre-embedding of size of 64. The LSTM and the BLSTM have two layers each. The decoder MLP has four layers, and the MLPs used for distance prediction both have two layers. We use a *rectified linear unit* activation function between all the layers of the model.

4.2 Training Data

The dataset used for training the BoA model is composed of 6000 randomly generated formal contexts split into training and validation, and the corresponding intents computed using the Coron system[5]. To generate a context of $|O|$ objects and $|A|$ attributes we sample $|O| \times |A|$ values from a Poisson distribution and apply a threshold of 0.3. Values under the threshold correspond to 1 in the context, which leads to a density around 0.3. Note that the random generation

[5] `http://coron.loria.fr/site/index.php`

Table 1: Descriptive statistics on the dataset of randomly generated contexts.

	Dataset	# Object	# Attribute	# Concept	Context density
Mean	train	12.83 ± 6.11	12.98 ± 6.03	77.93 ± 78.39	0.329 ± 0.057
± std.	test	12.83 ± 6.13	12.97 ± 6.04	78.12 ± 77.27	0.332 ± 0.057
Range	train	1 to 20	2 to 20	1 to 401	0 to 0.56
	test	2 to 20	3 to 20	2 to 401	0 to 0.49

process may result in empty rows and columns, which will be dropped by Coron if they are at the extremities of the context. For this reason, the actual size of the generated context may be smaller than the requested one. We generate a training set of 5000 contexts and a test set of 1000 samples. For the training phase, a development set of 10% of the training set is randomly sampled from the training set. For each set, we generate different sizes of contexts, 20% of each: 5×5, 10×10, 10×20, 20×10 and 20×20 contexts ($|O| \times |A|$). The statistics of the generated datasets are reported in Table 1.

4.3 Data Augmentation

We rely on plain random generation for the formal contexts, and not on more involved generation processes as discussed in [7,6], so the random data is biased. We introduce a simple way to compensate some of those biases while improving the generalization capability of the model. We implement the following data augmentation pipeline: *(i)* duplicating of objects and attributes, *(ii)* inverting the value of entries and *(iii)* shuffling objects and attributes. With this process, we simulate identical (duplication) and nearly identical (duplication + drop) objects or attributes that appear in real-world datasets.

Objects and attributes have a probability p of being duplicated. If duplicated, they have the same probability p of being duplicated again. From this definition, the number of copies of an object (or attribute) follow a geometric law with a probability of success p. Consequently, the exact number of objects and attributes actually seen during training do not match the ones reported in Table 1. Nonetheless, the duplication follows a geometric law so we can estimate the number amount of object and attribute seen as $number/(1-p)$. Inverting some randomly selected values in the formal context is our adaptation of dropout, a common technique in deep learning. The shuffling after duplication avoids model's reliance on order of the objects and the attributes. We set the duplication probability to $p = 0.1$ and the drop probability to 0.01. In this setting, the estimated average object and attribute numbers are respectively 14.25 and 14.42, for both the training and development sets.

When co-intent similarity is used, duplication and shuffling are reproduced on the intents. However, drops in a formal context alter the corresponding lattice, so they are not applied at all when using data from the lattice. This precaution avoids making the model insensitive to small variations in the input.

4.4 Issues With KL Divergence

When adding the KL divergence to the prototype of BoA (initially a simple auto-encoder) the performance of the model was greatly impaired. The analysis of the predictions revealed the model was going for "low hanging fruits" and ignored the embeddings themselves, as described in [3]. To solve this issue we apply *annealing* [3] and multiply the KL divergence by a lambda that we set to 10^{-3}. This reduces the impact of the KL divergence on the training and allows the model to learn some features before the KL divergence comes into effect. However, it reduces the benefits we get from using a VAE.

5 Exploring The Limits Of BoA

We now explore the limits of BoA *w.r.t.* input data. All experiments are performed on randomly generated data to control of the evaluation process.

