Data Science Challenge Report

Nikolaos Kafritsas

Thomas Saltos

Athens University of Economics and Business

Introduction

The purpose of this document is to report the results for the Team Project of class Data Science Challenge, April - June 2021. The objective is to study and apply machine learning techniques to a link prediction problem. More specifically, given a citation network, we should create a classifier in order to predict whether a paper cites another paper. The goal is to use edge information to learn the parameters of a classifier and then to use the classifier to predict whether two papers (nodes) are linked by an edge or not.

Dataset

The dataset contains papers that have been published at machine learning, artificial intelligence, data mining and natural language processing conferences and journals. The file **edgelist.txt** contains all edges of the citation network. The **abstract.txt** contains the abstracts of the 138,499 papers. The **authors.txt** contains the authors for each paper. Finally, the **test.txt** file contains 106,692 node pairs, the source and the target node to predict whether they are linked by an edge.

Data Preparation

This assignment tackles the challenge in 2 parts. In the first part, the 5% percent of edges is removed from the main graph in order to be used as a validation set. This is also the most fundamental part of this assignment because it involves the feature engineering process. In the second part, the whole dataset was used as input to the best algorithm with the optimal hyperparameters that were found in the first part.

Feature Engineering

The feature engineering process was essentially the focus of this assignment and critical for solving this challenge. In general, the features can be categorised as ***graph-related***, which took advantage of the graph architecture (nodes and edges) and ***textual-related*** which drew information from the abstracts of each paper. Also, many of these features were created using both unsupervised and supervised methods. More specifically:

Graph-related:

A. Centralities and Node Proximities

Obviously, preferential centrality was used, which takes into account the centrality degree of each node. The intuition is that popular articles with many citations are more likely to get new links, and the classification algorithm considers the degree of both nodes of a pair to make a prediction. Essentially, the sum of degrees of each node pair and the absolute value of difference of degrees of each node pair were used as features. Another helpful feature was the number of common neighbours, a method which is based on the assumption that two nodes with many common neighbours may be connected.

Additionally, the Adamic Adar indexes and Katz indexes were also tested, however both methods were not fruitful. The Adamic Adar assigns lower score when the number of links of the common neighbour is higher. Probably, this metric is better suited for social networks. The Katz index takes the lengths of all paths between each pair of nodes into consideration, effectively measuring 1-hop, 1-hop dependencies. Nevertheless, it didn't work well in the current case study because this method requires calculating the alpha parameter (Attenuation factor). This factor depends on finding the principal eigenvalue, which due to the size of the graph it was very difficult to experiment with.

Last but not least, the intersection of common authors of each node pair is used as a feature. In particular, it’s a feature that makes sense to use and slightly helps the model, albeit not significantly like other features.

B. Unsupervised Methods

Two of the most popular methods regarding graphs belonging to this category is Deepwalk and Node2vec. Both methods belong to the ‘Random walks’ class, where the properties/importance of each node is estimated probabilistically through random walks/traversal of the graph.

More specifically, both methods attempt to estimate the likelihood of observing node v_i given all the previous nodes visited so far in the random walk. Then for each node, an embedding is learned, from which, like words in each sentence, a neural network can learn their representations in an unsupervised way (e.g using Skip-gram). The main difference between these methods is that Node2vec uses the BFS approach, essentially becoming more biased and focusing on the local neighbourhood, while Deepwalk uses the DFS approach.

In the initial phases of the challenge, both methods were proven helpful in boosting the model’s efficiency, despite the fact that they were overfitting and regularization was mandatory. However, after the autoencoders were introduced, both methods were not helping the model.

C. Supervised Methods

Another experiment was made in order to use graph representations using a graph autoencoder (Figure 1). Autoencoders are a special kind of neural network used to perform dimensionality reduction. Autoencoders are being composed of two networks, an **encoder** e and a **decoder** d. The encoder learns a non-linear transformation $e: X \to Z$that projects the data from the original high-dimensional space X to a low dimensional latent space Z. The $z = e(x)$ called latent vector. This vector is a low representation of a data point that contains information about x. A decoder learns a non-linear transformation $d: Z \to X$ that project the latent vectors back to the original high-dimensional input space X. An autoencoder is just a composition of the encoder and the decoder $f(x) = d(e(x))$. It is trained to minimize the difference between the input x and the reconstruction $\hat{x}$ using a kind of reconstruction loss.

