Improving Conversational Entailment with Instance-Specific Knowledge Graphs

Anonymous ACL submission

Abstract

Conversation entailment, the task of determining if a hypothesis can be inferred from a multi-turn dialogue, presents challenges due to the complex nature of conversational dynamics. Transformer-based models like BERT excel in capturing language patterns and have shown strong performance in entailment tasks. However, as highlighted by Storks and Chai (2021), these models often lack coherence in intermediate reasoning, relying on spurious correlations that undermine interpretability and trust. To address this, we proposed augmenting transformers with instance-specific knowledge graphs to enhance reasoning coherence and accuracy. While our approach demonstrated improvements in accuracy and coherence metrics, the complexity and computational overhead involved suggest that the gains may not justify the additional effort for most applications.

Code for our project can be found in our GitHub repository. [1].

1 Introduction

Understanding and predicting entailment in conversations is a difficult but crucial task in natural language processing. Conversational entailment involves determining whether a hypothesis can logically be inferred from a multi-turn dialogue. Unlike conventional textual entailment, which deals with relatively static and structured texts, conversation entailment introduces additional complexities, such as dialogue turns, implicit references, and long-distance dependencies between statements. This makes it a uniquely challenging problem for current natural language processing models.

Transformer-based models, such as BERT, have shown significant promise in handling natural language processing tasks, including entailment. Their success largely comes from pretraining on large datasets, allowing them to learn representations of language. However, these models often fall short when applied to conversational entailment because they focus on surface-level correlations within the data, sometimes leading to inaccurate or logically inconsistent predictions. As shown by previous research, this can reduce the trust and interpretability of these models, especially in uses where logical consistence is important.

The main issue lies in the intermediate reasoning process of transformer-based models. While they can achieve high accuracy metrics, the way they reach their conclusions is a black box and their logical reasoning could be incorrect which can lead to inconsistent results. This lack of coherence not only affects the robustness of these models but also reduces their usability in real-world scenarios. To address these limitations, we propose using instance-specific knowledge graphs with a transformer-based model. By explicitly modeling the relationships and entities within a dialogue, knowledge graphs provide structured, contextual information that can enhance the model's reasoning process.

1.1 Dialogue Example for Coherent Text Classification

Dialogue:

- **A1:** "I finally submitted my application for the job."
- **B1:** "That's great! How do you feel about it?"
- **A2:** "Honestly, I'm not sure. I keep wondering if I should have double-checked everything one more time."

Hypothesis:
Speaker A feels unsure about their application submission.

[1] https://github.com/jack2kiwi/NLP-595-Conversational_Entailment_Instance_Specific_KG

In this example, the hypothesis cannot be directly inferred from any single segment in the dialogue. The hypothesis requires an understanding of the overall context:

Evaluation:

Segment	Hypothesis Support
"I finally submitted my application for the job."	X
"That's great! How do you feel about it?"	X
"Honestly, I'm not sure. I keep wondering if I should have double-checked everything one more time."	X
Full dialogue (A1, B1, A2 combined)	✓

While the hypothesis aligns with the final segment, the context from all three segments are necessary for a coherent inference. A model relying solely on statistical patterns might overlook the connections between these statements or give unnecessary importance to isolated phrases. By introducing a knowledge graph that captures relationships (e.g., "Speaker A" $\rightarrow$ "submitted application" and "Speaker A" $\rightarrow$ "feels unsure"), the model gains access to a structured representation that supports logical reasoning.

2 Related Work

Our work builds upon several previous developments in natural language processing, particularly conversational entailment and graph-based learning. Such advancements in NLP tasks form the foundation of our methodology.

2.1 Conversational Entailment

Previous research in conversational entailment has addressed challenges in reasoning over dialogue. Zhang and Chai (Zhang and Chai, 2010) emphasized how traditional textual entailment frameworks are inadequate when applied to dialogue, which can contain elements such as turn-taking, lingustic phenomena of utterances, and implicature. In order to handle these elements, they developed a modeling framework using two levels of semantic representation: a basic representation based on syntactic parsing of utterances, and an augmented representation that incorporates conversational features, such as dialogue acts, to capture a more in-depth context. Additionally, Zhang and Chai experimented with long-distance relationship (LDR) modeling, which bridges semantic gaps between constituents through two approaches: implicit modeling, which looks at the distance between terms, and explicit modeling, which represents semantic paths as strings to capture more nuanced relationships. Combining conversation structures with explicit LDR modeling provided the highest accuracy, showing that both structural and relational modeling play vital roles in conversational entailment.

