Applying Recurrent Layers to the Decomposable Attention Model for Natural Language Inference

Ryan Camilleri

Department of Artificial Intelligence

University of Malta

Msida, Malta

ryan.camilleri.18@um.edu.mt

Index Terms

Natural Language Inference, Textual Entailment, Attention Models, Recurrent Neural Networks

I. Introduction

Natural Language Inference (NLI) also referred to as Recognising Textual Entailment (RTE), "*is the problem of determining whether a natural language hypothesis h can reasonably be inferred from a natural language premise p*" [1]. The ability to determine whether a pair of sentences share no relation, are contradictory or entail each other is core to many applications such as those based on information retrieval, multi-document summarisation and machine translation, to name a few. Previous solutions to RTE relied heavily on feature engineering and specialised components [2]–[4], as the lack of large datasets greatly limited the performance of end-to-end neural architectures. Using these models not only meant that specific assumptions about the underlying language could be avoided, but these could also allow for better language representation and more complex relationships among sequences. Fortunately, with the publishing of the Stanford Natural Language Inference (SNLI) corpus [5], enough data was present for research to focus on neural models instead of hand-engineered classifiers. Accompanied by a long short term memory network (LSTM) [6], Bowman et al. confirm the viability of their SNLI dataset to be used with neural models without the need for extensive feature engineering.

A. Problem Definition

Following Bowman et al.'s efforts, other RTE datasets have been proposed for facilitating the use of neural architectures such as the MNLI [7] and the XNLI [8] corpora. Despite these, the utility of the SNLI dataset has not diminished, as state-of-the-art models continue using the dataset as a benchmark, with the most recent achieving a 92.1% accuracy on the test set [9]. Unfortunately, to attain such results, recent studies have favoured the use of large, composite architectures which depend on a significant number of parameters, much larger than earlier solutions. Despite being able to represent deeper and more complex relationships, having millions of parameters impacts training and inference times considerably, and is also very inefficient in terms of memory.

B. Motivation

Past machine translation solutions have applied encoder-decoder models which suffered in handling large sentences due to their inability to compress all information into a fixed-length vector [10]. To address this issue, Bahdanau et al. propose their attention mechanism [11], which builds upon the concept of pixel-based attention introduced by Larochelle and Hinton in computer vision [12]. Bahdanau et al.'s extension makes use of soft-alignment which searches a source sentence to determine where relevant information is concentrated, based on the word being generated. Results show that by using this approach, the solution is able to choose a subset of context vectors and focus solely on richer information instead of all information. Utilising this, prior RTE solutions have managed to achieve unprecedented results as using attention meant that a highly efficient weight distribution could be computed over the input sequences whilst using simpler architectures [13]. Parikh et al.'s decomposable attention model (DAM) demonstrates this perfectly, as by utilising solely feed-forward networks, their model achieves accuracies of 86.3% on the SNLI test set, with just 380k parameters [14]. Solutions following the DAM have secured better results by utilising recurrent architectures at the cost of more parameters. Wang et al.'s bilateral multi-perspective matching (BiMPM) model [15] is a great example of this, which uses the matching-aggregation framework [16] to achieve 87.5% using 1.6m parameters - the least amount of parameters used for recurrent networks following DAM.

C. Aims and Objectives

The aim of this report is to investigate whether adding recurrent components to the decomposable attention model [14] could enhance the performance of the overall network. Additionally, the proposed enhancements will be strict in maintaining a lower amount of parameters than those required for the BiMPM model [15], so as to determine if an increase in performance could be achieved with minimal increase in parameters. This will be accomplished by fulfilling the following objectives:

- **Objective (O1)**: Construct the DAM and modify specific layers to include recurrent components.
- **Objective(O2)**: Using Parikh et al.'s model as a baseline, evaluate the accuracies achieved by the models, to determine their over all effectiveness.

II. Literature Review

This section aims to give an overview of neural solutions applying the attention mechanism, highlighting specific aspects such as the techniques used and the parameters required excluding embeddings. For completeness, the section concludes with a brief review of state-of-the-art solutions applying transformers.

A. *Attention-based Models*

To evaluate the efficacy of their newly proposed SNLI dataset, Bowman et al. apply three separate models to their corpus [5]. Results show that the simple sequence embedding model with an LSTM component achieved the best performance out of the three. Due to its robust ability of learning long-term dependencies, Rocktäschel et al. follow on Bowman et al.'s work by applying their own LSTM model [17]. Whereas Bowman et al. encode the premise and hypothesis as dense fixed-length vectors, Rocktäschel et al. propose an attentive network capable of reasoning over entailments by processing the hypothesis over the premise. The authors achieve this by applying word-by-word attention similar to that done by Bahdanau et al. [11]. Using just 250k trainable parameters, the model proves to be the first generic end-to-end system which outperforms the best neural and hand-engineered baselines, at the time.

Liu et al. argue however, that single directional LSTMs and gated recurrent units (GRUs) suffer in utilising contextual information from future tokens [18]. As an alternative, the authors propose a sentence encoding-based model which uses a bidirectional LSTM (BiLSTM), to process sequences in two directions, thereby maintaining previous and future contexts more efficiently. A unique approach to attention is also developed; Instead of using the target sentence to attend words in the source sentence, the authors apply their 'inner-attention' to attend words appearing in itself [18]. At the expense of having 2.8m parameters, their model proves superior to Rocktäschel et al.'s solution. In contrast to Liu et al.'s claims, Wang and Jiang prove through their matchLSTM (mLSTM) that bidrirectionality is not always essential [19]. The authors achieve better accuracies with less parameters (1.9m) by utilising word-by-word matching to process sequences sequentially and match the current word with an attention weighted representation of the target sentence. Cheng et al. improve on these findings through their LSTMN, by embedding an LSTM with a memory network for storing contextual representations of input tokens [20]. Coupling this with a form of intra-attention, the authors manage to achieve 86.3% on the test set, setting it above all prior models at the expense of using 3.4m parameters.

As it stands, research has progressed into favouring more complex architectures in order to achieve better accuracies. Parikh et al. notice such trends and argue that NLI does not require deep modeling of sentence structures [14]. Instead, the authors propose their lightweight decomposable attention model (DAM), which only makes use of feed-forward networks and attention to achieve comparable results. Extending their model with Cheng et al.'s intra-sentence attention [20], DAM manages to outperform LSTMN using just 580k parameters. With these results, the authors conclude that pairwise comparisons are relatively more important than global sentence-level representations. Despite their efforts, research still favours more complex systems as these are shown to achieve better accuracies. Among such models succeeding the DAM, the BiMPM model manages to achieve 87.5% against DAM's 86.8% using just 1.6m parameters [15]. Even though the number of parameters is much larger than those used in Parikh et al.'s solution, when compared to other networks, the amount seems minuscule. With inspiration from Wang and Jiang's mLSTM, the authors base their model on the matching-aggregation framework, which matches each timestep of one sentence against all timesteps of the target sentence and aggregates the result into a fixed-length vector. Wang et al. also explore different matching techniques such as phrase-by-sentence, instead of just using solely word-by-word matching. Contradictory to Wang and Jiang, the authors are shown to agree with Liu et al.'s statements and describe the neglection of matching in the reverse direction as a limitation. To address this, their model revisits the use of BiLSTMS and applies these at different layers of the network [15].

