Evidence Reasoning is a Graph Problem.

Andrew Georgiou
King's College
apjg4@cam.ac.uk

1 Introduction

Claim Verification is the main component of the fact-checking pipeline. Fact-checking consists of retrieving documents relating to a given claim, extracting evidence that relates to a claim and finally verifying the claim given the extracted evidence. FEVER (Fact Extraction and VERification) (Thorne et al., 2018) is one such dataset that enables the full fact-checking pipeline. Work by (Zhou et al., 2019) and (Wang et al., 2019) investigated the applications of Graph based approaches for claim verification with promising results.
In my project I wish to see to what extent can we improve upon the claim verification task with the application of more up to date SoA applications for document extraction such as GENRE (De Cao et al., 2020) in order to improve the evidence reasoning step that graph neural networks perform so well at. To do this I propose a graph attentional neural network and a message passing graph neural network. Which will be trained and evaluated on the full FEVER shared task score system.

Claim: [SUPPORTED] There was an attempt to incorporate Cyprus into Greece.
True Evidence: {51984, 61837, 'Cyprus', 16}
Predicted Evidence: {'Cypriot_Nationalism', 5}, {'Northern_Cyprus', 1}, {'Cypriot_Nationalism', 1}, {'Northern_Cyprus', 9}, {'Cyprus', 16}
Evidence Data: Cyprus (16): "On 15 July 1974, a coup d'etat was staged by Greek Cypriot nationalists and elements of the Greek military junta in an attempt at enosis, the incorporation of Cyprus into Greece."

Table 1: Example case, that only requires a single piece of predicted evidence, predicted with k=5 evidences.

2 RELATED WORK

2.1 FEVER Shared Task

The FEVER Shared Task (Thorne et al., 2018) was built as a challenge for participants to build systems which can classify whether factual claims are either *supported*, *refuted* or *not enough information* given a set of evidence extracted from Wikipedia. This challenge is characterised as a pipeline problem as it has different components which build to generate a final FEVER score.
These components are *Document Retrieval*, this surrounds the process of extracting the correct wikipedia documents for a given claim. *Sentence Selection*, the process by which we extract the top 5 sentences from our documents, intending to identify the best possible sentences to either support or refute our claim. *Claim Verification*, this is the focus of my paper and is the process of utilising the sentences extracted to be able to classify whether a given claim is *supported*, *refuted* or there is *not enough information*.
The proposed Baseline system which I intend to outperform has an overall FEVER score of 0.2771 which used a combination of techniques for each stage of the pipeline. These include k-nearest document extraction through cosine similarity between unigram and bigram Term Frequency Inverse Document Frequency (TF-IDF) (Qaiser and Ali, 2018) vectors, they then rank these vectors for sentence selection and finally they use a multi-layer-perceptron with a single hidden layer for claim verification.
In the original shared task, the best FEVER score was produced by UNC-NLP (Nie et al., 2018) which achieved an overall FEVER score of *0.6398* through utilising three homogeneous neural semantic matching models that conduct document retrieval, sentence selection and claim verification.

In this paper I will be focusing on the claim verification stage although will be comparing my full-FEVER score results against the baseline model and a similar graph neural network model that should provide competitive results. Table 1 shows an example of what the evidence data might look like for a given claim.

2.2 Pre-trained Language Models

Large Pre-trained language models such as BERT (Devlin et al., 2018) (Vaswani et al., 2017) and OpenAI's GPT3 (Brown et al., 2020) have proven to be highly successful at a large domain of Natural Language Processing tasks. RoBERTa (Liu et al., 2019) is an improvement over BERT that utilises better pre-training methods, removed next sentence prediction and added dynamic masking patterns.
These models allow for true bi-directionality for predictions unlike Bi-LSTM's. Throughout this project Language Models are utilised as previous work has proven they are effective in every facit of the fact-checking pipeline (Soleimani et al., 2020). Language models applied to both evidence retrieval and claim verification have been highly successful in fact-checking as pre-trained language models are already great encoders of information, acting as knowledge bases (Petroni et al., 2019).
Transformer architecture can be constructed and have proved to be quite effective as graph transformers (Yun et al., 2019) which is out of the scope of this project but something to look at for architectural design choices in my project. For my project I will be utilising both a BERT and RoBERTa model for sentence selection from a list of given documents and experiment with utilising the transformer architecture in some layers of the graph neural network.

