Pungene at DialAM-2024: Identification of Propositional and Illocutionary Relations

Sirawut Chaixanien*, Eugene Choi*, Shaden Shaar, Claire Cardie,
Cornell University
{sc2343, ec727, ss2753, ctc9}@cornell.edu

Abstract

In this paper we tackle the shared task DialAM-2024 aiming to annotate dialogue based on the inference anchoring theory (IAT). The task can be split into two parts, identification of propositional relations and identification of illocutionary relations. We propose a pipelined system made up of three parts: (1) locutionary–propositions relation detection, (2) propositional relations detection, and (3) illocutionary relations identification. We fine-tune models independently for each step, and combine at the end for the final system. Our proposed system ranks second overall compared to other participants in the shared task, scoring an average f1-score on both sub-parts of 63.7.

1 Introduction

This paper is a system design paper for the DialAM-2024 task. This task involves the creation of dialogue annotations from dialogue text. Specifically, annotations in the format of a graph under the Inference Anchoring Theory (IAT) Framework. The IAT (Ruiz-Dolz et al., 2024) framework allows for dialogue argumentation annotations in a way that retains relevant information and structural data irrespective of domain.

For this task, we are provided with a dataset that contains numerous .json files where each document represents a graph under the IAT framework. The data used is the QT30 corpus (Hautli-Janisz et al., 2022), where dialogue is taken from 30 episodes of the show Question Time.

Our system is a pipeline that splits the tasks into three steps. At the first step we utilize BERTScore to produce similarity scores to find connections. Then, for each step, we fine-tune a BERT model to perform multiclass classification using information gained from previous steps as input. We fine-tune each model separately and combine it into a pipeline at the end where we create a finished graph. With regards to scoring, we were second in the General case, beating third place by 8 points. We also scored third in the Focused setting.

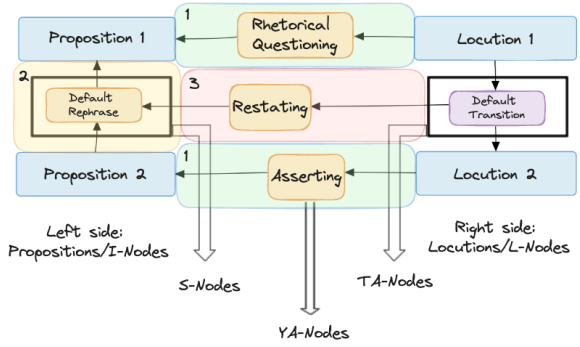

Figure 1: Example final output. The blue nodes on the left are I-nodes (propositions), the blue nodes on the right are L-nodes (locutions), the orange nodes are S-nodes (relations), the yellow nodes are YA-nodes, and the purple nodes are TA-nodes (which depict a transition from one utterance to the next). The ordering of the pipeline mentioned later is also depicted starting with green to yellow then to red as the final step.

2 Related Works and Background

2.1 Task Background

The main goal of this task is to construct a dialogue graph under the IAT format. The input of this task is an unfinished graph that contains L-nodes (Locutions), I-nodes (Propositions), and TA-nodes (Direct Transition). The L-nodes are all connected in the order they were uttered in, with a TA-node between each L-node. On the other hand, the I-nodes are unconnected with anything else.

The expected output is a fully populated graph (see Figure 1) that contains other node types. These include S-nodes which go between I-nodes, YA-nodes which go between an I- and an L-node, or between a TA- and an S-node. Unlike the given input nodes, these also have a type assigned to them.

This task is split into two subtasks. Subtask

Proceedings of the 11th Workshop on Argument Mining, pages 119–123

A: Identification of Propositional Relations which involves the detecting argumentative relations between I-nodes and Subtask B: Identification of Illocutionary Relations which involves detecting illocutonary relations between I-nodes and L-nodes.

S-nodes can have the type of Default Inference (RA-node), Default Rephrase (MA-node), or Default Conflict (CA-node). YA-nodes can have the type of Asserting, Agreeing, Arguing, Challenging, Disagreeing, Default Illocuting, Pure Questioning, Assertive Questioning, Rhetorical Questioning, Restating, and Analysing.

2.2 Related Works

This is the first year for this shared task thus there are no prior works on it. However, many existing systems can help in finding a solution. One core system that was used in our approach is BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019). This Large Language Model meets many of our requirements. Firstly, BERT is very adaptable, being able to perform well on a wide variety of tasks with some fine-tuning. Furthermore, it has been pre-trained on a large corpus allowing our approach to be able to handle information from any domain. One such byproduct of this system is BERTScore (Zhang* et al., 2020) which is also used in our approach. It is a text-generation evaluation metric that utilizes BERT's pre-trained contextual embeddings to calculate similarity scores. A similar method is ROUGE (Lin, 2004) which compares the words directly, causing it to be more easily fooled by similar surface forms as compared to BERTScore.

3 Method

To tackle the problem of populating this graph, we decided to isolate each part to create a pipeline. Rather than following the subtasks laid out, we split it into three different steps instead. The first step is to connect the I- and L-nodes and label the YA-nodes that lie between them. Secondly, as we now know the ordering of the I-nodes, we can then label the S-nodes that go between each I-node. Thirdly, we would connect the TA-nodes to the created S-nodes and label the YA-nodes that go between them. This ordering is labeled in Figure 1 as well.