B. State-of-the-art Models

The reviewed solutions thus far, all make use of some sort of word embedding as this is a classic approach to representing text in vector format. Since this assigns one specific vector to each word, the approach is not robust in handling word polysemy as words may have more than one meaning. Contextualised/Dynamic Word Embeddings such as ELMo [21] overcome this limitation by using BiLSTMs to look at the entire sentence and assign an embedding to each word, based on context. To obtain better performances, Vaswani et al. introduce transformers and demonstrate how sequence models like LSTM could be replaced by attention mechanisms entirely [22]. The creation of these tools facilitated Devlin et al.'s BERT [23], which combines ELMo and several transformers to assign unique vectors to words based on their context. Utilising such transformers, the state-of-the-art maintains its lead above all other solutions at the expense of millions of parameters [9], [24], [25].

III. Methodology

The previous section gave a brief review of models applying attention whilst maintaining a small amount of parameters compared to the current state-of-the-art. This section will describe the SNLI dataset used and provide information on its distribution. Moreover, a description of the models implemented and their architecture will also be given. Aspects such as the embeddings used and how text was transformed into sequences will also be described.

A. SNLI Dataset

This paper makes use of the SNLI dataset [5] as this has been widely used and is among the top-leading corpora for NLI. The corpus is readily split into training, validation and test sets consisting of 550152, 10000 and 10000 sentence pairs respectively. For each pair, a gold label is defined which specifies the relationship between them. This label would represent the majority vote by the annotators' judgements.

1) Dataset Distribution and Filtering: When going through the dataset, some instances were found in which a single majority couldn't be determined. In such cases, a ' - ' character was found to be the placeholder for the gold label. Since the main focus of this paper is to evaluate the models solely on (i) contradiction, (ii) neutrality and (iii) entailment, this had to be addressed. The first option was to override the placeholder with a random label from the two that got equal scores. The downside to this however, is that a specific class may be given a higher majority which would effect label frequency and overall prediction results. Avoiding this, such instances were disregarded completely in each set so that the final training, validation and test sets would now have 549367, 9842 and 9824 sentence pairs respectively. As a further check, the sets were also verified to have equal number of premise and hypothesis sentences for each label.

Observing label distribution following the above filtering, it is noted that the labels in each set do not occur in equal frequencies. Despite this, the frequencies do not vary by extreme amounts and so, the models would still be able to recognise each class fairly. It is assumed that results shown on the SNLI board were achieved using the full imbalanced dataset and so, data stratification was not applied.

B. Feature Engineering

Since neural networks require numerical inputs, feature engineering was applied to convert both the text and labels into the appropriate format.

1) Label Indexing: Labels could either be represented using one-hot encoded vectors or by decimal indexes. Even though the former is usually preferred to avoid bias in network predictions, such bias usually exists when a substantial amount of classes are present for classification. Since RTE is a three-class problem, indexing is used for its simplicity and easier handling of model outputs.

2) Text to Sequence: Before replacing words into their token equivalents, the texts were all sanitised by removing punctuation and setting everything to lower case. This was ensured to avoid the processing of punctuation and avoid setting different tokens to the same words having varied character cases. Using Tensorflow's tokeniser, a word index was generated when applied to the concatenated training and validation texts. The test set wasn't included as training is not done on this. However, when applying the tokeniser to the test set for evaluation, unseen words are handled using an out-of-vocabulary (OOV) token. Using the tokeniser, sentences are converted into sequences with each word having its unique index. This is not enough however as neural networks also require that inputs have the same shape. Therefore, a max sequence length was determined and zeros were appended to the sequence vectors until their length matches the max. If the sequence contains more words than the max, the additional words are truncated from the end of the sequence. Initially, the max length was defined to be equal to the largest sentence in the dataset which as found to be 78 words. However, since the majority of sentences contained much less, the max length was defined as 50, to find a balance between reducing the sparsity of the short sequences whilst avoiding a big losses in information from larger sequences.

3) Embeddings: To avoid training embedding layers and adding to the trainable parameter count, pre-trained embeddings were used. An additional benefit to using these is that better word representations are ensured due to larger contexts. The gensim data word2vec-google-news-300 was chosen as this contains information on 100 billion words [26].

C. Models

1) Model Selection: As reviewed, Parikh et al's DAM improves greatly on earlier solutions by achieving the same exact results as the state-of-the-art at the time, using only 380k parameters [14]. As an extension, the authors also apply intra-sentence attention [20] to bolster the accuracies of their model using slightly more parameters (580k). Since this paper aims to encourage more lightweight architectures, the original DAM will be implemented along with its intra-sentence attention equivalent. Another aim of this paper is to investigate whether recurrent components could enhance the results achieved by DAM. With inspiration from Wang et al.'s BiMPM [15], two further models will be defined, one making use of BiLSTMs and the other with Bilateral gated recurrent units (BiGRU). GRU is used for the fourth model as these are computationally more efficient than LSTMs, since no memory units are used.

2) Model Implementations: Each of the 4 models were implemented using Keras with Tensorflow, and the final architectures for each can be observed in Section A of the Appendix. Using this framework, the necessary layers were defined to model the DAM architecture composed of 3 main steps. The first step called **Attend**, finds word-level alignments between sentences by computing pairwise similarity and soft-alignment. To compute pairwise similarity, the embedded representations of sequences are passed through a feed-forward network (FFN) to yield non-linear transformations. To achieve similar results, the FFN was defined to be equal to that used by Parikh et al., which uses 2 hidden layers each with 200 nodes and ReLU activations. Dropout layers with a dropout rate of 0.1, were also added in-between layers to regularise the network and help prevent over-fitting. Vector similarities are then computed using the non-linear outputs and passing these through a dot product layer. To carry out soft-alignment, the similarities are normalised and the result along with the target embeddings are put through a further dot product layer. The results from these generate α which is the sub-phrase in the premise that is softly aligned with the hypotheses and vice versa for β. The second **Compare** step, concatenates the previous results with the embeddings of the target sequences and passes the concatenated result through a further FFN to achieve the comparison vectors *v1* and *v2*. The third **Aggregate** step aggregates each of the vectors and concatenates both results together. Lastly, A final FFN receives the concatenated result as input and applies its output to a dense layer of 3 nodes with softmax activation for outputting the model's predictions. The demonstrated network uses 381,803 parameters which is equivalent to those used by the original DAM.

