Universiteit van Amsterdam

MSc Artificial Intelligence
Master Thesis

Knowledge Generation

by
Florian Wolf
12393339

October 22, 2020

48 Credits
April 2020 - December 2020

Supervisor:
Dr Peter Bloem
Chiara Spruijt

Asessor:
Dr Paul Groth

Abstract

We generate Knowledge! [**kipf˙contrastive˙2020**]

1 Introduction

Here comes a beutiful introduction. Promise!

2 Related Work

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.

2.1 The Graph VAE – one shot method

VAE
The VAE as first presented by [**kingma˙auto-encoding˙2014**] 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)

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 [**tiao˙variational˙nodate**].

RGCN
Realtional Graph Convolution Net (RGCN) was presented in [**kipf˙semi-supervised˙2017**] 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.

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 [**simonovsky˙graphvae˙2018**]. 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?***

2.2 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.

Graph matching algorithms

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 presented by

The 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 aalgorithm (refference) takles this problem by?? and is scalable.

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

Graph VAE loss The loss is discribed in [**simonovsky˙graphvae˙2018**] 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.

2.3 Knowledge Graphs

Knowledge Graphs are great! The best in the world.

3 Background

4 Metodology

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

4.2 Graph VAE

hi
Convolution part
RCGN relation Convolution neural net
MLP encoder
Latent space
reparametrization trick

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

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

4.4 Experiments and Metrics

First: Link prediction to make a proof of concept, not achieve SOTA.
Node classifier by only using encoder.
latent space interpolation to find analogies to smile vector in the latent space of a face VAEs.
Identify if VAE learns semantics. OWL, onthology datasetbatch$_size, required.$
$Wassersteindistance???Linkprediction?blabl$

5 Experiments

6 Results

7 Discussion & Future Work

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.

7.1 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.