Building on this foundation, Storks and Chai (Storks and Chai, 2021) proposed a broader evaluation of model performance through coherence-based metrics, which also evaluates the internal consistency of model predictions rather than just the accuracy. Their coherence framework prioritizes alignment with human reasoning, identifying inconsistencies in intermediate model decisions. This coherence-based approach contributes to developing more interpretable, robust systems capable of aligning with human logic in entailment. Together, these works highlight the importance of structure, coherence, and relationship modeling to achieve reliable performance in conversation understanding tasks.

2.2 Knowledge Graphs

The importance of knowledge graphs in organizing and representing data has also been explored in the past. Peng et al. (Peng et al., 2023) details how knowledge graphs have significantly impacted AI tasks such as recommendation systems, question answering, and information retrieval by allowing complex information to be efficiently modeled and accessed. Even though there are still challenges regarding knowledge graph construction and application, the potential for knowledge graphs to enhance logical consistency in dialogue-based situations presents opportunities for improving conversational entailment.

2.3 Graph-based Learning

Graph-based learning has been explored as an innovative area for improving machine learning models in situations involving structured data. Algorithms and frameworks such as Node2Vec and GraphGPT have introduced novel approaches for incorporating graph representations in AI tasks.

2.3.1 Node2Vec

Node2Vec is a method of node embedding that transforms graphs into numerical representations (Grover and Leskovec, 2016). It introduces a flexible framework for learning feature representations of nodes in graphs. In contrast to traditional embedding methods, which rely on rigid definitions of node neighborhoods, Node2Vec employs biased random walks to sample neighborhoods dynamically. These walks utilize both breadth-first sampling (BFS), which emphasizes local structural equivalence, and depth-first sampling (DFS), which captures global community structures. This flexibility allows Node2Vec to generate embeddings that reflect both homophily, where nodes that belong to similar network clusters are embedded closely together, and structural equivalence, where nodes that have similar structural roles in networks are embedded closely together. By optimizing a neighborhood-preserving objective using stochastic gradient descent, Node2Vec achieves efficient performance in tasks such as multi-label node classification and link prediction. Its effectiveness has been demonstrated on large real-world networks, and it outperforms other approaches such as DeepWalk, LINE, and spectral clustering.

2.3.2 GraphGPT

Tang et al. (Tang et al., 2024) provides GraphGPT, a framework that aligns large language models with graph-structured data through an instruction-tuning approach. It addresses the challenges of incorporating graph structural information with textual data by introducing a text-graph grounding paradigm, dual-stage instruction tuning, and chain-of-thought distillation. The text-graph grounding paradigm aligns graph structures with the natural language space. Semantic understanding of textual information is connected with structural relationships within the graph, ensuring compatibility with LLMs.

The second part of GraphGPT consists of dual-stage graph instruction tuning. This instruction tuning aligns the language model's reasoning capabilities with the nuances of graph learning tasks, allowing for more accurate and appropriate responses. In the first stage, self-supervised instruction tuning, the language model's reasoning is improved by including structural knowledge specific to the graph domain. In order to enact this process, unlabeled graph structures are used to generate self-supervised signals that are used as instructions for model tuning. The second stage consists of task-specific instruction tuning, which tailors the language model's reasoning behavior for specific tasks. In this stage, specific text information is included in the instruction design in order to further assist the language model.

Following the instruction tuning, the step-by-step reasoning abilities of GraphGPT are improved through chain-of-thought distillation. Using knowledge from the comprehensive language model GPT-3.5, intermediate thought information is incorporated into instructions used for the task-specific instruction tuning portion. This improves coherence and consistency, allowing for better performance in situations involving more diverse graph data.

2.4 Named Entity Recognition

Named Entity Recognition (NER) is a process in natural language processing that involves identifying and categorizing entities such as people and organizations within unstructured text (Keraghel et al., 2024). Its effectiveness has been widely explored in the past, and there currently exist several libraries supporting it. SpaCy, which we use in our implementation, is one such library that offers tools and pre-trained models for NER. The models are based on both convolutional neural networks and transformer-based architectures such as BERT. The usage of SpaCy within natural language processing tasks allows for the structuring of textual data and the improved accuracy of downstream processing.