As an extension, intra-sentence attention was appended to the above model prior to the Attend step. The same exact process as that done in the first step is repeated, with some modifications. Instead of aligning vectors with their target sequences, the vectors are aligned with themselves. Additionally, distance sensitive bias terms are also generated and a separate normalisation function was defined to utilise such distance terms. After the intra-sentence attention step, the resulting self-aligned vectors are concatenated with the non-aligned embeddings and used as input to the Attend layer. The resulting DAM-INTRA model uses 542,203 parameters which is smaller than the 580k used in the original paper. The DAM-BiLSTM and DAM-BiGRU models were simply defined to replace the FFN within the compare layer with their Bilateral equivalents, to achieve higher dimensional comparison vectors. To avoid potential loss in vital memory units of LSTM or GRU cells, recurrent dropout was not applied in this solution. By using such recurrent components and applying bidirectionality, the number of trainable parameters increase to 1,583,003 and 1,263,803 parameters for the DAM-BiLSTM and DAM-BiGRU respectively.

3) Training and Hyperparameters: Each model was compiled similarly using sparse categorical cross entropy and the Adam optimiser. The reason for using the specific loss type was due to the fact that labels were indexed instead of one-hot encoded. After testing the Adagrad, Adam, and Adamax optimisers on the DAM, the best results were achieved using Adam and so, this was applied to all models. To retrieve the best model version during training, Tensorflow's EarlyStopping and ModelCheckpointer callbacks are used. By monitoring the validation loss, the weights of the best model are automatically saved when no improvement is detected after 5 iterations. For training, models were fit to a batch size of 512 and training was performed over 50 epochs. Even though this number might seem large, no model managed to train the full epoch amount since early stopping was utilised.

When training the DAM-INTRA model, results were observed to steadily improve in accuracy with each epoch. Around the sixth epoch mark however, the model got exceedingly worse and accuracies fell significantly. This sudden decrease in accuracy may be the cause for the network getting knocked into a local minimal or poor parameter space. From this position, the model was indeed recovering slowly but was unfortunately interrupted, due to the early stopping callback. To account for this, the learning rate for all models was set to 0.0005 instead of the default 0.001 for the Adam optimiser. An alternative was also identified to dynamically update the learning rate through a scheduler, based on the current epoch. This was not done however as satisfactory results were achieved through the first approach.

IV. Results and Evaluation

Following the previous section which describes the implementation of the DAM, DAM-INTRA, DAM-BiLSTM and DAM-BiGRU models, a quantitative and qualitative evaluation of the models is provided by this section. Through these, the strengths and weaknesses of each are determined, and these are also compared to Parikh et al.'s model [14], used as a baseline. This section is also responsible for determining the usefulness of applying recurrent components to DAM, and providing a short error analysis of the results.

Model	Params	Train	Dev	Test
DAM (Parikh et al.)	380k	89.5	-	86.3
DAM-INTRA (Parikh et al.)	580k	90.5	-	86.8
DAM	381k	86.4	85.0	84.8
DAM-INTRA	542k	86.6	85.0	84.5
DAM-BiLSTM	1.58m	88.1	85.2	84.5
DAM-BiGRU	1.26m	88.4	85.5	84.7

TABLE I: Model results on the SNLI corpus [5].

A. SNLI Results

Baseline results, as well as the results achieved by each model after applying these to the SNLI dataset, can be seen in Table I. As shown, no model is observed to surpass or match Parikh et al.'s results. A possible cause to this, may be due to the small amount of patience attributed to the early stopping mechanism. Even though some models could be further trained to yield better results, the preventative over-fitting measure stopped the models before possibly converging to more optimal weights. It is also possible that the networks got stuck in local optima which prevent improvements. By comparing the custom models to themselves, DAM is shown to achieve superior results on the test set. DAM-BiGRU came in a close second, having just 0.1 less score than DAM's result, with DAM-INTRA and DAM-BiLSTM at last. The accuracies suggest that adding recurrent components to the DAM is not very effective and that other modifications to DAM's architecture should be explored. It is also surprising that adding intra-sentence attention did not yield better results and instead, reduced the accuracy achieved by the model. A reason for this may be related to the distance sensitive bias terms, which may not have been set equally to those of Parikh et al. during the intra attention step.

B. Multi-Class Results

When carrying out class evaluation, it is important to note that the frequency of classes is not the same, with the (i) *entailment*, (ii) *neutral* and (iii) *contradiction* classes having 3368, 3219 and 3237 instances respectively. The strengths and weaknesses of each model on specific classes is determined, by finding the precision, recall and F-measure for each class. Such values along with the mean average precision (mAP) of the models can be seen in Table II. Multi-class confusion matrices, ROC and precision-recall curves are also included within the Appendix, should the reader require further observations. Interestingly, the models manage to achieve the

Model	Class	Precision	Recall	F-Measure	mAP
DAM	entailment	0.86	0.89	0.87	0.85
	neutral	0.80	0.80	0.80	
	contradiction	0.88	0.86	0.87	
DAM-INTRA	entailment	0.85	0.89	0.87	0.85
	neutral	0.81	0.79	0.80	
	contradiction	0.87	0.86	0.86	
DAM-BiLSTM	entailment	0.86	0.88	0.87	0.85
	neutral	0.80	0.81	0.80	
	contradiction	0.88	0.85	0.86	
DAM-BiGRU	entailment	0.84	0.90	0.87	0.85
	neutral	0.81	0.79	0.80	
	contradiction	0.89	0.84	0.87	

TABLE II: Class results for each model.

same mAP throughout and have very similar class F-measures showing that the models do not vary by a significant amount from each other. When observing the precision and recall values however, some diversities are distinguished. Some models achieve higher precision and recall in specific classes when compared the other models. DAM and DAM-BiLSTM have the most accurate predictions for (i), DAM-INTRA and DAM-BiGRU for (ii), and DAM-BiGRU for (iii). In terms of recall, DAM-BiGRU manages to identify most instances of (i). DAM-BiLSTM for class (ii) and DAM and DAM-INTRA for (iii).

