Project description:

The purpose of this project is to implement a distributed version of LSTM Autoencoder using GPU. LSTM autoencoder is a part of a bigger model LSTM-Node2vec that is implemented and submitted for publication. LSTM-Node2vec model is a neural network model for graph embedding that can be used to represent graphs in different data mining tasks including link prediction, anomaly detection and node classification and outperforms state-of-the-art models in most of the cases.

We are interested in having a distributed implementation of this model. LSTM-Node2vec consists of two part: LSTM-autoencoder and Node2vec. In this project our goal is to implement the LSTM-autoencoder component. Below, we describe LSTM autoencoder structure and its training.

1) LSTM Autoencoder

Long Short Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) that is commonly used in Natural Language Processing. One common usage of LSTM is to generate word representation. Given a set of sentences, an LSTM autoencoder is trained to take a sentence as input and outputs the same sentence. Given a sentence S={s_1, s_2, ..., s_n} of length n where s_i is the ith word in the sentence, an LSTM autoencoder takes the words one by one using the one-hot word representation and is trained to produce the same word in the output layer. After the training, the weights in the input layers are the word representation for s_i is obtained from learned weights W_i in LSTM.

Inspired by the LSTM effectiveness in NLP, we use LSTM to learn node representations in graphs. Here, a sequence of neighbors of a node at different time points is considered as a sentence. These sequences reflect the way that the neighborhood of a node evolves over time. By training an LSTM over a set of sequences, LSTM can capture the temporal dependencies between the neighbors of nodes in the graph evolution. Student will have access to the generated sequences for existing dataset.

LSTM autoencoder is previously implemented in Keras library, Python. The student is expected to convert the implementation to a tensorflow based model. Our final goal is to implement this model in GPU. Therefore, student can start with a CPU implementation and then convert the model to GPU implementation. Finally, the model should be implemented in a distributed way and run on the lab GPU cluster.

2) LSTM autoencoder structure and training

LSTM autoencoder consists of one LSTM layer as the encoder and one LSTM layer as the decoder. This model takes a sequence as input and outputs the same sequence. For training the model, we generate the input sequences to this model. Figure 1, shows the general structure of the model.

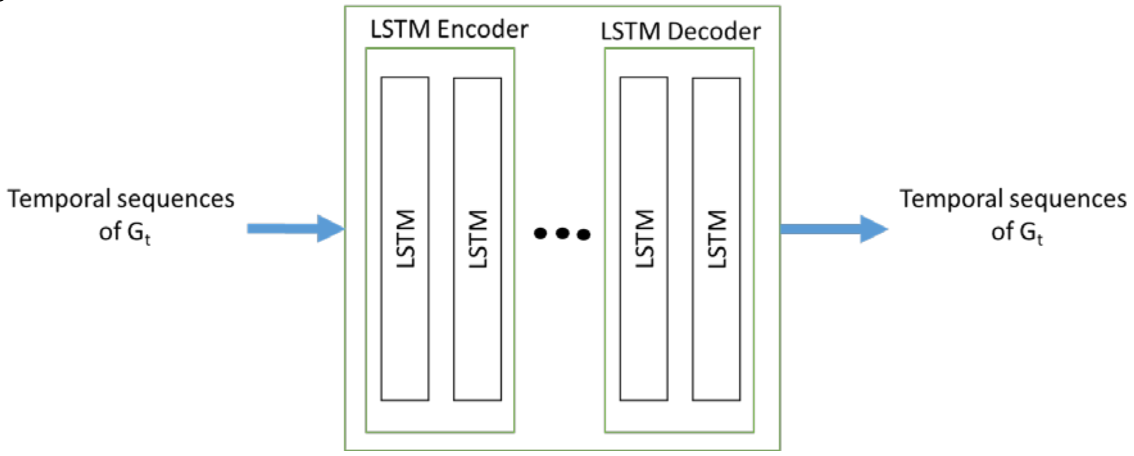

Figure 1, LSTM autoencoder model

Requirements:

- Prerequisite courses: list of courses already taken, minimum mark requirement

  Data Mining (4412), B.

- Corequisite courses: list of courses to be taken concurrently

  None.

