UNIVERSITEIT VAN AMSTERDAM

MSc Artificial Intelligence
Master Thesis

Knowledge Generation

by
Florian Wolf
12393339

December 2, 2020

48 Credits
April 2020 - December 2020

Supervisor:
Dr Peter Bloem
Chiara Spruijt

Asessor:
Dr Paul Groth

Contents

Abstract

We generate Knowledge! [1]

1 Introduction

Here comes a beutiful introduction. Promise!

1.1 Motivation

Computer vision reached a point, where semantics, entities and relations can be infered in an simple image. Would it not be fantastic to be able to apply this to text too? Could we train a model to learn the semantics of a KG?

1.2 Expected Contribution

This thesis is aimed to be at fundamental research and provide insight, into if further research in this direction would be meaningful.

1.3 Research Question

`How well can a VAE learn the underlying semantics of a KG?`

2 Related Work

This section presents previous work which inspired and layed the fundamentals for this thesis.

2.1 Graph VAE

Present different papers with graph VAEs

- Belli recurrent VAE
- GraphVAE paper
- some more

The model architecture presented in the GraphVAE paper is the starting point of our model.

2.2 RGCN

Present the relational graph convolution model paper by Kipf and maybe others

2.3 Embedding models

Present RASCAL and one or two more.

3 Background

In this section we will go over related work and relevant background information. The depth of the explanation is adopted to the expected prior knowledge of the reader. The reader is supposed to know the basics of machine learning, including probability theory and basic knowledge on neural networks and their different architectures.

3.1 The Graph VAE – one shot method

3.1.1 VAE

The VAE as first presented by [2] is an unsupervised generative model consisting of an encoder and a decoder. The architecture of the VAE differs from a common autoencoder by having a stochastic module between encoder and decoder. Instead of directly using the output of the encoder, a distribution of the latent space is predicted from which we sample the input to the decoder. The reparameterization trick allows the model to be differentiable. By places the sampling module outside the model we get a deterministic model which can be backpropagated.

MLP The Multi-Layer Perceptron (MLP) was one on the first machine learning models. In its basic structure it takes a one dimensional input, fully-connected hidden layer, activation function and finally output layer with normalized predictions. Images or higher dimensional tensors can be processed by flattening them to a one dimensional tensor and adjusting the input dimensions. This makes the MLP a flexible and easy to implement model.(reference)

3.1.2 Graph convolutions

CNNs have shown great results in the field of images classification and object detection. This is due to the fact that a convolution layer takes into account the relation of one pixel to its neighbors. The same holds for graph CNNs where at each convolution the information of each node is passed on as messages to all its neighbors. Each convolution applies an activation function in between the steps. In this case the information is the edges each node has. To process node attributes and edge atributes we have to look at more complex models [3].

3.1.3 RGCN

Realtional Graph Convolution Net (RGCN) was presented in [4] for edge prediction. This model takes into account features of nodes. Both the adjacency and the feature matrix are matrix-multiplied with the weight matrix and then with them-selves. The resulting vector is a classification of the nodes.

3.1.4 Graph VAE

Now we have all the building bocks for a Graph VAE. The encode can either be a MLP, a GCNN or an RGCN. The same holds for the decoder with the addition that model architechture needs to be inverted. An verison of a Graph VAE presented in [5]. This model combines both the previous methods. The input graph undergoes relational graph convolutions before it is flattened and projected into latent space. After applying the reparametrization trick, a simple MLP decoder is used to regenerate the graph. In addition the model concatenates the input with a target vector y, which represents ???. The same vector is concatenated with the latent tensor. ***Elborate why they do that***.

Graphs can be generated recursively or in an one-shot approach. This paper uses the second approach and generates the full graph in one go. ***Cite?***

3.2 Graph Matching

Intro to graph matching on sparse graphs.