3 Motivation

Conversational entailment requires an in-depth understanding of relationships and speakers within multi-turn dialogues, making it a difficult challenge for natural language processing models. While transformer-based models such as BERT have demonstrated strong abilities in handling natural language, they often struggle to capture the logical coherence necessary for entailment reasoning. This limitation is rooted in their reliance on implicit statistical patterns learned from large-scale datasets, which can lead to predictions that lack interpretability.

A possible solution to this problem uses the integration of knowledge graphs. Knowledge graphs offer a structured representation of entities and their relationships, making them a strong complement to transformer-based models. Unlike language models that rely solely on token-level patterns, knowledge graphs explicitly encode semantic relationships and contextual information. This structured approach enables the model to reason over long-distance dependencies and relationships in dialogues, which are often missed by transformers alone.

By using instance-specific knowledge graphs, we can tailor the model's understanding to the specific context of each dialogue. This explicit representation allows the model to ground its reasoning in the actual content of the dialogue, rather than relying on heuristic shortcuts. The result is a system that is not only more accurate but also more interpretable and aligned with human logic. Therefore, knowledge graphs provide a way to bridge the gap between statistical language modeling and structured reasoning which allows us to develop models that are both accurate and coherent.

4 Methodology

The methodology for our project is centered around enhancing the performance of transformer-based models for the task of conversational entailment. While transformer models like BERT have achieved significant advancements in this area, they often fall short in maintaining coherence and logical consistently in their predictions. We hypothesize that this limitation arises due to their reliance on statistical patterns in large-scale data, rather than explicit modeling of relationships and entities within conversations.

To address this challenge, our methodology introduces a structured pipeline that integrates knowledge graphs into the learning process. Knowledge graphs are very powerful tools for representing entities and their relationships in a structured format, capturing information that is often implicit in text, and spans across different turns in dialogue. By incorporating these graphs, we aim to provide the model with additional contextual information, allowing it to reason more effectively about the dialogue and improve both accuracy and coherence in its predictions.

The proposed pipeline can be delineated by these following key stages:

1. **Entity and Relationship Extraction:** Extracting key entities (e.g., people, organizations, or concepts) and relationships from dialogues using Named Entity Recognition (NER) and dependency parsing techniques. This step ensures that the knowledge graph is grounded in the specific context of each conversation.

2. **Instance-Specific Knowledge Graph Construction:** Querying a global knowledge graph (e.g., DBpedia) using the extracted entities to retrieve subgraphs that represent relevant information. These subgraphs form the foundation of the instance-specific knowledge graph.

3. **Node Embedding Generation:** Transforming the knowledge graph into numerical representations (embeddings) using graph neural networks (GNNs) or node embedding techniques like Node2Vec. These embeddings capture the structural and relational information of the graph.

4. **Augmenting Transformer Input:** Combining the node embeddings from the knowledge graph with token embeddings from a transformer model (BERT) to create an enriched input representation. This integration allows the model to leverage both the linguistic and relational information.

5. **Model Training and Evaluation:** Fine-tuning the augmented transformer model on a conversational entailment dataset, followed by evaluating its performance using metrics such as accuracy and coherence.

We intend that the integration of instance-specific knowledge graphs would address the knowledge gaps in current transformer models. We believe that in using this approach, the predicted results would not only be more accurate, but also logically consistent and interpretable, aligning with human reasoning.

In the following subsections, we dive deeper into each stage of the methodology, providing detailed explanations of the techniques, tools, and processes used to implement this pipeline.

4.1 Entity and Relationship Extraction

Entity and relationship extraction forms a critical foundation for our methodology, providing the structured data needed to construct instance-specific knowledge graphs. The aim of this stage is to identify key entities, such as individuals, organizations, and locations, as well as their relationships

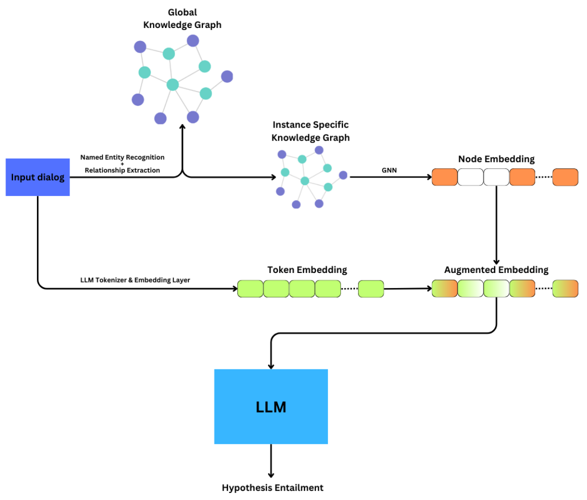

Figure 1: Example diagram of our approach. Named entity recognition and relationship extraction of dialog are used to query a global knowledge graph to construct a instance-specific knowledge graph. Node embeddings are generated from the graph using GNNs and then concatenated with the dialogue's model token embeddings. These augmented embeddings are passed through to the LLM to generate the hypothesis entailment prediction.

within the dialogue. These extracted elements enable the knowledge graph to represent the interactions and contextual dependencies present in the conversation.