C. Error Analysis

Through manual analysis of incorrect predictions, a better idea as to why the models made the wrong choices could be understood. This paper carries analysis only on the DAM model, as this achieved the best results. Analysis is also carried using a single instance for each class, shown in Table III. Should the reader require visualisation of further incorrect predictions by other models, with the inclusion of softmax values, the reader should refer to the attached Jupyter notebooks. When comparing the premise and hypothesis of result 1, it unclear as to why the model predicts contradiction when the sentences are in a clear neutral relationship. When focusing on each sentence however, a possible explanation to this may be related to the model's inability to determine a correlation between the terms 'manuals' and 'books'. Should the model utilise further context when working with vectors, the prediction may be corrected. The 2nd result highlights the limitation of the model, in that it is incapable of inferring that whenever a

Result	Premise and Hypothesis	Gold Label	Prediction
1	A blond-haired doctor and her African american assistant looking threw new medical manuals. A doctor is looking at a book	neutral	contradiction
2	A woman with a green headscarf, blue shirt and a very big grin. The woman has been shot.	contradiction	neutral
3	An old man with a package poses in front of an advertisement. A man walks by an ad.	contradiction	entailment

TABLE III: Incorrect predictions made by the DAM.

person is shot, most often, that person would not be grinning. The prediction stands for debate, as technically, even if a person is shot, a fake grin could still be made. However, as described for the 1st result, context awareness must be adopted for the model to improve. In the last result, the term ’ad’ is successfully correlated with ’advertisement’ thereby predicting entailment. The model fails however, in determining the variance between ’walks’ and ’poses’, resulting in the incorrect prediction.

V. Conclusion

This study investigates the effectiveness of adding recurrent components to Parikh et al.’s Decomposable Attention Model (DAM) [14] whilst trying to maintain a lightweight solution. Four models were defined, as extensions to the original DAM. One of the models was defined to represent exactly that from the original paper. The three others extend this model to include intra-sentence attention (DAM-INTRA), a bilateral LSTM (DAM-BiLSTM) and a bilateral GRU (DAM-BiGRU) respectively. Applying the models to the SNLI dataset, the original DAM achieves superior results with DAM-BiGRU coming in a close second. None of the implemented models manage to surpass the results achieved by the original solutions and no benefit was determined when adding recurrent layers. To further improve on these findings, this paper encourages future research to explore alternate means on how to incorporate recurrent networks better to the DAM. Additionally, to have better contextual vectors, pre-trained transformers could be used. When training, the models were observed to stop early due to the satisfying of early stopping callback conditions. In order to ensure model convergence, the patience could be increased to a higher amount. Lastly, this paper also encourages research to experiment with text summarisation, before processing data; As this may yield better results, even to state-of-the-art solutions.

References

[1] B. MacCartney and C. D. Manning, “Natural logic and natural language inference,” p. 18.

[2] O. Glickman and I. Dagan, “Web Based Probabilistic Textual Entailment,” *Proceedings of the First Challenge Workshop Recognising Textual Entailment*, Jan. 2005.

[3] B. MacCartney, M. Galley, and C. Manning, “A Phrase-Based Alignment Model for Natural Language Inference.,” pp. 802–811, Jan. 2008.

[4] Y. Mehdad, R. Moschitti, and F. M. Zanzotto, “SemKer: Syntactic/semantic kernels for recognizing textual entailment,” in *In Proc. of the Text Analysis Conference*, 2009.

[5] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, “A large annotated corpus for learning natural language inference,” *arXiv:1508.05326 [cs]*, Aug. 2015. arXiv: 1508.05326.

[6] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, pp. 1735–1780, Nov. 1997.

[7] A. Williams, N. Nangia, and S. Bowman, “A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, (New Orleans, Louisiana), pp. 1112–1122, Association for Computational Linguistics, June 2018.

[8] A. Conneau, G. Lample, R. Rinott, A. Williams, S. R. Bowman, H. Schwenk, and V. Stoyanov, “XNLI: Evaluating Cross-lingual Sentence Representations,” *arXiv:1809.05053 [cs]*, Sept. 2018. arXiv: 1809.05053.

[9] J. Pilault, A. Elhattami, and C. Pal, “Conditionally Adaptive Multi-Task Learning: Improving Transfer Learning in NLP Using Fewer Parameters & Less Data,” *arXiv:2009.09139 [cs, stat]*, Sept. 2020. arXiv: 2009.09139.

[10] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, “On the Properties of Neural Machine Translation: Encoder-Decoder Approaches,” *arXiv:1409.1259 [cs, stat]*, Oct. 2014. arXiv: 1409.1259.

[11] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate,” *arXiv:1409.0473 [cs, stat]*, May 2016. arXiv: 1409.0473.

[12] H. Larochelle and G. Hinton, “Learning to combine foveal glimpses with a third-order Boltzmann machine,” vol. 1, pp. 1243–1251, Jan. 2010.

[13] A. Galassi, M. Lippi, and P. Torroni, “Attention in Natural Language Processing,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–18, 2020. arXiv: 1902.02181.

[14] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, “A Decomposable Attention Model for Natural Language Inference,” *arXiv:1606.01933 [cs]*, June 2016. arXiv: 1606.01933 version: 1.

[15] Z. Wang, W. Hamza, and R. Florian, “Bilateral Multi-Perspective Matching for Natural Language Sentences,” *arXiv:1702.03814 [cs]*, July 2017. arXiv: 1702.03814.

[16] S. Wang and J. Jiang, “A Compare-Aggregate Model for Matching Text Sequences,” *arXiv:1611.01747 [cs]*, Nov. 2016. arXiv: 1611.01747.

[17] T. Rocktäschel, E. Grefenstette, K. M. Hermann, T. Kočiský, and P. Blunsom, “Reasoning about Entailment with Neural Attention,” *arXiv:1509.06664 [cs]*, Sept. 2015. arXiv: 1509.06664 version: 1.

[18] Y. Liu, C. Sun, L. Lin, and X. Wang, “Learning Natural Language Inference using Bidirectional LSTM model and Inner-Attention,” *arXiv:1605.09090 [cs]*, May 2016. arXiv: 1605.09090 version: 1.

[19] S. Wang and J. Jiang, “Learning Natural Language Inference with LSTM,” *arXiv:1512.08849 [cs]*, Dec. 2015. arXiv: 1512.08849 version: 1.

[20] J. Cheng, L. Dong, and M. Lapata, “Long Short-Term Memory-Networks for Machine Reading,” *arXiv:1601.06733 [cs]*, Sept. 2016. arXiv: 1601.06733.

[21] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” *arXiv:1802.05365 [cs]*, Mar. 2018. arXiv: 1802.05365.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention Is All You Need,” *arXiv:1706.03762 [cs]*, Dec. 2017. arXiv: 1706.03762.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” *arXiv:1810.04805 [cs]*, May 2019. arXiv: 1810.04805.

[24] X. Liu, P. He, W. Chen, and J. Gao, “Multi-Task Deep Neural Networks for Natural Language Understanding,” *arXiv:1901.11504 [cs]*, May 2019. arXiv: 1901.11504.

[25] Z. Zhang, Y. Wu, H. Zhao, Z. Li, S. Zhang, X. Zhou, and X. Zhou, “Semantics-aware BERT for Language Understanding,” *arXiv:1909.02209 [cs]*, Feb. 2020. arXiv: 1909.02209.