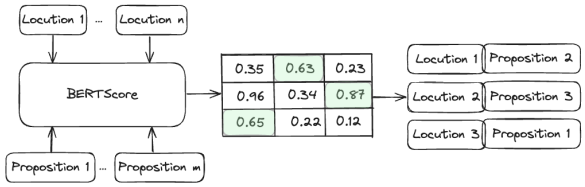

Figure 2: Each combination of locution and proposition is assigned a similarity score. The matching that maximizes the total similarity score with no overlap is chosen as the ideal matching.

3.1 Step 1: I- and L-node Connection and Classification

This step can be further split into 2 subtasks. Namely, the identification of a connection between a locution and a proposition, and the classification of the node type that connects the two.

To tackle the first subtask of identifying the connection to create a pairing, we decided to use different evaluation metrics in order to get a similarity score between propositions and locutions. For each locution, we would compare it to every proposition and select the proposition with the highest similarity score to that locution to be a pair. We calculated accuracy by taking the number of correct pairs over all the pairs in the dataset. We tried 3 different evaluation metrics: ROUGE-1, ROUGE-2, and BERTScore. Here, we found that BERTScore had the highest accuracy (97%) in identifying connections followed by ROUGE-2 (96%) and then ROUGE-1 (94%).

For a more complex approach, we also tried fine-tuning a BERT model to perform inference as to whether a locution and a proposition were connected. The data we used for this was created by going through each document in the dataset extracting each locution and proposition. We traverse through the graph to find every instance of a connection of L-node to YA-node to I-node. For each instance of this, we say that that L-node and that I-node are connected. Next, we generate every possible pair of L-node and I-node. These pairs have a label of 1 if they are connected and 0 otherwise. By doing so, we are able to create a larger dataset through negative sampling.

The results of the fine-tuned BERT not only took more time to infer but also did not perform as well as simply using BERTScore. We believe that this is because a lot of the time the proposition is just a rephrasing of the locution therefore allowing a simple technique to work fine.

As a result, we decided to use BERTScore in our final system (see Figure 2). However, rather than simply sorting by the highest similarity score and picking one by one, we used an algorithm that maximized the total similarity score by checking the total score of every possible pairing.

For the second subtask, we also decided to use a BERT model as they also perform well on classification tasks. We first decided to fine-tune a DistilBERT model (Sanh et al., 2019) to perform multiclass classification to select what type of YA-node would go between the locution and proposition pair. Due to low performance, we tried BERT-base which performed much better.

The data used for this step was just taking each example of a YA-node in between an I- and an L-node. The downside of this straightforward method is that the amount of data was quite small due to the small dataset already being a limiting factor. Furthermore, out of all the labels, "Asserting" was the most common one at 14765 samples while the rest had less than a thousand samples. Because of this class imbalance, we duplicated every other class 10 times.

To finish this step we update the graph with new nodes and edges and pass the updated graph along to the next step.

3.2 Step 2: I-node Connection and Classification

The purpose of this step is connect the I-nodes that follow one another and label the S-node that goes between them. Since all the I-nodes are connected to an L-node, this means that an ordering has already been established. Thus, the only thing we need to do is to decide whether or not two I-nodes have a connecting S-node and what the label of the S-node is. Figure 3 shows what a possible result of step 2 looks like.

This part is quite similar to step 1 where we need to decide whether there is a connection or not and then label what type of connection it is. However, unlike the first step, we already know what the pairing is (namely I-node n and I-node n+1). Therefore, instead of splitting it into two parts like the first step, we decided to do it in one go. We decide to follow a similar method of fine-tuning a BERT model to perform multiclass classification. The model would take in the two I-nodes and have the option of no connection or any one of the possible labels. The four options are: RA (Default Inference), MA (Default Rephrase), CA (Default Conflict), or no connection.

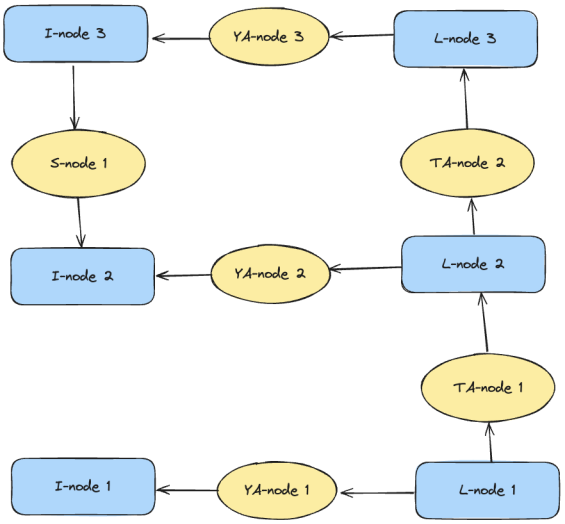

Figure 3: Possible result of the second step. Some I-nodes have an S-node between them and some don't. This image also contains the results from the first step namely the YA-nodes between the connected I- and L-nodes.