We leverage several Natural Language Processing techniques and libraries to perform this step effectively. The key tools used are as follows:

- **Named Entity Recognition (NER):** The purpose of this step is to preliminarily identify entities in the text and categorize them into predefined types such as `PERSON`, `ORG`, and `GPE`. We used SpaCy, a state-of-the-art NLP library, for NER. Specifically, we used SpaCy's pre-trained model `en_core_web_lg` model to analyze each piece of dialogue in separation and label the named entities for every turn.

  In determining the appropriate model for NER, we had numerous options, as this is a field undergoing constant development. Our objective was to balance the speed of the model with the quality of entity recognition. SpaCy offered four model sizes, ranging from small to large, and we observed that the smaller models performed poorly in terms of quality, which is why we opted for the large model. Beyond SpaCy, we also experimented with larger transformer-based NER models, which performed exceptionally well and consistently on this task. However, due to the significantly higher latency and computational resources required to run and load a transformer model, we decided not to use it in our final implementation.

- **Dependency Parsing:** The purpose of this is to identify grammatical relationships between words in a sentence to extract meaningful relationships between entities. We performed dependency parsing using SpaCy's provided dependency tree. It provides information about syntactic dependencies, such as subject (`nsubj`), object (`dobj`), and attributes (`attr`).

- **Coreference Resolution:** The purpose of this process is to resolve references to the same entity across different sentences within the dialogue (e.g., pronouns such as "he," "she," or "it"). This approach significantly enhances the quality of entity and relationship extraction, resulting in more useful and effective instance-specific knowledge graphs. For coreference resolution, we use SpaCy's `coreferee` library.

The tools and techniques mentioned above are systematically integrated into a unified pipeline to extract entities and relationships efficiently from dialogue data. Below, we outline the sequential process and its implementation details:

First, the dialogue text is preprocessed to standardize references using coreference resolution. This step ensures that entities mentioned across multiple turns or sentences are consistently recognized and linked. For example, in the dialogue snippet, *"John Doe submitted his application. He was nervous,"* the pronoun "He" is resolved to "John Doe," producing the standardized version *"John Doe submitted his application. John Doe was nervous."* Unlike the NER model, we opted to use the transformer-based `en_core_web_trf` model, since coreference resolution is not a bottleneck for the overall NER process.

Next, we perform Named Entity Recognition (NER) on the coreference-resolved dialogue text.

Each turn of dialogue is processed individually, and entities such as people, organizations, and locations are identified using SpaCy's pre-trained `en_core_web_lg`. For example, in the sentence *"John Doe is a software engineer at XYZ Corporation,"* NER could identify two entities: "John Doe" as a `PERSON` and "XYZ Corporation" as an `ORG`.

Following NER, we use dependency parsing to identify relationships between the extracted entities. Dependency parsing analyzes the grammatical structure of sentences to uncover syntactic relationships, such as subject-object pairs and attributes. For example, in the sentence *"John Doe is a software engineer,"* dependency parsing identifies "John Doe" as the subject (`nsubj`) and "software engineer" as the attribute (`attr`). By focusing on these syntactic dependencies, we extract meaningful relationships such as `('John Doe', 'software engineer')` and `('XYZ Corporation', 'employer')`.

Finally, the outputs from these stages are consolidated into a structured format. The entities and relationships are stored as lists, which serve as inputs for constructing the instance-specific knowledge graph in the subsequent step. For instance, given the sentence *"John Doe is a software engineer at XYZ Corporation. He graduated from ABC University,"* the pipeline produces the following results:

- **Entities:** `['John Doe', 'XYZ Corporation', 'ABC University']`
- **Relationships:** `[('John Doe', 'software engineer'), ('John Doe', 'XYZ Corporation'), ('John Doe', 'ABC University')]`

4.2 Instance-Specific Knowledge Graph Construction

After extracting entities and relationships from the dialogue, the next step is constructing an instance-specific knowledge graph (KG). This KG provides a structured representation of the dialogue, embedding contextual and relational information essential for enhancing the transformer model's reasoning capabilities. The goal of this step is to integrate both the extracted relationships from the dialogue and external knowledge from global KGs to create a robust and context-aware graph structure tailored to the specific dialogue instance.