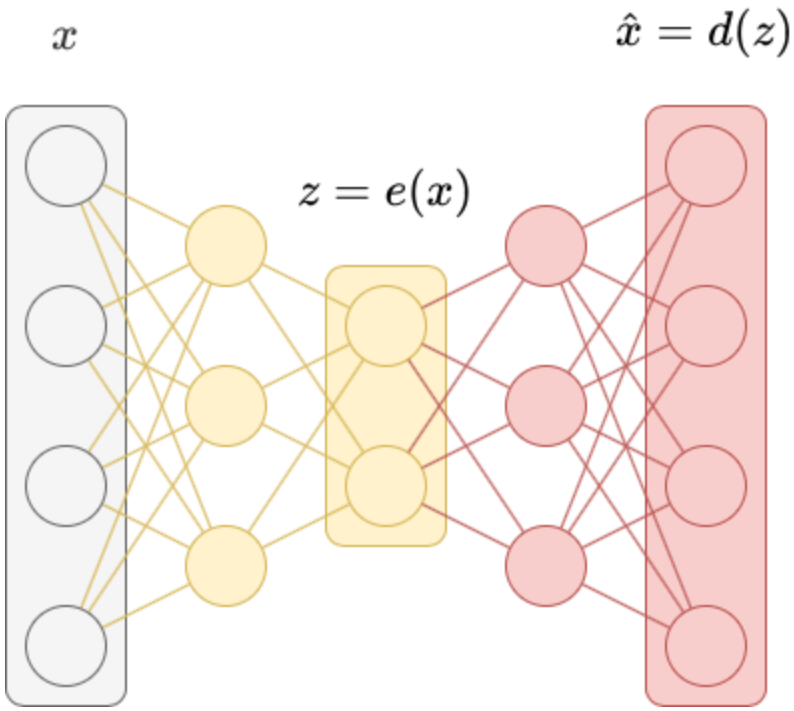

Figure 1: Graph *Autoencoder architecture*

The only constraint on the latent vector representation for traditional autoencoder is that the latent vectors should be easily decodable back to the original data point. As a result the latent space Zcan become disjoint and non-continuous. **Variational autoencoders** [1] try to solve this problem. Specifically, instead of mapping the input x to a latent vector $z = e(x)$, they map it instead to a mean vector $\mu(x)$ and a vector of standard deviations $\sigma(x)$. These parametrize a diagonal Gaussian distribution $N(\mu, \sigma)$, from which a latent vector are sampled $z \sim N(\mu. \sigma)$.

However, this does not completely solve the problem. There may still be gaps in the latent space because the output means may be significantly different and the standard deviations may be small. To reduce that, an auxiliary loss is added that penalizes the distribution $p(z|x)$ from being

too far from the standard normal distribution $N(0,1)$. This penalty term is the KL divergence, where the KL divergence is summed for each dimension.

When graph embeddings (cosine similarity and euclidean distance between nodes) was added the log-loss decreased significantly. Finally, graph embeddings from Adversary Regularized Graph Autoencoder [2] were tested but they didn't give the expected results.

Textual-related:

The basis for all text processing were the abstracts of each paper. Specifically:

A. Abstracts as sets of words

In this setup, the words of each abstract were converted to a set. To do this, each abstract underwent a series of transformations, which involved:

i) Converting all words to lowercase.
ii) Remove all special characters, http addresses, stopwords and digits.
iii) Lemmatization of words.

Then, the sets of each node are used to create the following features: i) The sum of lengths of each node pair, ii) their difference and iii) their intersection.

An attempt was also made to further extend this process by using td-idf vectorization, but was found inferior to the more advanced techniques which will be discussed below. On the other hand, using just the sets of words as described in this case was found beneficial for the challenge, probably because all words were used (even rare ones, most of which were specific terminology, and thus relevant for link prediction).

B. Doc2vec

Doc2vec is a technique based on Word2vec which creates a numeric representation /embedding for each document instead of words. Moreover, it proved to be superior to the bag of words model. As far as tuning is concerned, the PV-DM variation was used. In this case, the document embeddings are obtained by training a neural network on the task of predicting a center word based on an average of both context word-vectors and the full document's doc-vector.

Finally, although Doc2vec embeddings were not eventually used in the final dataset, they were utilized though as attributes in the autoencoder.

C. Bert

Last but not least, Bert-based models were also considered for this assignment, because they provide context-aware embeddings which are able to provide better precision. The particular case study makes use of the SPECTER language model [3], which is a pre-trained language model that provides document-level embeddings. The edge of this architecture over other transformers is that it is specifically trained on a subset of the Semantic Scholar corpus (uses SciBERT under the hood) and is optimized on inter-document relatedness, leveraging the citation graph.