3.2.1 Permutation Invariance

Permutation invariance refers to the invariance of a permutation of an object. An visual example is the the image generation of numbers. If the loss function of the model would not be permutation invariant, the generated image could show a perfect replica of the input number but due to positional permutation the loss function would penalize the model. OR: An example is in object detection in images. An object can have geometrical permutations such as translation, scale or rotation, none the less the model should be able to detect and classify it. In that case, the model is not limited by permutations and is there fore permutation invariant. In our case the object is a graph and the nodes can take different positions in the adjacency matrix. To detect similarities between graphs we apply graph matching.

3.2.2 Graph matching algorithms

These are three of the state of the art graph matching algorithms.

- Wasserstein
- Maxpooling
- one more

There are various graph matching algorithms. The one we will implement is the max-pooling (Finding Matches in a Haystack: A Max-Pooling Strategy for Graph Matching in the Presence of Outliers).

The max-pooling graph matching algorithm returns a symmetric affinity matrix for all nodes

The resulting similarity matrix gives us $X*$ which is continuous and therefore useless. To transform is to a discrete X we use the hungarian algorithm (GPU-accelerated Hungarian algorithms for the Linear Assignment Problem)

Hungarian algorithm

The hungarian algorithm is used to find the shortest path within a matrix. This could be the most efficient work-distribution in a cost matrix. Or ...

It consists of four steps of which the last two are repeated until convergence. This algorithm is not scalable. The Munks algorithm (reference) tackles this problem by?? and is scalable.

second option: Compare only graph structure. NX algorithms: Greedy, Shortest path ...

3.2.3 Graph Matching Loss

The loss is discribed in [5] as.

In contrast to his implementation we assume, that a node or edge can have none, one or multiple attributes. Therefore our attributes are also not sigmoided and do not sum up to one. This leads o the modification of term logF and logE where we do not matrix multiply over the attribute vector but take the BCE as over the rest of the points. KG can have multiple or no attributes vs molecular graphs can be one hot encoded.

Okay, further we need to treat the log_pE and log_pF just like log_pA and subtract the inverse. Otherwise the model learn to predict very high values only.

A note to the node features, these stay softmaxed and one-hot encoded since we will use them as node labels.

3.3 Knowledge Graphs

Knowledge Graphs are great! The best in the world. Knowledge graphs have very different formats. The datasets we will be sorking with are in rdf format. This format can include an defined onthology or not. This means the KG consists of triples subject, relation, object. when indexing these triples. we get a dense representation of the KG. About sparse KGs

4 Methodology

This section describes the methodology for this thesis. The first part includes the presentation of the model, the reprocessing of the input and the evaluation metrics. The second part describes the experimental setup and the different experimental runs. The work of this thesis has aimed to be fully reproducable, thus the code is opensourced and available on Github [1].

4.1 Knowledge graph representation

The first step in our pipeline is the representation of the KG in tensor format. In order to represent the graph structure we use an adjacency matrix A of shape $n \times n$ with n being the number of nodes in our graph. The edge attribute or directed relations between the nodes are represented in the matrix E of shape $n \times n \times d_E$ with d_E being the number of edge attributes. Similarly for node attributes we have the matrix F of shape $n \times d_N$ with d_N number of node attributes. The input graph can have less nodes than the maximum n but not more. The diagonal of the adjacency matrix is filled with 1 if the indexed node exists, and with 0 otherwise. The number and encoding of the attributes must be predefined and cannot be changed after training. This way we can uniquely represent a KG.

Graph embeddings? unsupervised approach

[1] ***Thesis Repo Link***

4.2 Graph VAE

hi
Convolution part
RCGN relation Convolution neural net
MLP encoder
Latent space
reparametrization trick
MLP decoder
Graph matching
Discretization of prediction

4.3 Loss function

If loss function should be permutation invariant we need to do some kind of graph matching.
Different options for graph matching.
Maxpooling-algorithm:
Assumptions
Node to edge affinity equals 0

Self-loops are possible, adjacency matrix can be zero or one.