- Other experience: technical know how needed, familiarity with languages and systems, mathematical background, etc.

  Good programming experience

Resources: This is worth spelling out because it may be more helpful to the student in choosing a project than the project description.

- Software: list of programming languages, tools, operating systems, etc. (e.g. C++, Unix, Tcl/Tk, Slowgo spreadsheet)

Programming language: Python
Libraries in Python: Tensorflow, Keras

• Hardware: list of computer and equipment characteristics (e.g. Linux workstation with X windows, Acme laser scanner)

The programs will run from Linux servers or Windows workstations. The information about the CPU and GPU cluster can be found in the following links:

https://wiki.eecs.yorku.ca/lab/datamining/gpuservers:start

https://wiki.eecs.yorku.ca/lab/datamining/brain:start

Readings: A short list of research papers/books that could/must be studied or referenced for the project, that can help give some idea of what your project is about and the background required.

1. P. Goyal, E. Ferrara, Graph embedding techniques, applications and performance: a survey, Knowledge based systems journal 2018
2. Y. Zuo, G. Liu, H. Lin, J. Guo, X. Hu, J. Wu, Embedding temporal network via neighborhood formation, KDD 2018
3. Wang, Peilu, et al. "Learning distributed word representations for bidirectional lstm recurrent neural network." Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2016.
4. Chen, Jinyin, et al. "E-LSTM-D: A Deep Learning Framework for Dynamic Network Link Prediction." arXiv preprint arXiv:1902.08329 (2019).
5. Chen, Jinyin, et al. "GC-LSTM: Graph Convolution Embedded LSTM for Dynamic Link Prediction." arXiv preprint arXiv:1812.04206 (2018).
6. Reading on deep learning methods specifically on LSTM
7. Reading on design a distributed algorithm
8. Reading on tensorflow and distributed tensorflow libraries of Python
   https://www.tensorflow.org/
   code: https://github.com/tensorflow/examples/blob/master/community/en/docs/deploy/distributed.md
   video tutorial: https://www.youtube.com/watch?v=-h0cWBiQ8s8

Deliverables: List of artifacts to be completed by the final due date; includes programs, documentation, reports, user guides, etc.

1. Distributed implementation of LSTM autoencoder using tensorflow on GPU server
2. Source programs implemented in the project.
3. A user guide/documentation on how to use the implemented system and programs.

4. Partial reports of the progress for monthly meetings
5. A report that includes an introduction, a brief description of related work, a detailed description of the system that the student designs and implements for the project, the reasons why the system is so designed, a description of the experimental results on graph datasets, and open issues that remain to be solved in the future.

Work Plan and Milestones List: Milestones to be reached during the project. Useful for the project contract and progress evaluation.

1. Jun 7: Background reading completed
2. Jun 14: Implementing a simple neural net model in the cpu
3. Jun 21: Implementing a simple neural net model in the gpu
4. Jun 28: Implementing a simple neural net model in a distributed way
5. Jul 5: Implementing LSTM autoencoder model in tensorflow based on the existing code in keras
6. Jul 12: Implementing LSTM autoencoder model in tensorflow
7. Jul 19: Implementing the tensorflow based LSTM autoencoder model in gpu
8. Jul 26: Implementing the tensorflow based LSTM autoencoder model in gpu in a distributed way
9. Aug 2: Testing the implemented model in gpu server
10. Aug 9: Evaluating the implemented model
11. Aug 16: Final checks
12. Aug 23: Submitting the final report
13. TBD: Project presentation

As the project progresses, we may discover complications and other issues. In such cases, the supervisor (or the PhD student supervising the work) and student may make certain changes to the milestones, deliverables, and evaluation criteria. We also have monthly project meeting. The student is expected for deliver partial reports of the progress for the meeting.

Evaluation: Some indication as to how the project will be evaluated. For example, the project can be marked on the basis of the progress towards stated goals, and the clarity and completeness of programs, documentation and reports. It is strongly recommended and encouraged that evaluation has a number of components and not just a grade at the end. Grades can be associated with different deliverables or their parts. Components could include: documentation, programming style, quality of the interface design, project proposal presentation, project status presentation, and overall evaluation.