This graph enables the model to capture long-distance dependencies, infer implicit relationships, and incorporate external knowledge to improve coherence and accuracy in entailment predictions.

Retrieving External Knowledge: To enrich the extracted entities, we queried external knowledge bases such as DBpedia using SPARQL, a query language for semantic data. Each query returns triples (`subject`, `predicate`, `object`) describing relationships involving the entities. For example, querying DBpedia for the entity `"John Doe"` might return the following triples:

- `(John Doe, affiliatedWith, XYZ Corporation)`
- `(John Doe, publishedBook, Software Engineering 101)`

The retrieved triples are then filtered to retain only the most relevant and contextually appropriate results. This filtering process ensures that the KG remains focused on relationships that align with the dialogue context.

Combining Extracted and External Knowledge: After retrieving relevant external triples, the next step is to combine them with the relationships extracted directly from the dialogue. Relationships from the dialogue are prioritized to preserve contextual relevance, while global triples are used as supplementary information to enrich the graph. For instance, given the dialogue:

> *"John Doe is a software engineer at XYZ Corporation. He has a degree in Computer Science from ABC University."*

The extracted entities are:

- *"John Doe," "XYZ Corporation," "Computer Science," "ABC University."*

And the extracted relationships include:

- (*John Doe, engineer*)
- (*John Doe, degree*)
- (*XYZ Corporation, engineer*)
- (*ABC University, degree*)

Querying DBpedia for additional information about these entities may yield triples such as:

- (*John Doe, affiliatedWith, XYZ Corporation*)

- (*ABC University, locatedIn, Ann Arbor*)
- (*Computer Science, branchOf, STEM Fields*)

These external triples provide additional contextual knowledge, enhancing the graph's representation.

Graph Representation: The instance-specific KG is constructed programmatically using NetworkX. In this representation:

- Nodes correspond to entities (e.g., *"John Doe," "XYZ Corporation," "Computer Science," "ABC University"*).
- Edges represent relationships (e.g., *"John Doe" is an "engineer," "John Doe" has a "degree," "ABC University" offers "Computer Science"*).

For example, the graph constructed from the above dialogue would include nodes for each entity and edges for relationships derived from both the dialogue and external knowledge.

Challenges in KG Construction: One key challenge in constructing the KG is handling ambiguities in entity names. For instance, entities with the same name but different contexts (e.g., "John Smith" from different organizations) could introduce noise into the graph. To resolve this, additional attributes such as associated organizations or locations are used to disambiguate entities. Relationships extracted from the dialogue also take precedence during integration to ensure contextual relevance.

Final Integration: Once the KG is constructed, it provides a unified representation that integrates dialogue-specific relationships with external knowledge. The graph is structured as a directed graph, where edges capture the flow of relationships between nodes. This structured representation forms the foundation for the subsequent step of generating node embeddings, enabling the model to leverage this enriched context.

4.3 Node Embedding Generation

Once the KG is constructed, the next step involves generating node embeddings. Node embeddings are critical because they allow the model to capture the nuanced connections and dependencies between entities in the graph. By embedding nodes into a high-dimensional space, the structural information of the graph (e.g., proximity, relationships, and importance of nodes) is encoded in a form that our machine learning model can leverage for reasoning and inference tasks.

Node Embedding Algorithm: To generate embeddings, we utilize the Node2Vec algorithm. It captures the graph's structural properties by simulating random "walks" and learning embeddings based on co-occurrence patterns in these walks. The key steps include:

1. Random Walk Simulation: For each node, Node2Vec generates multiple random walks of a predefined length. These walks explore the graph structure, capturing both local and global relationships.

2. Optimization: The walks are treated as sequences (similar to sentences in natural language), and a Skip-Gram model is applied to optimize embeddings such that nodes appearing in similar walks have similar embeddings.

