UNIVERSITEIT VAN AMSTERDAM

MSc Artificial Intelligence
Master Thesis

Generative models on Knowledge Graphs

by
Florian Wolf
12393339

July 9, 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.

Graph convolutions
CNNs have shown great results with images. 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 node attributes are passed on as messages to all their neighbors. Each convolution applies an activation function in between the steps [**?**].

Graph VAE
The model architecture of the Graph VAE presented in [**?**] combines both the previous methods. The input graph undergoes graph convolutions before it is flattened and projected into latent space. Next the reparametrization trick is applied and a simple MLP decoder is used to regenerate the graph. Graphs can be generated recursively or in an one-shot approach, as this paper does. This model will be the starting point for our research.

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 similartiry 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)

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

adjacency matrix edge attribute matrix node attribute matrix

Graph embeddings? unsupervised approach

4.2 Graph VAE

Convolution part
RCGN relation Convolution neural net
MLP encoder

[1] ***Thesis Repo***

Latent space
reparametrization trick
MLP decoder
Graph matching
Discretization of prediction

4.3 Loss function

If loss function should be permutaion invariant we need to do some kind of graph matching.
Different options for graph matching.
Maxpooling-algorithm:
Assumptions
Node to edge affinity equals 0 Summing over the neighbors means summing over the whole column
Hungarian algorighm for discretization of X
second option: Compare only graph structure. NX algorithms: Greedy, Shortest path ...

4.4 Metrics

WHAAAA O.o Link prediction?

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

NetworkX offers graph matching algorithms as do other repos. The drawback is that they are programmed for smaller graph such as molecule graphs with undirected edges and no edge atribues, or these attributes are not taken into account and only the structure is compared. My implementation of the max-pooling graph matching algorithm from [**?**] return a $n \times n \times k \times k$ matrix with integer values $0, 2$. Most probably something went wrong here since this makes little sense regarding interpretation. When graphs are compared with NetworkX using the greedy-matching or minimum distance, the similarity matrix is a 2×2 matrix for comparing two graphs. My question is which sort of graph matching makes more sense for our usecase. Further I struggle to interprete the similarity matrix - is that my job tho? Maybe its just meant to be used in the loss function as described!