The input of this model is the pair of I-nodes. We considered using other nearby nodes to provide more information, however, our results show that adding more information does not improve performance. Using only the two I-nodes gives us an F1 score of 52.5 while adding the nearby L- and YA-nodes results in a lowered F1 score of 51.6.

To fine-tune the model, the data we used was all the real connections as well as pairing up non-adjacent I-nodes. These non-adjacent sampels would be used as samples for the option of no connection. From this, RA had 5,566 samples, MA had 4,508 samples, CA had 882 samples, and no connection had 8,186 samples. Due to CA having such a low number up samples, we decided to upsample it by adding copies of the samples into the dataset. We multiplied it by 4 to give CA a total of 3,528 samples.

To finish this step, we update the graph with the new S-nodes and edges and pass the updated graph to the next step along the pipeline.

3.3 Step 3: YA-node between TA- and S-nodes Classification

The main purpose of this step is to label the YA-node that lies between every S- and TA-node in the same rank (For example S-node 1 and TA-node 2 in Figure 3). To do this, we used the same method of fine-tuning a BERT model. We took every instance

of YA-nodes between S- and TA-nodes as our data. We also faced a similar problem of class imbalance which we decided to solve by upsampling the classes with fewer samples.

The input into our model was all the surrounding nodes, many of which were created in the previous steps. For example, in Figure 3, figure out the label of the YA-node between S-node 1 and TA-node 2, our inputs would be L-nodes 2 and 3, I-nodes 2 and 3, YA-nodes 2 and 3 and S-node 1. The input would be one long chunk of text that concatenated the texts of the I- and L-nodes and the labels of the YA- and S-nodes represented as an integer. The information from each node would be separated by a [SEP] token. Example input: `there's obviously some schools are going to go back on 1st June [SEP] Fiona Bruce : There's obviously some schools are going to go back on 1st June [SEP] 0 [SEP] some schools are not going to go back on the 1st June [SEP] Fiona Bruce : Some are not [SEP] 0 [SEP] 2`.

4 Experiment Details

4.1 Step 1

For the connection part, we used the Hugging Face implementation of BERTScore and our own algorithm for trying every matching. For the classification model, we used bert-base-cased. We finetuned on a GPU with a batch size of 32 and a learning rate of 5e-5 for 5 epochs.

4.2 Step 2 and 3

For the two classification models for steps 2 and 3, we used the same parameters. They both used bert-large-uncased and were fine-tuned on a CPU with a batch size of 8 and a learning rate of 2e-5 for 3 epochs.

5 Results and Analysis

The main method used to measure the success of our system is by calculating precision, recall, and macro-F1. A score will be calculated for Subtask A (ARI) and Subtask B (ILO) each as well as a global score which is the aggregate of the two. Furthermore, they will be split into two different versions: Focused and General. Focused evaluates the performance looking at only the related classes in the evaluation files only while General also includes the non-related class. This means that a high performance in the General version but low performance in the Focused versions shows a pessimistic approach (overly relies on the non-related class) while the inverse shows an optimistic approach that relates too many propositions and locutions.

Metric	F1
ARI - Focused	20.51
ARI - General	46.22
ILO - Focused	69.95
ILO - General	81.17
GLOBAL - Focused	45.23
GLOBAL - General	63.70

Table 1: F1 Score for each evaluation metric. Both Focused and General ILO are quite high. The Focused ARI has a low score while General ARI has a better score. Overall GLOBAL score, which is the aggregate of the two, is good with the General case performing better.

Another thing to note is that a big downside of our pipelined system is that it is very prone to cascading errors. This is also an additional reason as to why in step 2 we opted to use as few inputs as possible in order to prevent the cascading of errors. The only part which used a lot of the information from the previous steps was step 3. However, the added information allowed the model to get an F1 score of 96.2 which is very strong.

From Table 1 we can see that the main part that performs well is Subtask B (ILO). Both focused and general cases perform quite well indicating a good balance of predictions in every class.

6 Conclusion and Future Work

Overall our system performed quite well, especially on Subtask B which was the identification of illocutionary relations. The recurring technique that we used was fine-tuning a BERT model which proved to be quite effective. A strong point of our system is its ability to get quite similar scores among both General and Focused cases. This is likely due to our upsampling which helped with the largely imbalanced dataset. The one downside in our system seems to be the issue of cascading errors. This is reflected in the scores as we do part of Subtask B first before moving on to Subtask A and the ILO scores are much higher than our ARI scores. Moving forward we will need some way to eliminate the impact of these errors.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Annette Hautli-Janisz, Zlata Kikteva, Wassiliki Siskou, Kamila Gorska, Ray Becker, and Chris Reed. 2022. QT30: A corpus of argument and conflict in broadcast debate. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 3291–3300, Marseille, France. European Language Resources Association.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Ramon Ruiz-Dolz, John Lawrence, Ella Schad, and Chris Reed. 2024. Overview of DialAM-2024: Argument Mining in Natural Language Dialogues. In *Proceedings of the 11th Workshop on Argument Mining*, Thailand. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.