2.3 Graph Neural Networks

Previous work of applying Graph Neural Networks began with (Zhou et al., 2019). They used a evidence reasoning network to propagate information between evidence nodes and an evidence aggregator to experiment with different concatenations of data that result in the final hidden state representation. Their overall result shown a FEVER score of 0.6710, although their full pipeline did make use of a combination of top performing FEVER task techniques such as applying language models for sentence selection and utilising UKP Athene for Document Retrieval.
The current SOTA in graph neural networks employs a Kernal Graph Attention Network (KGAT) (Wang et al., 2019). Kernal-based attentions enable the graph to conduct more fine-grained claim verification, which resulted in them scoring a FEVER score of 0.7038, which was the top score at its time.

3 Methodology

I approached this project in an attempt to compete competitively in the full FEVER shared task. For my specific approach I utilised the three section pipeline I discussed in section 2.1.
In this methodology section I will discuss how I approached each section of the pipeline with a detailed focus on the graph neural network architecture.

3.1 Document Retrieval - GENRE

For the document retrieval component I made use of recent work by Nicola et al (De Cao et al., 2020). Their system Autoregressive Entity Retrieval (GENRE) has not been thoroughly tested and evaluated on FEVER, although in their original paper they note the highly competitive results of GENRE in page-level document retrieval. Through my own testing I have found it to perform competitively against the baseline system for document retrieval, although in Table 2 we can see it still underperforms in comparison to the best document retrieval system from the original FEVER shared task, Athene UKP (Hanselowski et al., 2018).
This result is expected as it also utilises an entity linking approach but without the same precision in fine-tuning only on the FEVER Wikipedia dataset.
Implementing GENRE was done through their library. It poses many issues in implementing as it is not up-to date with many python libraries and requires downgrading. Further, GENRE is built using fairseq (Ott et al., 2019) which resulted in more library versioning issues.
Once I passed these issues I then found GENRE's models and trie required for Wikipedia entity linking exceeded the 8GB VRAM on my system thereby requiring external GPU processing. During this stage I would select the top 5 document's generated from GENRE for each claim and process this into a separate CSV file that the sentence selection model task will utilise to select the top 5 sentences across the documents.
One problem uncovered when attempting to compute the top 5 sentences for the entire training dataset is that the pipeline between GENRE and

sentence selection is relatively slow (around 30 seconds per 5 sentences) which for all 140,000 claims in the training set results in 70,000 hours of extraction time or close to 3000 days to extract. Attempts to optimise by utilising dictionary lookup and batch predictions reduced the time but only allowed for a subset of the training set to be utilised. I ended up extracting 100,000 evidential sentences for 20,000 claims which took almost 2 weeks of continuous computing time.

3.2 Sentence Selection - Pre-trained Language Models

Before I began training a model for sentence selection I had to optimise my approach to this problem. I began by building the Wikipedia dataset which links page titles to content. I took the dataset and converted it to a dictionary for O(1) lookup for each document extracted from GENRE, which exceeded 850,000 documents. Using the gold-standard evidence which lists the document and the sentence choice for a given piece of evidence. I then used GENRE to extract all the sentences from each given document and classify sentences used for evidence as '1' and sentences not used for evidence as '0'. This would be used in training the final sentence selection model.

For sentence selection I employed both a binary BERT and RoBERTa models. Implementing them was done through a sentence-pair classifier with a linear layer that outputs to a single feature output for binary classification. The input to the model is a pair of sentences [Claim, Evidence] and the label which is a binary output of whether the evidence is used verify that claim or not. The input is first tokenised using the BERT/RoBERTa tokeniser to convert each string to a sequence of integers that equate to words in a vocab file. This results in a heavily 0-class-weighted dataset, this was handled by biasing the class-weights for both models.