[26] “RaRe-Technologies/gensim-data,” Feb. 2021. original-date: 2017-10-13T18:22:15Z.

APPENDIX

A. Model Architectures

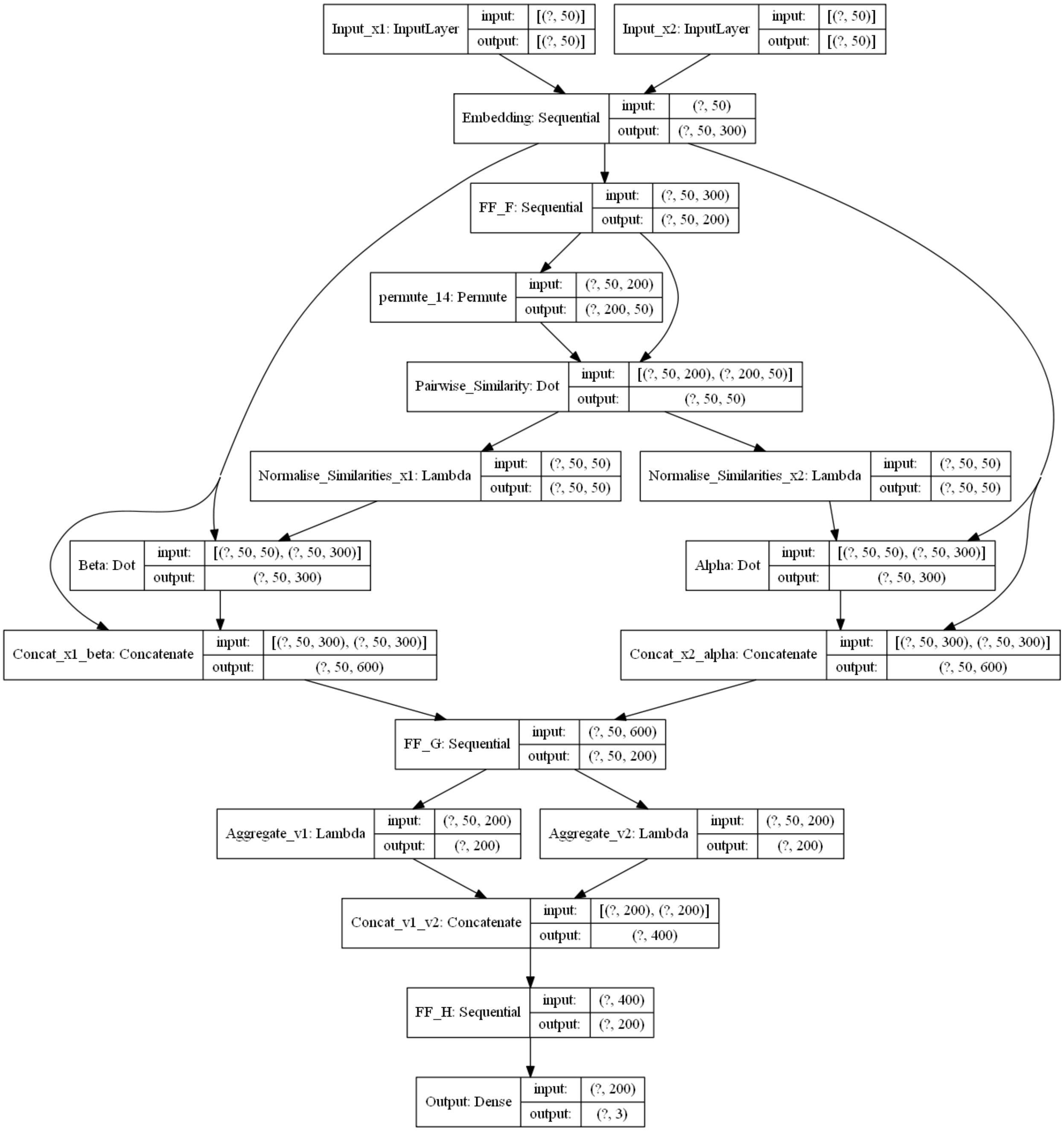

Input_x1: InputLayer | input: | [(?, 50)] | output: | [(?, 50)]
Input_x2: InputLayer | input: | [(?, 50)] | output: | [(?, 50)]
Embedding: Sequential | input: | (?, 50) | output: | (?, 50, 300)
FF_F: Sequential | input: | (?, 50, 300) | output: | (?, 50, 200)
permute_14: Permute | input: | (?, 50, 200) | output: | (?, 200, 50)
Pairwise_Similarity: Dot | input: | [(?, 50, 200), (?, 200, 50)] | output: | (?, 50, 50)
Normalise_Similarities_x1: Lambda | input: | (?, 50, 50) | output: | (?, 50, 50)
Normalise_Similarities_x2: Lambda | input: | (?, 50, 50) | output: | (?, 50, 50)
Beta: Dot | input: | [(?, 50, 50), (?, 50, 300)] | output: | (?, 50, 300)
Alpha: Dot | input: | [(?, 50, 50), (?, 50, 300)] | output: | (?, 50, 300)
Concat_x1_beta: Concatenate | input: | [(?, 50, 300), (?, 50, 300)] | output: | (?, 50, 600)
Concat_x2_alpha: Concatenate | input: | [(?, 50, 300), (?, 50, 300)] | output: | (?, 50, 600)
FF_G: Sequential | input: | (?, 50, 600) | output: | (?, 50, 200)
Aggregate_v1: Lambda | input: | (?, 50, 200) | output: | (?, 200)
Aggregate_v2: Lambda | input: | (?, 50, 200) | output: | (?, 200)
Concat_v1_v2: Concatenate | input: | [(?, 200), (?, 200)] | output: | (?, 400)
FF_H: Sequential | input: | (?, 400) | output: | (?, 200)
Output: Dense | input: | (?, 200) | output: | (?, 3)

Fig. 1: DAM Architecture.