3. Hyperparameter Configuration: The following hyperparameters are used in our implementation:

   - **Dimensions:** Embeddings are generated in a 768-dimensional space to align with the transformer model's embedding size.
   - **Walk Length:** Each random walk consists of 30 steps, which we found to be the best at sufficiently exploring the graph structure.
   - **Number of Walks:** Each node is explored through 200 random walks to capture diverse contexts.
   - **Window Size:** A context window size of 10 is used during the Skip-Gram optimization process.

Output of Node Embeddings: The output of this step is a dictionary, mapping each entity node to its corresponding embedding vector. Each vector is a 768-dimensional numerical representation that captures both the local and global graph structure around the node. For instance:

- Node: *"John Doe"* $\rightarrow$ Embedding: `[0.32, -0.54, 0.78, ...]`

- Node: *"XYZ Corporation"* $\rightarrow$ Embedding: `[0.12, 0.34, -0.67, ...]`

Challenges in Node Embedding Generation: One challenge in this step is ensuring that embeddings capture the relevant context without introducing noise from unrelated nodes or relationships. To address this:

- Random walks are parameterized to balance exploration (capturing global context) and exploitation (focusing on local neighborhoods).
- Only nodes corresponding to entities extracted from the dialogue are retained in the final embedding dictionary, ensuring relevance to the entailment task.

Summary: These node embeddings serve as the bridge between the knowledge graph and the transformer model. By encoding the structural and relational information of the KG into a numerical format, they enable the downstream model to leverage this enriched context for more accurate entailment predictions. The next section describes how these embeddings are integrated with the transformer model's token embeddings to form an augmented input representation.

4.4 Augmenting Transformer Inputs

After generating node embeddings from the KG, the next step involves integrating these embeddings with the transformer model's token embeddings. Transformer models like BERT are highly effective at capturing textual patterns and semantics but lack an inherent understanding of structured relationships and graph-based dependencies. We will show how we combine an inherently sequential model with graphical node embeddings from the previous step to provide an additional layer of context for the model.

Token Embedding Generation: As is common for regular text tokens, dialogue text is tokenized in our model using the transformer model's tokenizer (BERT's WordPiece tokenizer). Each token is mapped to a unique ID and subsequently passed through the embedding layer of the transformer, generating a 768-dimensional token embedding for each token. These embeddings capture the semantic context of the dialogue text.

For instance, in the sentence *"John Doe is a software engineer at XYZ Corporation,"* the tokens *"John," "Doe," "is," "a," "software," "engineer,"* and *"XYZ Corporation"* are converted into their respective token embeddings.

Node Embedding Alignment: To integrate node embeddings, we align the tokens with their corresponding nodes in the KG. Each entity extracted from the dialogue is matched to its corresponding tokens in the tokenized text. For example:

- Entity: *"John Doe"* → Tokens: *"John," "Doe"*
- Entity: *"XYZ Corporation"* → Tokens: *"XYZ," "Corporation"*

If a token corresponds to a node in the KG, its embedding is augmented with the node embedding generated in the previous step.

Augmentation Process: The augmentation process involves concatenating the token embeddings and their corresponding node embeddings. This step ensures that each token in the input sequence is enriched with additional context from the KG. The concatenation is performed as follows (with $\mathbf{e}$ representing the vector embedding):

$$\mathbf{e}_{\text{aug}} = \text{Concatenate}(\mathbf{e}_{\text{token}}, \mathbf{e}_{\text{node}})$$

The resulting augmented embedding has a dimensionality of 1536 (768 from the token embedding and 768 from the node embedding).

If a token does not correspond to any node in the KG, its embedding is concatenated with a zero vector of size 768 to maintain consistency.

Example: For the sentence *"John Doe is a software engineer at XYZ Corporation,"* consider the token *"John Doe"* and its corresponding node embedding from the KG. The token embedding for *"John Doe"* might be:

$$\mathbf{e}_{\text{token}} = [0.32, -0.45, 0.67, \ldots] \quad \text{(768 dimensions)} \tag{1}$$

The node embedding for *"John Doe"* could be:

$$\mathbf{e}_{\text{node}} = [0.12, 0.56, -0.89, \ldots] \quad \text{(768 dimensions)} \tag{2}$$

The concatenated embedding becomes:

$$\mathbf{e}_{\text{aug}} = [0.32, -0.45, 0.67, \ldots, 0.12, 0.56, -0.89, \ldots] \quad \text{(1536 dimensions)} \tag{3}$$

A linear layer then reduces this augmented embedding to 768 dimensions.

Challenges:

- **Token-Node Alignment:** We had to do extra processing to ensure accurate alignment between tokens and nodes, especially for multiword entities.