Summing over the neighbors means summing over the whole column Normalize matrix with Frobenius Norm
Batch version: Only matmul and dot. keep dimension of S with shape (bs,n,n,k,k) When maxpooling, flatten Xs (n,k) for batch dot multiplication. This way (i think) we sum over all j nad b neighbors instead of taking the max.
Hungarian algorithm for discrimination of X

The hungarian algorithm as presented in section 2 return the shortest path in a matrix. We use this shortest path as bast match between the two graphs. The node paris identified as optimal are masked as 1 and the rest of the matrix as 0. This way we discretizice $X\star$ to X.
The equations

5 Experiments

This section presents the experiments we ran. We covered link and node prediction and compared those to SOTA scores. Further we ran experiments on investigating the coherence of the reproduced graph structure. Lastly we measured the adherence of our model to the KG's underlying syntax.

5.1 Datasets and Training

For this sake of meaningful results, we chose to use the earlier presented, two most popular dataset used in this field, FB15k237 and WN18rr.
Training models on each dataset for 333 epochs, without early stopping.

5.2 Link Prediction

We used link prediction as evaluation protocol and for comparison state of the art models. For each triple in the dataset, we remove the tail and combine it with all possible entities in our dataset. EVen though this is called 'triple-corruption', also correct triples can be generated, which could appear in the trainset. These have bo the filtered out before evaluation. Link prediction on unfiltered test data is termed 'raw'. Our model then computes the ELBO loss for all corrupted triples, which are sorted and stored in ascending order. The same procedure is repeated the triple's head in place of the tail.

The metrics used to evaluate the predictions, is the mean reciprocal rank (MRR) and hits at 1,3 and 10. The MRR ??? explain what the MRR is.
Hits at 1 indicates what percentage of the triples in the test set have been ranked the highest compared to their corruptions. Similar, hits at 3 and 10, give a percentage of triples ranked in the top 3 and top 10. These metrics allow a fair comparison on a dataset between models, regardless of the ranking method of each model.

5.3 Sanity Checks

Check if adj matrix adheres to edge attribute matrix.

5.4 Interpolate Latent Space

We tale two random triples and interpolate the latent space of these two triples. The interpolations result in: HERE AN EXAMPLE.

Further we go ahead and test what happens if we modify one latent dimension of a triple. HERE AN EXAPMPLE. Can the model assign logical features to latent dimensions?

5.5 Subgraph Generation

Until now our model trained on only one triple per sparse graph. What will happen if we train it on more than one triples?

5.6 Syntax coherence

Check if generated triples follow basic logic.

- Generate triples by random signals
- Filter these triples on a certain relation
- Check if the entities are part of the linked class

Since there is no preset for how to check the semantics of a KG, we will use simple basic logical criteria. The generated triples are filtered for the relation 'is capital of', thus the subject enntity should be a city and he object entity member of the class 'country', Hope this gives good results.

6 Results

Good results only!

7 Discussion & Future Work

We will discuss certain aspects of our results and give advice on further research.

7.1 Discuss Aspect 1

Well well well

7.2 Future Work

Basing on the believe that the experiments were successful, we recommend:

- Improve the model (deeper)
- link the latent space to word signals e,g text
- prior not normal, e.g NF
- try a GAN

Good luck amigos!

Ideas

The bigger picture of this thesis is to efficiently generate a representation of the information hold in plain text. This representation has the form of a knowledge graph and consists of subject-relation-object triples.
Challenges:

- What knowledge are we looking for? I would like the model to focus on the most important information. This could for example be topic specific.
- Another option would be to extract based on a query or point of interest. Especially if we build the KG incrementally we can use this input as starting point and recursively build from there.

7.3 Use Normalizing Flows

- Is it possible to generate a graph from text using Normalizing flow (NF)?
- Can we train such a NF unsupervised, using the output distributions as loss function?
- NFs can build KG at once or incrementally. No one is using it, could have a reason.

7.4 Recurrent VAE

- Recursively generate graph using a Variational Auto Encoder.
- Use query as start node.
- expand graph on that node. Thiviyan and others work on VAEs as well. Seems an easier approach. Can analyze the latent space.

