AFGCN: AN APPROXIMATE ABSTRACT ARGUMENTATION SOLVER

Lars Malmqvist
Department of Computer Science
University of York
York, United Kingdom
lm1775@york.ac.uk

ABSTRACT

AFGCN is an approximate abstract argumentation solver presented to the ICCMA 2021 competition. It approximates solutions to the credulous or sceptical acceptance of arguments using an approximation method based on a Graph Convolutional Network model. This model is trained on a dataset taken from past ICCMA competitions, using a randomised training regime designed to maximise generalisation. This is deployed at runtime using a Python script that computes input features and then uses the GCN model for inferring the acceptability status on arguments. AFGCN extends past work on approximating the acceptability of abstract arguments by introducing improvements to the training regime, incorporating the grounded extension as an input feature, and combining all elements in a single runtime script.

1 Introduction

Abstract Argumentation is a formalism for non-monotonic reasoning that bases its representation on the modelling of conflict. It is typically represented in the form of a directed graph in which vertices represent arguments and edges a relation of attack. This gives rise to a range of reasoning problems that determine the acceptability of arguments or the joint acceptability of sets of arguments. The majority of these reasoning problems are known to be NP-hard[1, 2].

The AFGCN approximate abstract argumentation solver uses a Graph Convolutional Network, a subspecies of Convolutional Graph Neural Networks [3], to compute approximate solutions to the credulous or sceptical acceptability of arguments in a given abstract argumentation framework. The model has been trained on a dataset consisting of argumentation frameworks from past ICCMA competitions using a randomised training methodology that seeks to maximise generalisation from the input frameworks. Additionally, the solver uses the pre-computed grounded extension as an input feature to the neural network to speed up computation and slightly increase accuracy. The solver also deploys a configurable probability threshold that can vary by semantic and framework size for increased runtime accuracy.

2 Approximating Argumentation Frameworks Using Convolutional Graph Neural Networks

Convolutional graph neural networks [3] (CGNNs) draw on the popularity and success of traditional Convolutional Neural Networks that define the state of the art in several subfields of deep learning, particularly in computer vision. There are, however, different ways of defining the convolutional operation when it is applied to graphs, which gives rise to different types of CGNNs.The most common approach bases itself on the digital signal processing where convolution is seen effectively as a noise removal operation. This is also the approach taken by the seminal Graph Convolutional Network (GCN) by Kipf and Welling [4], which is the architecture that AFGCN adapts to the approximation of the acceptability of abstract arguments.

The core GCN architecture has, however, been extended using deep residual connections between layers, input features based on the grounded extension, and a randomised training regime that shuffles both the frameworks to predict and what values within those frameworks on a continuous basis to improve generalisation. AFGCN is an extension of the work presented by Malmqvist et al. at SAFA 2020 [5].

The core components of the GCN architecture used includes the following elements:

1. Randomised input features combined with input features generated from the grounded extension of the argumentation framework
2. An input layer receiving these inputs
3. 4 repeating blocks of a GCN layer [4] and a Dropout layer [6]
4. Residual connections feeding the original features and the normalised 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 [7] with Binary Cross-Entropy as the loss function and a variable learning rate. The training regime used a combination of randomised training batches, dynamic rebalancing of the training data, and automated outlier exclusion to prevent overfitting and reach a high degree of accuracy.

Compared to the SAFA 2020 version, AFGCN has had significant changes in the way input features are handled, especially by the inclusion of the grounded extension as an input, minor changes to the residual connections between layers to avoid training errors in rare circumstances, and has introduced additional randomisation techniques applied during training to generate randomised training masks on an argument by argument basis.

3 Implementation

3.1 Design of the Solver

The model chosen for the final solver runtime is a 4-layer model with 128 features per layer. It was trained on a dataset containing all instances from the 2015 and 2019 competitions and a 538 graph subset from the 2017 competition.

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

At runtime the solver is called using a shell script wrapper that conforms to the specifications of ICCMA 2021. This shell script calls a Python script that loads the relevant parameters into the GCN model based on the semantic in question. It then pre-computes 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 ICCMA 2021 solver specification it only outputs the predicted status for the particular argument under consideration.

3.2 Competition Specific Information

The solver implements functionality for the approximate track of the ICCMA 2021 competition [8]a. It is not submitted for any other tracks. Within the implements functionality only for the 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, conforming to the solver requirements set out in the ICCMA 2021 call for solvers:

./solver.sh -p <problem> -f <file> -fo <format> -a <argument>

Example:

./solver.sh -p DS-ST-D -f myFile.tgf -fo tgf -a a2

References

[1] Günther Charwat, Wolfgang Dvořák, Sarah A Gaggl, Johannes P Wallner, and Stefan Woltran. Methods for solving reasoning problems in abstract argumentation - A survey. *Artificial Intelligence*, 220:28–63, 2015.

[2] Stefan Woltran. Abstract Argumentation – All Problems Solved ? In *ECAI 2014*, pages 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*, pages 1–21, 2020.

[4] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, sep 2019.

[5] Lars Malmqvist, Tommy Yuan, Peter Nightingale, and Suresh Manandhar. Determining the acceptability of abstract arguments with graph convolutional networks. *CEUR Workshop Proceedings*, 2672:47–56, 2020.

[6] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[7] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *ICLR 2015*, pages 1–15, 2015.

[8] Rossit J Lagniez JM, Lonca E, Mailly JG. Introducing the Fourth International Competition on Computational Models of Argumentation. In *Proc. of SAFA'20*, pages 80–85, 2020.