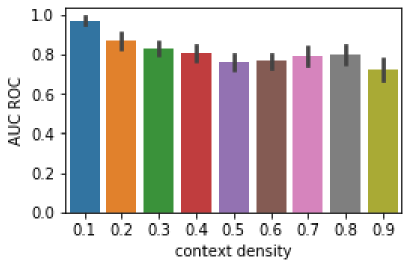
5.1 Reconstruction Performance

To assess the reconstruction performance of the BoA auto-encoder, we use the *area under the ROC curve* (AUC ROC). It allows to determine whether the BoA has good predictive capacity and, similarly to the F1 measure, gives a general account of performance. To determine if the results are significantly different, we use Student t-test on means. The results are presented in Figure 2.

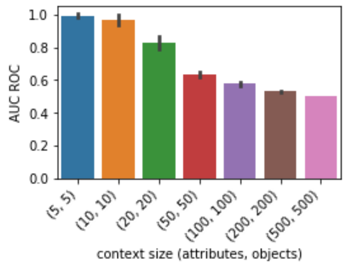
We first evaluate the impact of the density on the reconstruction by comparing the performance on random contexts with densities from 0.1 to 0.9. We use 100 samples per density with a fixed size of 20 objects and attributes. Student's t-test show significant differences between the performance with the various densities: all the p-values are under 0.01 except between 0.4 and 0.8 (0.24), 0.5 and 0.6 (0.39), and 0.7 and 0.8 (0.19). However, the model performance stays overall stable across the densities, while slightly better with smaller densities. We suspect this tendency is due to the composition process of the object embeddings: the higher the density, the more attributes are present for an object, so more attribute embeddings are involved in the composition of the object embeddings, making it more complex to decode.

We also examine the effect of the size on the AUC ROC. Square random contexts ($|O| = |A|$) of sizes in $\{5, 10, 20, 50, 100, 200, 500\}$ and a fixed density of 0.3 are used for this experiment, with 20 samples per size. The model performs very well for seen data sizes with a slight drop to 0.83 for 20 objects and attributes. As expected, the performance drops when manipulating larger contexts. For 50 objects and attributes (2.5 times the maximum seen size), the AUC ROC is above 0.63, but from 100 objects and attributes onward, it drops under 0.6. Finally, with 500 objects and attributes (25 times the largest seen data size and 4 times the embedding size), the reconstruction AUC ROC falls to 0.50 in average. This is the limit of reconstruction performance with the current training process.

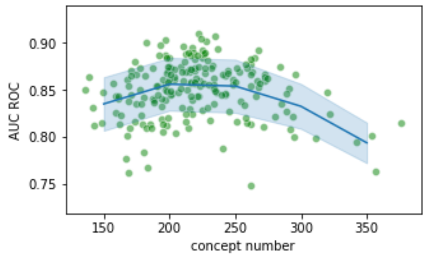
Finally, we examine the impact of the number of concepts on the performance of the model. We use contexts of fixed size (20 objects and attributes) from the



(a) Impact of the density, from 0.1 to 0.9, 100 samples per density.



(b) Impact of the size, from 5 to 500 objects and attributes, 20 samples per size.



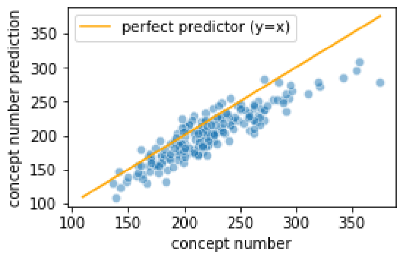
(c) Impact of the concept number, for 200 sample with 20 objects and attributes. The blue line is the general tendency when rounding the concept number to 50.

Fig. 2: Reconstruction performance on random contexts. The error bars and the shaded area correspond to the standard deviation.

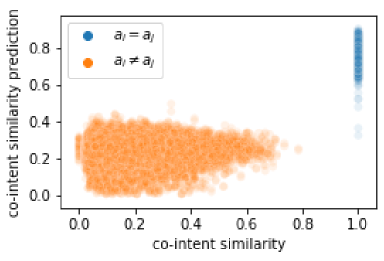
test set, totaling 200 contexts. We consider the concept number as an indicator of the variety of attributes and objects in the context. Indeed, if the concept number is high for a given size of context, we can expect the context to be close to the clarified context (context with no equivalent objects or attributes). This implies a lower amount of duplicate objects and attributes. In addition, we can expect the model to have a harder time encoding and decoding irregular contexts than repetitive ones. Consequently, the drop of the AUC ROC for higher concept numbers is not surprising. However, we also observe a lower performance around 150 concepts. This second decrease requires further investigation.