- **High Dimensionality:** Concatenating embeddings increases the dimensionality, which necessitated a reduction step to match the transformer model's input requirements.

- **Handling Missing Nodes:** Most of the time, tokens do not correspond to nodes in the KG. In these instances, we used zero-padding for tokens to represent their non-existent node embedding.

Summary: By augmenting the transformer's input layer, we ensure that the model has access to both semantic and relational information during training and inference, effectively bridging the gap between unstructured dialogue data and structured knowledge graphs.

4.5 Training Pipeline

The goal of our training pipeline is to fine-tune a transformer-based model on the conversational entailment task, with augmented inputs that include both token embeddings and graph-based node embeddings.

We used the Conversational Entailment dataset from the SLED Lab. This dataset includes structured conversation data, where each entry comprises of dialogue text, a type classification (fact, intent, desire, or belief), a hypothesis sentence based on the dialogue, and an entailment that describes if the hypothesis can be inferred from the dialogue.

Data Preprocessing: The input dialogue data is processed to prepare it for training:

- **Tokenization:** Dialogue text is tokenized using the transformer model's tokenizer (e.g., BERT WordPiece tokenizer). This step converts the dialogue and hypothesis into a sequence of tokens, which are then numericalized into unique token IDs.

- **Feature Augmentation:** The tokenized dialogue is aligned with the node embeddings generated from the instance-specific knowledge graph. Token embeddings and node embeddings are concatenated to form augmented embeddings, as described in the previous subsection.

- **Padding and Masking:** To ensure uniform input length, sequences are padded to a fixed length (e.g., 128 tokens). An attention mask is generated to differentiate between real tokens and padding tokens, allowing the model to ignore padding during training.

Model Initialization: The transformer model (e.g., BERT) is initialized with pre-trained weights. A custom embedding layer is added to handle the augmented inputs, which consist of both token and node embeddings. This layer includes a fully connected linear layer to reduce the dimensionality of the augmented embeddings back to 768 dimensions, matching the transformer's input requirements.

Training Setup: The training process involves optimizing the model's parameters using a supervised learning approach. Key components include:

- **Loss Function:** Cross-entropy loss is used to measure the difference between the predicted probabilities and the ground-truth labels. We used this loss function since it is commonly used for binary or multi-class classification tasks.

- **Optimizer:** AdamW (Adaptive Moment Estimation with Weight Decay) is used as the optimizer. This model is very popular, especially for fine-tuning transformer models, as it has adaptive learning rates and regularization.

- **Learning Rate Scheduler:** A linear learning rate scheduler with warm-up steps is used to gradually increase the learning rate at the beginning of training, to prevent gradient instability and improves convergence. An overall faster learning rate was utilized in our BERT+KG than within the baseline. This came from both experimental results and training the linear layer to augment embeddings.

- **Batching:** Training data is divided into mini-batches, allowing the model to process multiple examples simultaneously. We found a batch size of 32 to be a good balance between computational efficiency and memory constraints (considering limited computational resources).

Cross-Validation: To reduce over fitting, we used an 8-fold cross-validation approach. The dataset is split into 8 folds, with 7 folds used for training and 1 fold used for validation in each iteration. This process is repeated 8 times, rotating the validation fold each time, ensuring that all examples are used for both training and validation.

Model Training: The model is trained for a fixed number of epochs (8) on each fold. During training, the model processes each mini-batch, computing the loss and updating its parameters through backpropagation. We also found gradient clipping to be useful in preventing exploding gradients. Training progress is monitored, and hyperparameters are adjusted using metrics such as loss and accuracy on the validation set.

Evaluation: After each training epoch, the model is evaluated on the validation set. Validation accuracy and loss are recorded and used to assess the model's performance and identify if our model is overfitting. Metrics such as precision, recall, and F1-score are computed to enumerate the model's performance. Additionally, consistency and coherence metrics are evaluated to measure the logical alignment of the model's predictions with human reasoning.

Final Model Selection: Once cross-validation is complete, the model with the best validation performance is selected for further evaluation on the test set.

Testing on Unseen Data: The selected model is tested on the held-out test set to evaluate its generalization capabilities. Metrics are computed on this unseen data to provide a comprehensive assessment of the model's performance.

Challenges and Considerations: We encountered several challenges while constructing our training pipeline.