The encoder decoder strategy has been applied successfully in many cases. Bellis thesis about modeling a graph from images can be applied to text as well by changing the input to a text vector [6]. Note that this is a supervised method and would require a dataset of text labeld with resulting KGs. To the best of my knowledge this does not exist yet.
Same holds for the contrastive world model by Kipf. If we input text vectors instead of an image the model could recognize different objects in the text as it does with pixels [1]. Here the model is trained unsupervised using a loss over the energy function of the graph embedding space TransE [7]. An open question is if this would work for our approach.
A bit more abstract is the idea of the feedback recurrent VAE [8] where sound signals are encoded and decoded. This could also be adopted to text for instance with one sentence at a time, or a fixed number of tokens. Here the text vector would be induced as the latent input to the decoder. This would mean finding an translation of the models latent space to the work embedding. The dataset would consist of positive and negative examples of resulting graphs over timesteps.
While text can be easily vectorized by word embeddings like word2vec, the graph representation seems more tricky. A reasonable approach following Bellis example would be to output a coordinates vector for the nodes and an adjantency matrix for their relationships. Here one node at a time is outputted and the relation is conditioned on the number of previous nodes. Alternatively we could make use of the graph embeddings RASCAL or TransE. Lastly the question remains how to model the graph when the nodes need to be predefined. A subgraph of DBpedia could be a good starting point.

Open Issues

A section where note will be made on open issus. These aissues are ment to be discussed with Peter or Thivyian and ultimatly solved. No open issues should remain at the end of November.

7.5 Graph matching

Hungarian algorithm: Should we assume the affinity matrix is a cost or a profit matrix. If we assume it is a cost matrix and n != k the algorithm returns an error while padding. If we assume it is a profit matrix and convert it to a profit matrix it can handle any dimension. The shortest path varies with both approaches. Further I assume the shortest path is what we are looking for and mask these entries with 1 and the rest of the matrix with 0.
Diagonal of A is zeros. As soon as I do this, code crashes. Think it is because of divided zero, Martin adds 1e-7

to the X norm
Do we backpropagate through the graph matching? then all torch functions, otherwise np.
Martin takes 300 iterations for hungarian. also not looping over pairs.

References

1. Kipf, T., van der Pol, E. & Welling, M. Contrastive Learning of Structured World Models. *arXiv:1911.12247 [cs, stat]*. arXiv: 1911.12247. http://arxiv.org/abs/1911.12247 (2020) (Jan. 2020).
2. Kingma, D. P. & Welling, M. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*. arXiv: 1312.6114. http://arxiv.org/abs/1312.6114 (2020) (May 2014).
3. Tiao, L., Elinas, P., Nguyen, H. & Bonilla, E. V. Variational Graph Convolutional Networks. en, 11.
4. Kipf, T. N. & Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. en. *arXiv:1609.02907 [cs, stat]*. arXiv: 1609.02907. http://arxiv.org/abs/1609.02907 (2020) (Feb. 2017).
5. Simonovsky, M. & Komodakis, N. GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders. *arXiv:1802.03480 [cs]*. arXiv: 1802.03480. http://arxiv.org/abs/1802.03480 (2020) (Feb. 2018).
6. Belli, D. & Kipf, T. Image-Conditioned Graph Generation for Road Network Extraction. *arXiv:1910.14388 [cs, stat]*. arXiv: 1910.14388. http://arxiv.org/abs/1910.14388 (2020) (Oct. 2019).
7. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J. & Yakhnenko, O. in *Advances in Neural Information Processing Systems 26* (eds Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z. & Weinberger, K. Q.) 2787–2795 (Curran Associates, Inc., 2013). http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.pdf (2020).
8. Yang, Y., Sautière, G., Ryu, J. J. & Cohen, T. S. Feedback Recurrent AutoEncoder. *arXiv:1911.04018 [cs, eess, stat]*. arXiv: 1911.04018. http://arxiv.org/abs/1911.04018 (2020) (Feb. 2020).