The model would then train for 5 epochs with the best model being selected from each checkpoint. This number of epochs seems low but due to the size of the model it very quickly generalises and test loss begins to increase after around 1 epoch. The RoBERTa model showed much higher performance after 1 epoch with an Mathew's Coefficient Correlation score of 0.63 compared to BERT's 0.33. Following from this the models still began to over-predict a single class therefore only the first epoch model's were used.

3.3 Claim Verification - Graph Neural Network

Sentence Encoding

My approach to claim verification is through building a fully connected graph of evidence nodes, each piece of evidence is concatenated with the original claim. Each node represents one of the five evidence sentences and finally one node is the claim, for extracting the best possible features from each piece of evidence I used a pre-trained language model to generate the embedding for each sentence. This required using the sentence-transformers library with the 'all-mpnet-base-v2' model $MPNET$as it maps to a 768 dimensional dense vector space. What this signifies is that given a list of N sentence evidences $\{e_1, e_2, e_3 ... e_N\}$ and a single claim c we feed each evidence, claim pair (c, e_i) into our all-mpnet-base-v2 model as well as the claim c without any concatenation to finally obtain our embedding representation er_i.

$$er_i = MPNET(c, e_i)$$

$$er_i = MPNET(c)$$

This representation will be the features used for our graph nodes.

Graph Structure

Graph: Consider a graph $G = (V, E)$ where V is the set of n nodes, and E is the set of edges associated with the nodes. For each node $i \in V$ I was given a d_n-dimensional initial feature vector $h_i \in R$ where $d_n = 768$ dimensional dense vector.

Each node V_i is connected to every other node V_n including a self-loop to propogate self-node information.

Label / Target: With each Graph Structure I attach the target label $y = Z$ where y can be the integer's $0, 1, 2$ for the classes:

0 : $NOT\_ENOUGH\_INFORMATION$

1 : $SUPPORTS$

2 : $REFUTES$.

Message Passing Neural Network

In order to encourage information propagation, I designed a message passing neural network which propagates information, concatenates embedding features between two nodes and updates them through a series of linear, batch normalisation layers with ReLU activation function. To formalise the structure of our message passing network:

Message Passing Layer: The message passing layer functions by iteratively updating the nodes encoded features ${h^\ell}_i \in R^d$ from layer ℓ to layer $\ell + 1$ via the following equation with which I will further formalise below:

$$h_i^{\ell+1} = \phi(h_i^\ell, \oplus_{j \in \mathcal{N}_i}(\psi(h_i^\ell, h_j^\ell)))$$

Where ϕ, ψ are Multi-Layer Perceptrons (MLP), and $\oplus$ is the permutation-invariant local neighborhood aggregation function (sum, mean, min, max or mul). For my project I experimented with each of the aggregation functions and found mean to perform the best. For each pair of nodes we compute the message $m_{ij} = \psi(h_i^\ell, h_j^\ell)$, where the MLP $\psi = R^{2d} \rightarrow R^d$ takes the concatenation of our feature vectors and maps it to a single set of features. Similarly the node update function concatenates the aggregated ψ output and pass it through the MLP $\phi : R^{2d} \rightarrow R^d$ The aggregate function we utilise, denoted by the permutation-invariant function $\oplus$ is the mean i.e $\oplus_{j \in R_i} = \frac{\sum_{j \in N_i}}{N}$

Hyper Parameters: I experimented with very different variations of hyper parameters. Tuning to minimise training loss produced the best development set results, preventing overfitting with lower hidden channels and lower layers with a greater number of epochs.
$Layers$: 2
$Hidden\ Channels$: 32
$Edge\ Dimensions$: 2
$Epochs$: 150
$Learning\ Rate$: $1e - 5$

