AFGCN v2: A GCN-based Approximate Solver

1st Lars Malmqvist
Department of Computer Science
University of York
York, UK
lars.malmqvist@york.ac.uk

***Abstract*—AFGCN v2 is an approximate abstract argumentation solver that computes the credulous or skeptical acceptance of arguments using an approximation method based on a Graph Convolutional Network model. This model is trained on a dataset derived from various competitions, utilizing a randomized training regime designed to maximize generalization. At runtime, the solver employs a Python script to calculate input features, including the grounded extension and additional graph-based properties, and then uses the GCN model to infer the acceptability status of arguments. AFGCN v2 builds upon previous work on approximating the acceptability of abstract arguments by introducing improvements to the input features used for approximation.**

***Index Terms*—abstract argumentation, GCN, ICCMA**

I. Introduction

Abstract Argumentation is a formalism for non-monotonic reasoning that focuses on the representation of conflict. It is typically represented as a directed graph, where vertices denote arguments and edges indicate a relation of attack. This leads to various reasoning problems that determine the acceptability of arguments or the joint acceptability of sets of arguments. Most of these reasoning problems are known to be NP-hard [1], [2].

The AFGCN v2 approximate abstract argumentation solver employs a Graph Convolutional Network, a subclass of Convolutional Graph Neural Networks [3], to compute approximate solutions for the credulous or skeptical acceptability of arguments in a given abstract argumentation framework. The model has been trained on a dataset consisting of argumentation frameworks from past competitions using a randomized training methodology that aims to maximize generalization from the input frameworks. Moreover, the solver uses the precomputed grounded extension as an input feature for the neural network to expedite computation and slightly enhance accuracy. The solver also applies a configurable probability threshold that can vary according to the semantic and framework size for increased runtime accuracy.

II. Approximating Argumentation Frameworks Using Convolutional Graph Neural Networks

Convolutional Graph Neural Networks [3] (CGNNs) build on the success and popularity of traditional Convolutional Neural Networks, which define the state of the art in several subfields of deep learning, particularly in computer vision. However, there are various methods for defining the convolutional operation when applied to graphs, resulting in different types of CGNNs. The most common approach is based on digital signal processing, where convolution is essentially a noise removal operation. This is also the approach adopted by the groundbreaking Graph Convolutional Network (GCN) by Kipf and Welling [4], which is the architecture that AFGCN v2 adapts for approximating the acceptability of abstract arguments.

The core GCN architecture has been extended using deep residual connections between layers, input features based on the grounded extension, and a randomized training regime that continuously shuffles both the frameworks to predict and the values within those frameworks to improve generalization. AFGCN v2 is built upon previsou work by Malmqvist et al [5], [6].

The key components of the GCN architecture used include the following elements:

1) Randomized input features combined with input features generated from the grounded extension of the argumentation framework and input features based on graph properties
2) An input layer receiving these inputs
3) 4 repeating blocks of a GCN layer [4] and a Dropout layer [7]
4) Residual connections feeding the original features and the normalized adjacency matrix as additional input at each block
5) A Sigmoid output layer generating a probability for the acceptability of each argument in the framework

The model was trained using Adam [8] with Binary Cross-Entropy as the loss function and a variable learning rate. The training regime employed a combination of randomized training batches, dynamic rebalancing of the training data, and automated outlier exclusion to prevent overfitting and achieve a high degree of accuracy.

A. Input Features

AFGCN v2 incorporates input features by including the grounded extension as an input along with randomly initialized features. In addition to these, the AFGCN v2 solver incorporates a set of new features derived from various graph properties. The new features are calculated using the following graph metrics: graph coloring, PageRank, closeness centrality, eigenvector centrality, in-degrees, and out-degrees.

Graph coloring assigns a color to each node in the graph such that no two adjacent nodes share the same color. PageRank is an algorithm that measures the importance of nodes in the graph, assigning a higher rank to more central nodes. Closeness centrality is a measure of the degree to which a node is central in the graph, and it is calculated as the reciprocal of the sum of the shortest path distances between the node and all other nodes in the graph. Eigenvector centrality assigns a relative score to each node based on the principle that connections to high-scoring nodes contribute more to the score of the node in question than connections to low-scoring nodes. In-degrees and out-degrees represent the number of edges pointing towards and away from a node, respectively.

These raw features are computed for each node in the graph, and a feature vector is created by concatenating the values of each metric. To ensure that the features are on a comparable scale, the feature vectors are normalized using a standard scaler, which transforms the data such that it has zero mean and unit variance. The resulting normalized features are then used as input to the GCN model, providing a richer representation of the graph structure and potentially improving the solver's performance in approximating argument acceptability.

III. Implementation

A. Design of the Solver

The chosen model for the final solver runtime is a 4-layer model with 128 features per layer. It was trained on a dataset containing instances from various ICCMA competitions.

The solver has been developed using the Python programming language, leveraging the Pytorch framework for training and modeling, the Deep Graph Library for graph representation, and Numpy for numerical computation.

At runtime, the solver is invoked using a shell script wrapper that conforms to the specifications. This shell script calls a Python script that loads the relevant parameters into the GCN model based on the semantic in question. It then precomputes the grounded extension using a Numpy-based grounded solver and passes this information along with a random input feature to the GCN model for inference.

The output of the inference step is then passed to a probability threshold function, which applies a threshold for acceptance that is adapted to the size of the argumentation framework and the semantic under consideration. The solver calculates the acceptability status of all arguments in the argumentation framework in parallel during the inference step, but to conform with the solver specification, it only outputs the predicted status for the particular argument under consideration.

B. Competition Specific Information

The solver implements functionality for the approximate track. It is not submitted for any other tracks. Within the implements functionality for five included semantics: CO, PR, ST, SST, STG, and ID.

Both problem types (DC and DS) are supported for CO, PR, ST, SST, and STG semantics. For the ID semantic, DS is supported.

The solver can be called in the following manner:

```
python -W ignore::UserWarning:
afgcn_solve.py --filepath=<file>
--task=<problem_type>
--argument=<argument_num>
```

Example:

```
python -W ignore::UserWarning:
afgcn_solve.py --filepath=testaf1.txt
--task=DS-PR --argument=4
```

References

[1] G. Charwat, W. Dvořák, S. A. Gaggl, J. P. Wallner, and S. Woltran, "Methods for solving reasoning problems in abstract argumentation - a survey," *Artificial Intelligence*, vol. 220, pp. 28–63, 2015.

[2] S. Woltran, "Abstract argumentation – all problems solved ?" *ECAI 2014*, pp. 1–93, 2014.

[3] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2020.

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 9 2019. [Online]. Available: http://arxiv.org/abs/1609.02907

[5] L. Malmqvist, "Afgcn: An approximate abstract argumentation solver," *ICCMA 2021*, 2021. [Online]. Available: http://argumentationcompetition.org/2021/downloads/afgcn.pdf

[6] ——, "Approximate solutions to abstract argumentation problems using graph neural networks," 2022. [Online]. Available: https://etheses.whiterose.ac.uk/32152/6/thesis.pdf

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[8] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization." *ICLR 2015*, pp. 1–15, 2015. [Online]. Available: https://arxiv.org/pdf/1412.6980.pdf