5.2 Metric Learning Performance

We evaluate the performance of BoA for the co-intent similarity and number of concepts' prediction, by computing the attribute embeddings and applying the predictors trained together with the BoA model. We use the 200 contexts of 20 objects and attributes from the test set. The prediction results are reported Figure 3. The Pearson correlation coefficient is 0.9 between the actual concept number and the prediction, indicating a strong correlation. We can notice the tendency of the model to under-evaluate the concept number. Even though BoA



(a) Concept number.



(b) Co-intent similarity.

Fig. 3: Predicted pseudo-metrics against the actual values, for the 200 samples with 20 objects and attributes from the test set.

manages to differentiate a_i and a_j, when $a_i = a_j$, the predictions in the other cases are not clear: for similarities between 0 and 0.8, they seem randomly picked between 0 and 0.4. The analysis of the training process reveals a small difference of the MSE between the first and the last training epochs: from 0.17 to 0.05.

6 Experiments On Real-World Datasets[6]

To evaluate the performance of BoA on real-world datasets, we follow the empirical setting of [5]: we reproduce their link prediction and attribute clustering tasks, used to evaluate object2vec (o2v) and attribute2vec (a2v), respectively. We use the same ICFCA dataset as [5] for link prediction. For attribute clustering however, we use SPECT heart[7] as it is smaller than *wiki44k* [5], with dimensions closer to the training data: 68 objects and 23 attributes. We train the CBoW and SG variants of FCA2VEC models using the same settings as in [5], with 20 random iterations of each model and an embedding size of 3. To obtain comparable results, we reduce the embeddings produced by BoA to 3 dimensions by applying two standard dimensionality reduction techniques: *principal component analysis* (PCA) and *t-distributed stochastic neighbor embedding* (TSNE). We use Student t-test on means to determine if the results are significant.

We report the link prediction performance in Table 2. The three BoA variants show a significantly different performance from o2v SG, with all the p-values lower than 0.005. We found that the classifier based on BoA, the one with the best F1 score, systematically answers positive. Additionally, we fail to reproduce the performance of [5] (F1 score of 0.69 for o2v CBoW, 0.66 for o2v SG) Finally, the ICFCA context is very sparse: it has a density of 0.003 on the train and 0.005 on the test set. Due to this, the task may not be representative of the

[6] Experiments were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see `https://www.grid5000.fr`).

[7] `https://archive.ics.uci.edu/ml/datasets/SPECT+Heart`

Table 2: Performance on the link prediction task (mean ± std.).

Model	Precision	Recall	F1
o2v CBoW	0.63 ± 0.05	0.46 ± 0.05	0.53 ± 0.05
o2v SG	0.70 ± 0.04	0.49 ± 0.03	0.57 ± 0.02
BoA PCA 3d	0.65	0.42	0.51
BoA TSNE 3d	0.58	0.67	0.62
BoA	0.50	1.00	0.67

Table 3: Performance on the attribute clustering task with 2, 5 and 10 clusters.

Model	k = 2	k = 5	k = 10
a2v CBoW	0.66 ± 0.00	0.14 ± 0.02	0.063 ± 0.013
a2v SG	0.35 ± 0.13	0.11 ± 0.03	0.042 ± 0.010
BoA TSNE 3d	0.30	0.29	0.044
BoA PCA 3d	0.70	0.22	0.051
BoA	0.70	0.22	0.051

performance of the object embeddings on most datasets. These results hint that the task needs to be adapted to get proper insight on the object embedding performance. The attribute clustering performance is reported in Table 3. In this experiment, we find that the CBoW variant performs significantly better than the SG (all t-test p-values under 0.0005). This is the opposite of the result found by [5] for attribute clustering. However, this result may be due to using a different dataset. Interestingly, the BoA PCA variant performs equally to the full BoA. The performance of BoA (and BoA PCA) is significantly better than a2v CBoW for 2 and 5 clusters (p-values under 10^{-14}). For 10 clusters however, a2v CBoW performs significantly better (p-value under 0.001). The model improves the performance of a2v CBoW by 4% for 2 clusters and 8% for 5 clusters.