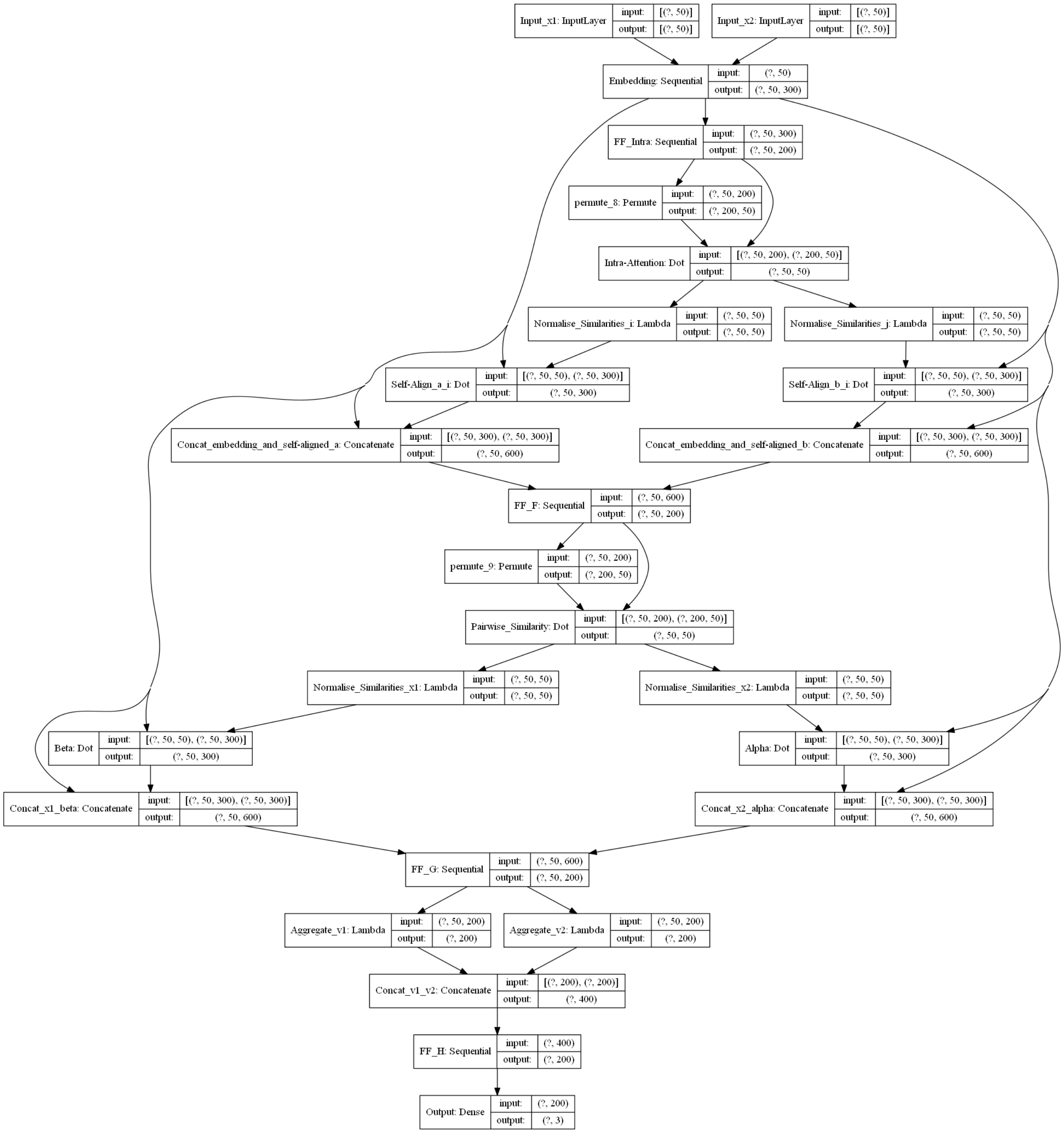


Fig. 2: DAM-INTRA Architecture.