Experiments in this case study indeed show that SPECTER outperformed the previous doc2vec model. Feature-wise speaking, the cosine similarity as well as the euclidean distance between the pretrained embeddings of a node pair were used as features.

Best Model

The best model with the optimal hyper parameters which was found in the first part was eventually used to produce the predictions for the challenge. The final part of this challenge was to find the most suitable classifier. Throughout the challenge, logistic regression was used as a point of reference to benchmark all setups. At the final stages of the challenge, more complex classifiers were tested, including MLPs, XGBoost and so on. The best algorithm which had an edge over the others was Random Forest. **Table 1** shows in detail the features which were used for the final model. Table 2 shows a few milestones setups with their respective scores:

Category	Feature
Centrality features	Sum of degree centralities
	Absolute difference of degree centralities
	Common Neighbors
Abstracts (set of words)	Sum of abstracts length
	Difference of abstracts length
	Intersection of abstracts length
Common authors	Intersection of common authors
Document embeddings (SPECTER)	Cosine similarity of embeddings
	Euclidean distance between embeddings
Variational Graph Autoencoder embeddings	Cosine similarity of embeddings
	Euclidean distance between embeddings

Table 1: *All features used in the final optimal model*

model_id	Model Features	Training Loss	Validation Loss	Training on the whole graph	Kaggle Score
1	Centrality features (CF)	0.257	0.255	0.253	0.257
2	CF + Abstracts with unpreproccesed bag of words	0.233	0.232	0.23	0.229
3	CF + Abstracts with optimized bag of words (BOW)	0.201	0.199	0.198	0.199
4	CF + BOW + common authors (CA)	0.199	0.199	0.198	0.197
5	CF + BOW + CA + Doc2vec	0.1794	0.1801	0.1738	0.1735
6	CF + BOW + CA + SPECTER	0.1543	0.1553	0.1520	0.1517
7	CF + BOW + CA + SPECTER + DeepWalk + Node2vec	0.04	0.1375	0.08	0.1213
8	CF + BOW + CA + SPECTER + Autoencoder	0.07	0.125	0.08	0.1145
9	CF + BOW + CA + SPECTER + Variational Autoencoder	0.08	0.125	0.085	0.10980
10	CF + BOW + CA + SPECTER + Variational Autoencoder with best hyperparams on Random Forest	0.075	0.1139	0.082	0.10754

Table 2: *Milestone experiments and their respective scores*

First of all, in **Table 2**, all models except the last ones were trained using Logistic Regression, while the last was trained with Random Forest. Moreover, models which used either Deepwalk, Node2vec, or Autoencoder embeddings were prone to overfitting, therefore regularization was necessary. The choice of Random Forest is ideal in this case, because it is a very efficient algorithm which simultaneously provides a variety of parameters to tune regularization.

As far as training time is concerned, the usage of more features obviously increases total execution. The creation of the training matrix takes at most 20 minutes, while model fitting takes around 2-3 minutes. The most time consuming tasks involved creating the embeddings, with the encoding of the sentence embeddings from Spectre taking the longest. Moreover, training the autoencoder was also time consuming, unless a GPU is available which significantly reduces training time.

Final Remarks

- The most crucial part of this challenge is to find the best features, and this is the task which provides the edge in this challenge. The choice of the classification algorithm is less important.
- Time is of the essence in this challenge, especially because it involves a relatively large dataset, and thus benchmarks should be constructed in such a manner that allows for quick experimentation
- It was extremely helpful to investigate the misclassified examples in the validation set and find out how feature engineering can be improved. For instance, initially, *large-bert* was used, but some common terminology present in some linked papers did not output the suitable high cosine similarity score. This gave the idea for searching more specific transformers for this task.
- Negative sampling is an important technique when creating the negative instances during the creation of the training dataset. In this case study, the negative sampling was performed by sampling at random 2 unconnected, different nodes. Additional effort was made to experiment with neighbor sampling, which is for a small subset of nodes, take the most similar unconnected nodes, in terms of node2vec and document Jaccard similarity. However, this method did not yield any useful results, in terms of log loss improvement

References

[1] T. N. Kipf and M. Welling, "Variational Graph Auto-Encoders," *ArXiv161107308 Cs Stat*, Nov. 2016, Accessed: Jun. 23, 2021. [Online]. Available: http://arxiv.org/abs/1611.07308

[2] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially Regularized Graph Autoencoder for Graph Embedding," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, Jul. 2018, pp. 2609–2615. doi: 10.24963/ijcai.2018/362.