Graph Attention Neural Network

Graph Attention Layer: To increase performance over convolutional networks I intended to use self-attentional layers to allow nodes to focus on neighbourhood features. I formulate the attentional layer as follows, for each pair of nodes i, j the attention mechanism a computes the coefficient e_{ij} by each nodes features h. $e_{ij} = a(h_i, h_j)$. The mechanism functions by attending only to the nodes in node $i's$ neighborhood $j \in N_i$. The coefficient is then normalised through a softmax function:

$$a_{ij} = \frac{exp(e_{ij})}{\sum_{k \in N_i} exp(e_{ik})}$$

Development Set Results	
System	**OFEVER**
GENRE	0.8920
FEVER Baseline	0.7020
UKP Athene	**0.9355**

Table 2: Oracle accuracy score for 3 high performing document retrieval systems

Graph Structure: The graph network is composed of 3 graph attentional layers each one with 256 hidden channels which pass through a relu activation and dropout layer upon completion applying a final dropout and global mean pooling.

4 RESULTS AND ANALYSIS

4.1 Document Retrieval

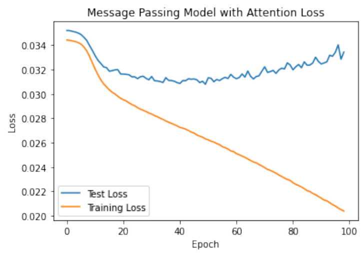
For analysing the results of the document retrieval system, I used the oracle accuracy (OFEVER) score, defined in the original 2018 shared task paper. This requires extracting the k (k=5) top documents using GENRE and only if the documents match all 5 pieces of evidence with a single claim then it can be considered correct, otherwise it is incorrect.
The results listed in Table 2 show that the performance actually was underwhelming when compared to current state of the art document retrieval systems. The belief for this is that GENRE is pre-trained and non-discriminate in its Wikipedia data output which results in a large number of documents being retrieved that do not exist in the original FEVER Wikipedia set. This is highly beneficial when producing for a real world system where we do not wish to limit the amount of Wikipedia documents we have access to but results in less than 90% accuracy on our development set.
GENRE's speed and performance is very high and therefore one technique that could be applied in the future if this project were to be continued would be to produce our own pre-trained fairseq trie for GENRE that only utilises the original FEVER shared task Wikipedia dataset.

4.2 Sentence Selection

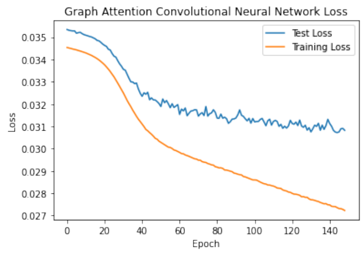
For the sentence selection we fine-tuned 2 different pre-trained language models, BERT and RoBERTa. RoBERTa outperformed BERT in every category after training on the train set for 5 epochs. Table 2 shows the performance of the RoBERTa model

Threshold	OFEVER	Precision	Recall	F1
0	**0.6558**	0.1257	**0.4836**	0.1996
0.1	0.6521	0.2365	0.4781	0.3165
0.5	0.6504	0.2547	0.4755	0.3317
0.8	0.6492	0.2699	0.4737	0.3439
0.9	0.6469	0.2918	0.4703	0.3601
0.95	0.6421	0.3398	0.4631	0.3920
0.98	0.6288	0.4426	0.4431	0.4429
0.99	0.5979	**0.5999**	0.3968	**0.4776**

Table 3: Sentence Selection performance with varying thresholds on probability outputs



(a) Message Passing Neural Network Loss



(b) Graph Attention Neural Network Loss

Figure 1: Training and testing loss of both models after 100 epochs

with thresholding applied to the probabilities for each evidential sentence output by the model. Since FEVER considers a piece of evidence complete if the predicted evidence sentences match all evidences for a given claim it makes sense not to apply thresholding when attempting to maximise FEVER score. Although in maximising precision accuracy, higher thresholds show better output.