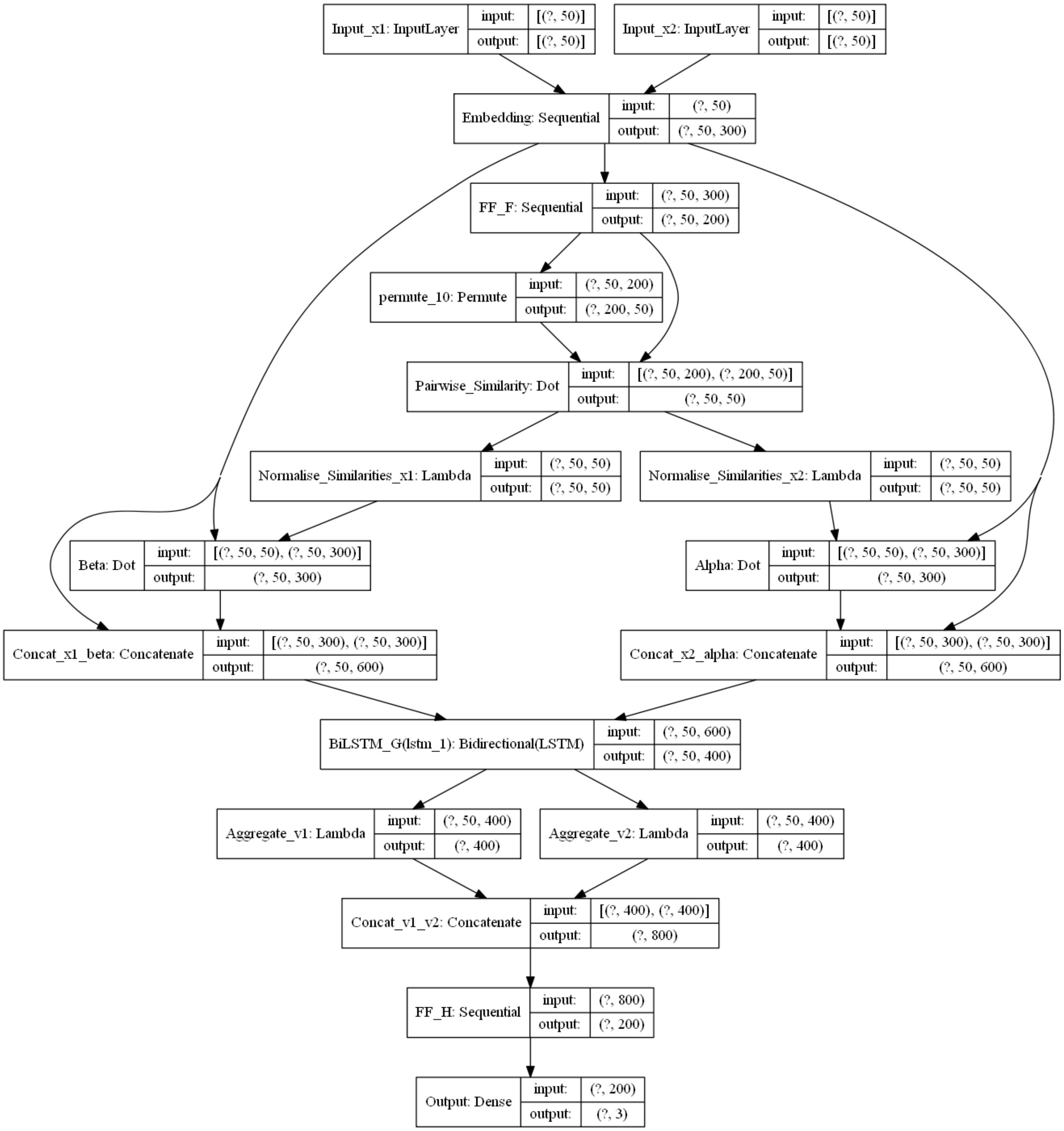

Input_x1: InputLayer | input: | [(?, 50)] | output: | [(?, 50)]
Input_x2: InputLayer | input: | [(?, 50)] | output: | [(?, 50)]
Embedding: Sequential | input: | (?, 50) | output: | (?, 50, 300)
FF_F: Sequential | input: | (?, 50, 300) | output: | (?, 50, 200)
permute_10: Permute | input: | (?, 50, 200) | output: | (?, 200, 50)
Pairwise_Similarity: Dot | input: | [(?, 50, 200), (?, 200, 50)] | output: | (?, 50, 50)
Normalise_Similarities_x1: Lambda | input: | (?, 50, 50) | output: | (?, 50, 50)
Normalise_Similarities_x2: Lambda | input: | (?, 50, 50) | output: | (?, 50, 50)
Beta: Dot | input: | [(?, 50, 50), (?, 50, 300)] | output: | (?, 50, 300)
Alpha: Dot | input: | [(?, 50, 50), (?, 50, 300)] | output: | (?, 50, 300)
Concat_x1_beta: Concatenate | input: | [(?, 50, 300), (?, 50, 300)] | output: | (?, 50, 600)
Concat_x2_alpha: Concatenate | input: | [(?, 50, 300), (?, 50, 300)] | output: | (?, 50, 600)
BiLSTM_G(lstm_1): Bidirectional(LSTM) | input: | (?, 50, 600) | output: | (?, 50, 400)
Aggregate_v1: Lambda | input: | (?, 50, 400) | output: | (?, 400)
Aggregate_v2: Lambda | input: | (?, 50, 400) | output: | (?, 400)
Concat_v1_v2: Concatenate | input: | [(?, 400), (?, 400)] | output: | (?, 800)
FF_H: Sequential | input: | (?, 800) | output: | (?, 200)
Output: Dense | input: | (?, 200) | output: | (?, 3)

Fig. 3: DAM-BiLSTM Architecture.

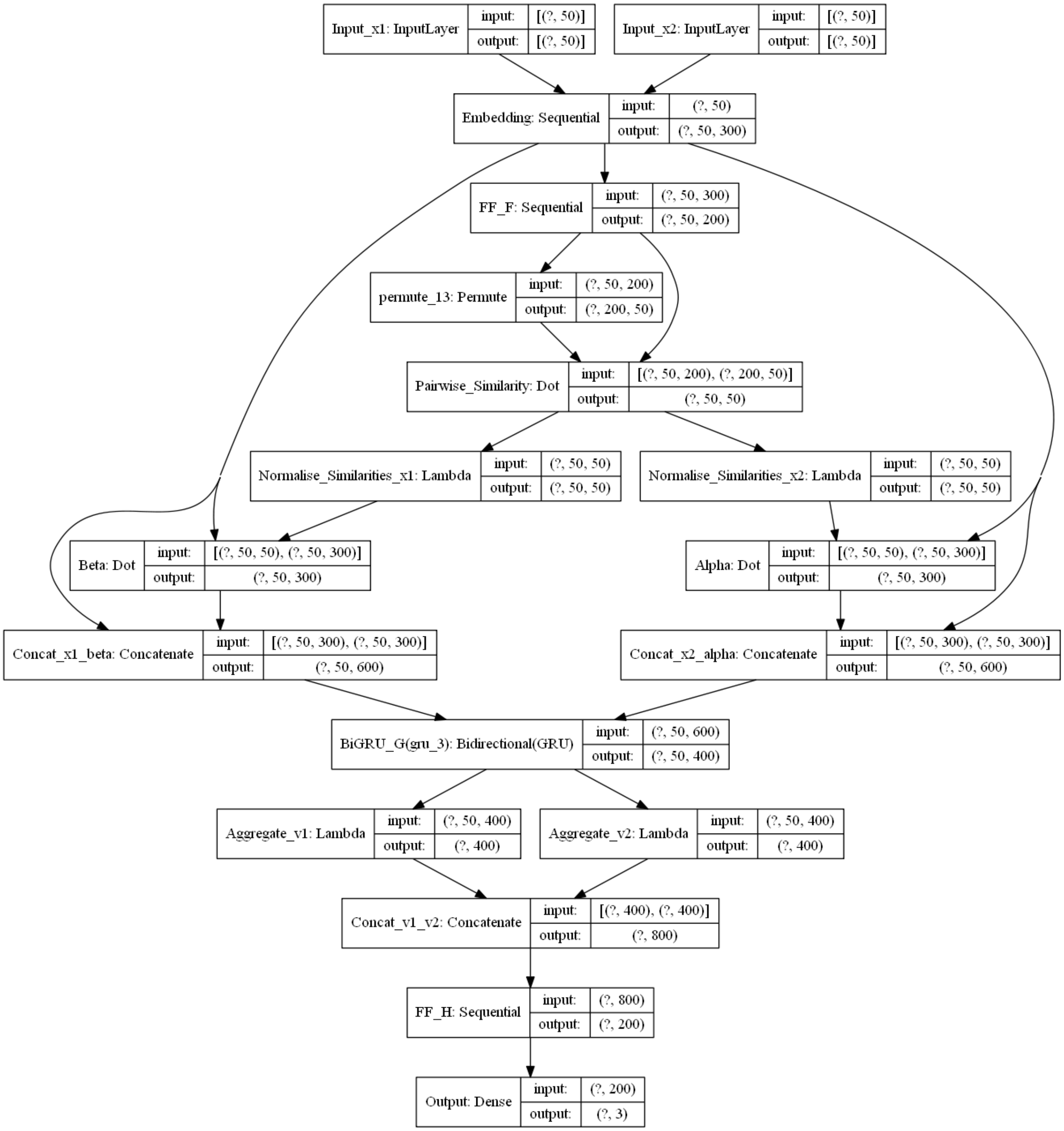
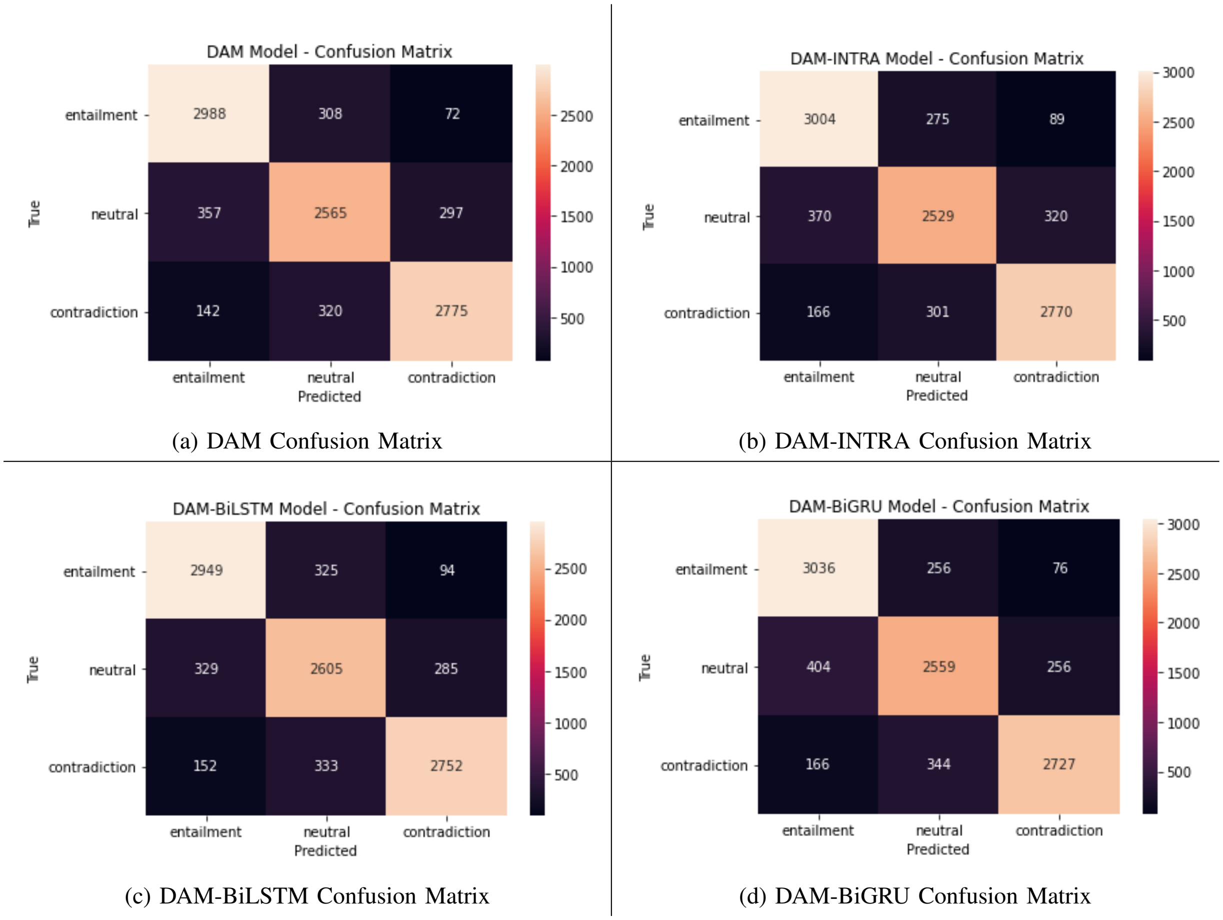