7 Conclusion And Future Work

We introduced the *co-intent similarity* for attributes and proposed BoA, a generalized embedding framework for formal contexts that integrates several FCA aspects. Our framework is data agnostic and scales to real-world datasets such as the SPECT heart dataset. It is also robust *w.r.t.* variations in the density and the concept number of formal contexts. The experimental results are encouraging, as our general approach achieves performance similar to FCA2VEC, a dataset specific one. In addition to being data agnostic, BoA constitutes a promising alternative to FCA2VEC since it uses a single embedding space for all contexts. Moreover, the asymptotic time complexity of embedding a formal context through BoA is $\theta(|O| \times |A|^2)$. In comparison, applying a linear embedding (the most basic embedding) or an LSTM embedding model (like our pre-embedding) to each entry has a complexity of $\theta(|O| \times |A|)$.

As future work we aim to tackle two active issues in the FCA community: the random generation of contexts and the scalability of concept lattices. Random context generation introduces several biases as it does not match the distribution of real-world formal contexts (see *e.g.*, the "stegosaurus effect" discussed in [2]). It could be beneficial to use more accurate generation algorithms than our current algorithm, like those discussed in [6,7]. Nonetheless, it is encouraging to see that BoA achieves acceptable performance when trained on simply generated contexts of relatively small size (up to 20 objects and attributes). Since this preliminary work enables the use of decoders in generation processes, using NNs seems a feasible direction for concept lattices construction. These are some directions of current ongoing work.

References

1. Besold, T.R., *et al.*: Neural-Symbolic Learning and Reasoning: A Survey and Interpretation. CoRR **1711.03902** (2017)
2. Borchmann, D., Hanika, T.: Some experimental results on randomly generating formal contexts. In: CLA. vol. 1624, pp. 57–69 (2016)
3. Bowman, S.R., *et al.*: Generating sentences from a continuous space pp. 10–21 (2016)
4. Chen, X., *et al.*: Adversarial deep averaging networks for cross-lingual sentiment classification. CoRR **1606.01614** (2016)
5. Dürrschnabel, D., *et al.*: Fca2vec: Embedding techniques for formal concept analysis. CoRR **1911.11496** (2019)
6. Felde, M., *et al.*: Formal context generation using dirichlet distributions. In: Graph-Based Representation and Reasoning. pp. 57–71. Cham (2019)
7. Ganter, B.: Random extents and random closure systems. In: CLA (2011)
8. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer (1999)
9. Gardner, A., *et al.*: Classifying unordered feature sets with convolutional deep averaging networks. CoRR **1709.03019** (2017)
10. Gaume, B.e.: Clustering bipartite graphs in terms of approximate formal concepts and sub-contexts. IJCIS **vol. 6**, 1125–1142 (2013)
11. Graves, A., *et al.*: Framewise phoneme classification with bidirectional lstm and other neural network architectures. Neural networks **18**(5-6), 602–610 (2005)
12. Ignatov, D.I.: Introduction to formal concept analysis and its applications in information retrieval and related fields. CoRR **1703.02819** (2017)
13. Iyyer, M., *et al.*: Deep unordered composition rivals syntactic methods for text classification. In: 53rd ACL and 7th IJCNLP. pp. 1681–1691 (2015)
14. Kingma, D., Welling, M.: auto-encoding variational bayes (2013)
15. Kipf, T.N., *et al.*: Variational Graph Auto-Encoders. CoRR **1611.07308** (2016)
16. Kulkarni, A., *et al.*: Deep Variational Metric Learning For Transfer Of Expressivity In Multispeaker Text To Speech (2020), to appear in SLSP 2020
17. Kuznetsov, S.O., *et al.*: On neural network architecture based on concept lattices. In: Foundations of Intelligent Systems. pp. 653–663. Cham (2017)
18. Lin, X., *et al.*: Deep variational metric learning. In: Computer Vision – ECCV. pp. 714–729. Cham (2018)
19. Mikolov, T., *et al.*: Efficient Estimation of Word Representations in Vector Space. pp. 1–12 (2013)
20. Rudolph, S.: Encoding closure operators into neural networks. In: NeSy. pp. 40–45. CEUR-WS. org (2007)