UNIVERSITEIT VAN AMSTERDAM

MSc Artificial Intelligence
Master Thesis

Generative models on Knowledge Graphs

by
Florian Wolf
12393339

August 14, 2020

48 Credits
April 2020 - December 2020

Supervisor:
Dr Peter Bloem
Chiara Spruijt

Asessor:
Dr Paul Groth

1 Introduction

Here comes a beutiful introduction. Promise!

2 Background knowledge

In this section I will present my literature research up-to-date. The included topics are either relevant as background knowledge or state-of-the-art models.

2.1 Knowledge

We humans aquire knowledge audio-visually. This can be in form of a lecture, a movie, or book. The most common way to transfer knowledge is by text. For machines to be able to reason and work with knowledge, it needs to be transferred to a machine readable format. The most popular format is a tabular database. A newer database approach is the knowledge graph (KR) which is based on relations between entities. This lets the machine reason on this knowledge, answer complex questions and make conclusions more similar to human thinking. The format of KG is a triple consisting of a subject, a directed relation and an object. The task of converting knowledge from text to KG format is non-trivial. In this thesis, we would like to focus on the extraction of accurate knowledge. The idea behind it is that text possesses various levels of knowledge and we, as reader, are biased by our prior knowledge and the intention or drive for reading the text.

- Semantic Parsing
  The most old-school and technical approach is to tackle each sentence with NLP methods. This means tagging each word, linking references and finding relations. This will extract all possible triples. This can then be aligned with an existing database [**?**].
- Knowledge Graph
  Form of representing knowledge or simply a database. Makes information machine readable. It is build of relation triples.
- Triple SRO
  Triples are formed by two nodes and on link. The starting node is the subject, the second node is the object and the directed link is the relation of the subject with the object.
- Semantic Web
  The future of the internet where the data presented on websites can be read and understood by the browser. For this to be possible, websites should present its information in KG format in the metadata.
- Existing Methods in NLP
  Different semantic parser have been developed. They extract all possible kind of triples from text. While being grammatically correct, the extracted triples do not represent the information a human reader would get by reading the text. The Stanford Parser might be the most popular and advanced one [**?**]. Further the Allen NLP and OpenIE projects offer powerful parsing tools.

2.2 Embeddings

For any model we need an encoding to input text. Word embeddings have established themselves over the last decade as the solution. There are differences in how they are trained , how the relations between words is captured and how the context is represented.

- Word Embeddings
  word2vec is the most established word embedding, easy to train and implement [**?**]. Yet this method is becoming outdated and being replaced by newer solutions.

  BERT, also by Google, seems to be the sate of the art. It is able to predict words or full sentences as vectors. Using a bidirectional architecture and an attention score for each token, this model is able to catch much more context than its predecessors [**?**]. The attention score indicates how much other words point towards the selected word within a sentence [**?**]. This way the importance of the words can be compared. The model on it's own is a text classifier but can be tweaked to output word embeddings.
- Graph Embeddings
  TransE represents entities in in low-dimensional embedding. The relationships between entities are represented by the vector between two entities [**?**]. (How are different relation between the same entities represented?)

- OntoUSP
  This method learns a hierarchical structure to better represent the relations between entities in embedding space.

2.3 Learning Methods

A big challenge of this field is the lack of labeled data. Public knowledge graphs like DBpedia hold huge collections of knowledge. In my opinion, the size is the problem. The plain amount of data makes it hard to incorporate it in an efficient pipeline. Further, since it needs to cover all topics, the representation becomes less specific. Looking at the usecases of KGs, most tasks actually do not require this total coverage and instead are topic specific. Thus, we would like to focus on a targeted approach.

- Supervised Learning
  Requires a labeled dataset. For the case of text to graph the options are limited. Adecorpus provides such a dataset for drug interactions. Facebook research offers a QA text to graph dataset called babi. This dataset is a benchmark for question answering algorithms.

- Distant supervision
  makes use of the a database like DBpedia and infers labels by comparing the similarity of entities.

- Contrastive Learning
  Can be supervised or unsupervised approach and focuses on similarities between predictions. The loss is computed by the energy function of the output. To me this seems like an interesting approach to create a world model [**?**] and it has not yet been applied to plain text.

2.4 Recurrent Graph Models

The idea of creating a graph recurrently node by node seems intuitive. One recent example of generating a graph using a recurrent VAE architecture was [**?**]. Here the model generates gets bird-view images of roadmaps and generates a graph representation. each generated node is fed back into the encoder as prior and a stop signal is generated once finished. Applying this to knowledge from text is one of my two main ideas **??**.

Another recent approach uses recurrent Graph GAN [**?**]. Learns distribution of the training graphs. Creates Graph sequentially.

2.5 Graph Normalizing Flows

Create Graph all at once. Would it be possible to generate it recursively? Conditioned on query?

- unsupervised learning with CFG In the application of context free grammar, NFs have been trained on plain text [**?**]. The approach was unsupervised with a loss function which takes in account the distribution of the output prediction and the KL divergence between the distributions of each output. This made the model generate outputs with high certainty and which lay far apart from each other. The loss also encourages the output to be close to the input. For the input and output two different embeddings are needed. Thus, we need a similarity measure between them.

2.6 Variational Auto Encoder (VAE)

These models consist of an encode and a decoder. The encoder encodes the input to an low-dimensional latent space. The decoder takes a signal from latent space and reconstructs the input. Exploring the latent space makes it ossible to use the decoder as generative model. The posterior can only be aproximated by the ELBO

3 Related Work summary

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

VAE

The VAE as first presented by [**?**] 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 [**?**].

RGCN

Realtional Graph Convolution Net (RGCN) was presented in [**?**] for edge prediction. This model takes into account features of nodes. ***Elaborate***

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 [**?**]. 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 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 [**?**] as:

$$\text{-}\log p(G \mid \mathbf{z}) = -\lambda_A \log p\left(A' \mid \mathbf{z}\right) - \lambda_F \log p(F \mid \mathbf{z}) -$$
$$- \lambda_E \log p(E \mid \mathbf{z}) \log p\left(A' \mid \mathbf{z}\right) =$$
$$= 1/k \sum_a A'_{a,a} \log \widetilde{A}_{a,a} + \left(1 - A'_{a,a}\right) \log \left(1 - \widetilde{A}_{a,a}\right) +$$
$$+ 1/k(k-1) \sum_{a \neq b} A'_{a,b} \log \widetilde{A}_{a,b} + \left(1 - A'_{a,b}\right) \log \left(1 - \widetilde{A}_{a,b}\right)$$
$$\log p(F \mid \mathbf{z}) = 1/n \sum_i \log F^T_{i,\cdot} \tilde{F}'_{i,}$$
$$\log p(E \mid \mathbf{z}) = 1/\left(\|A\|_1 - n\right) \sum_{i \neq j} \log E^T_{i,j}, \widetilde{E}'_{i,j,\cdot}$$

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 $\log_p Eandlog_p Fjustlikelog_p Aandsubtracttheinverse.Otherwisethemodellearntopredict$

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.

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

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

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

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$_size, required.$

Wasserstein distance??? Link prediction? blabl

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.

4.5 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.