4.3 Claim Verification

The two models performed quite differently over 100 epochs, the graph attention model produced better overall results as it was less prone to overfitting than the MPNN model. This can be noted in Figure 1. The Message Passing Neural Network overfits after 35 epochs when test loss begins to increase. This could be due to the smaller dataset (20,000 instead of 140,000 claims), or the fact that each message propogation passes through two MLP's for each node neighbour. The lack of edge features also play a factor and could be addressed by randomly initialising edge features or in the future defining edge features as a cosine similarity score between two evidence nodes.
The Graph Attention Network quickly increased testing performance until 40 epochs when it begins to stagnate although loss still decreases upto 100 epochs. This could be due to the lower number of layers and low amount of hidden layers in training which helped prevent overfitting and was aided by the lower learning rate hyperparameter.

5 DISCUSSION

Overall what I have seen from this project is the application of graph neural networks in the field of Natural Language Processing is an exciting and still evolving area of discovery. With my work I have seen close to state of the art results in the task of claim verification with the possibility for very serious expansion to true state of the art performance.
Some of the possible ways to achieve that I have catalogued during development of this project as things I would implement given more time. The first being experimenting with different sentence encoding models to compare performance for graph neural networks. Models which encode vector space better should perform better in message passing as it can enable aggregation between nodes to alter the vector space in a way that better defines the combination of evidence sentences.
Another area of interest for me was attempting to define methods for including edge features without increasing overfitting in the original model. Attempts with defining random edge features with oversampling nodes actually decreased performance. As discussed earlier I believe cosine similarity between nodes or word matching as done in GEAR's ERNET graph model could prove to be beneficial additional features.
I would also wish if given more time to use a word embedding model such as FastText to define the vector space of Wikipedia document names, this could then be utilised to define a distance between nodes as their vector distance between document embeddings.

Model	Accuracy	Recall	Precision	F1
Graph Attention Model	0.6120	0.6120	0.6120	0.6119
Message Passing Model	0.5160	0.5160	0.5158	0.5159
GEAR (ERNet)	0.6744	0.6744	0.6744	0.6743
FEVER Baseline	0.4892	0.4900	0.4873	0.4889

Table 4: Claim Verification on Development set between sample of FEVER models

The performance of both models exceeded initial expectations although the document retrieval stage of the pipeline performed sub-standard to the legacy FEVER state of the art system by UKP Athene which was trained specifically on the FEVER Wikipedia dataset. This could be one other topic of interest in the future, experiment training GENRE's end-to-end entity linking model on FEVER's Wikipedia set.

6 CONCLUSION

In this paper I proposed two Graph-based models. A Message Passing Neural Network model and A Graph Attention Network, for the sub-task of claim verification for the FEVER shared task. I made use of pre-trained language transformer models for Sentence Selection and evaluated their OFEVER performance metric. Utilising GENRE for Document Retrieval which was found to yield poorer than expected results when applied specifically to the full FEVER task. Overall our pipeline proved an efficient and new entry to the FEVER shared task list which outperforms the baseline FEVER model and introduces new techniques for applying Graph Neural Networks to the field of fact-checking that only cement the fact that the future of evidence reasoning is a graph based problem.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive entity retrieval.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. 2018. UKP-athene: Multi-sentence textual entailment for claim verification. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 103–108, Brussels, Belgium. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2018. Combining fact extraction and verification with neural semantic matching networks.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases?

Shahzad Qaiser and Ramsha Ali. 2018. Text mining: Use of tf-idf to examine the relevance of words to documents. *International Journal of Computer Applications*, 181.

Amir Soleimani, Christof Monz, and Marcel Worring. 2020. Bert for evidence retrieval and claim verification. In *Advances in Information Retrieval*, pages 359–366, Cham. Springer International Publishing.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018. The fact extraction and VERification (FEVER) shared task. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 1–9, Brussels, Belgium. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery &amp Data Mining*. ACM.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. 2019. Graph transformer networks.

Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2019. Gear: Graph-based evidence aggregating and reasoning for fact verification.