Fig. 4: DAM-BiGRU Architecture.

B. Multi-Class Confusion Matrices



	entailment	neutral	contradiction
entailment	2988	308	72
neutral	357	2565	297
contradiction	142	320	2775

(a) DAM Confusion Matrix

	entailment	neutral	contradiction
entailment	3004	275	89
neutral	370	2529	320
contradiction	166	301	2770

(b) DAM-INTRA Confusion Matrix

	entailment	neutral	contradiction
entailment	2949	325	94
neutral	329	2605	285
contradiction	152	333	2752

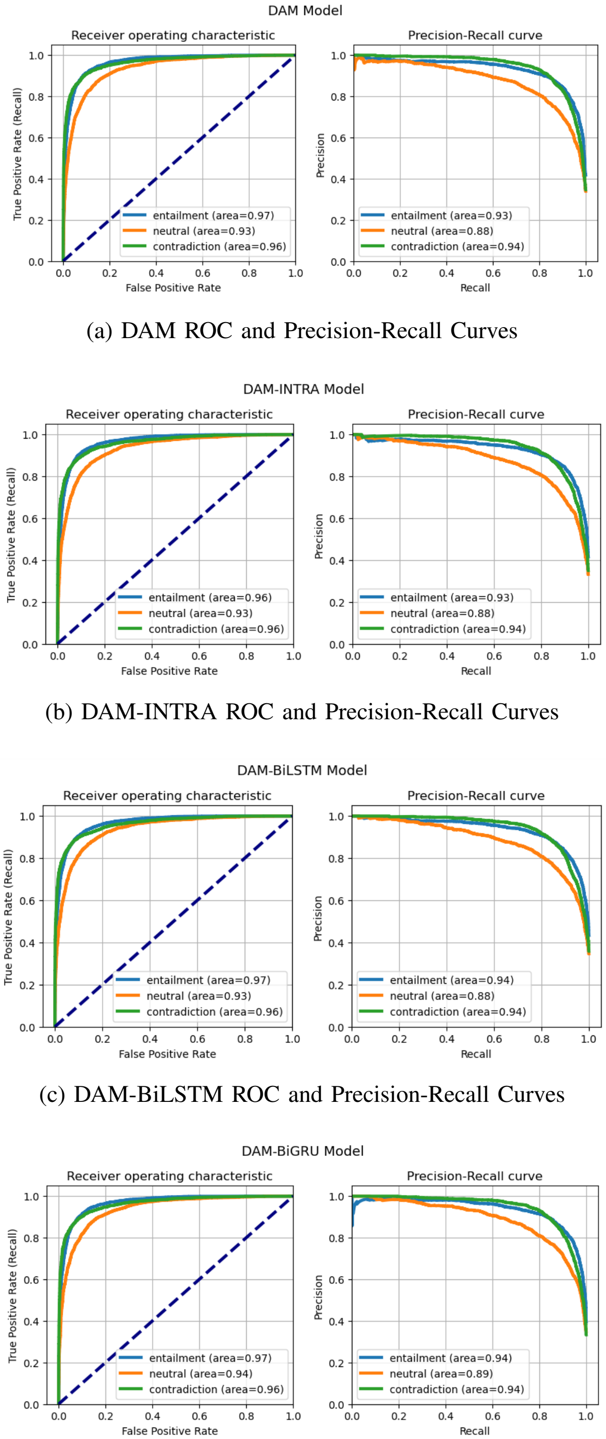
(c) DAM-BiLSTM Confusion Matrix

	entailment	neutral	contradiction
entailment	3036	256	76
neutral	404	2559	256
contradiction	166	344	2727

(d) DAM-BiGRU Confusion Matrix

TABLE IV: Multi-class confusion matrices for each model.

C. ROC and Precision-Recall Curves



(a) DAM ROC and Precision-Recall Curves

(b) DAM-INTRA ROC and Precision-Recall Curves

(c) DAM-BiLSTM ROC and Precision-Recall Curves

(d) DAM-BiGRU ROC and Precision-Recall Curves

TABLE V: ROC and Precision-Recall curves for each model..