- **Computational Overhead:** Augmented inputs increase the computational requirements, necessitating efficient batching and optimization techniques.
- **Alignment Errors:** Ensuring accurate alignment between tokens and nodes requires robust preprocessing and validation.
- **Hyperparameter Tuning:** Parameters such as learning rate, batch size, and the number of epochs are tuned to achieve optimal performance.

4.6 Coherence Checks

To evaluate the logical consistency and interpretability of the model's predictions, we conducted coherence checks as a supplementary evaluation process. Coherence, in the context of conversational entailment, refers to the model's ability to maintain consistent reasoning across subparts of a dialogue and align its intermediate decisions with human-like logical steps. This step ensures that the model is not only accurate but also reliable and explainable.

4.6.1 Motivation for Coherence Evaluation

While traditional metrics such as accuracy and F1-score provide a measure of predictive performance, they do not capture the logical validity of the model's intermediate reasoning. For instance, a model may arrive at a correct prediction by relying on spurious correlations or shortcuts rather than reasoning logically through the dialogue. Coherence checks address this limitation by checking:

- **Consistency of Subspan Predictions:** The model should make consistent entailment decisions for individual subspans of the dialogue that logically align with the overall entailment decision.
- **Alignment with Human Reasoning:** The model's predictions should follow logical reasoning paths that are interpretable and understandable to humans.
- **Mitigation of Spurious Patterns:** Coherence checks help identify and reduce reliance on patterns or correlations that do not contribute to meaningful reasoning.

4.6.2 Subspan-Level Entailment Evaluation

To perform coherence checks, the dialogue was split into subspans, each representing a contiguous portion of the conversation. For example, given a dialogue with three turns:

- Full dialogue: *"Speaker A: I submitted my application. Speaker B: That's great! How do you feel? Speaker A: I'm not sure, maybe I should have double-checked."*
- Subspan 1: *"Speaker A: I submitted my application."*

- Subspan 2: *"Speaker A: I submitted my application. Speaker B: That's great! How do you feel?"*

The model was tasked with predicting entailment for the hypothesis based on each subspan.

4.6.3 Coherence Metrics

We employed multiple coherence metrics to evaluate logical consistency:

- **Span Accuracy:** The proportion of subspans where the model's entailment prediction aligns with the overall prediction for the full dialogue.
- **Strict Coherence:** Measures whether the model consistently predicts entailment across all subspans that logically lead to the hypothesis.
- **Lenient Coherence:** Measures whether the model makes at least one consistent entailment prediction for subspans that support the hypothesis.
- **Consistency Score:** A binary metric indicating whether any intermediate subspan predictions contradict the overall entailment decision.

4.6.4 Implementation

The coherence evaluation was implemented by segmenting each dialogue-hypothesis pair into subspans of increasing length. The model's predictions for each subspan were recorded alongside its prediction for the full dialogue. Logical consistency was assessed by comparing the predictions for the subspans with the prediction for the full dialogue. Finally, coherence scores were aggregated across the dataset to provide an overall assessment.

4.6.5 Role in the Overall Framework

Coherence checks play a vital role in ensuring the reliability and interpretability of the model's predictions. By identifying and addressing inconsistencies in reasoning, coherence checks enhance the robustness of the model and align its behavior with human logical reasoning. This evaluation step also provides actionable insights for further refinement of the model, such as improving alignment between token and node embeddings or augmenting the training data with additional examples that emphasize logical reasoning.

4.7 Summary

The methodology implemented in this project represents a comprehensive approach to enhancing conversational entailment through the integration of instance-specific knowledge graphs and transformer-based models. Each component of the pipeline, from entity and relationship extraction to evaluating coherence, was carefully designed to address specific challenges in conversational reasoning, enabling the model to make more accurate and interpretable predictions.

5 Results & Analysis

Model	Acc.	L. Coher.	S. Coher.
BERT	0.57558	0.33256	0.24419
BERT+KG	0.61628	0.41379	0.34302

Table 1: Accuracy (Acc.), lenient coherence (L. Coher.), and strict coherence (S. Coher.) for baseline BERT and *new* BERT with graph encodings from instance-specific knowledge graphs (BERT-KG).

Table 1 displays the resulting accuracies and coherences of the baseline BERT model and the new BERT enhanced with graphical encodings from an instance-specific knowledge graph (BERT+KG). Overall results display consistent improvement of BERT+KG accuracy and coherence over the baseline BERT, with a larger leap in coherence (8% increase in lenient coherence and 10% increase in strict coherence) than seen in accuracy (4% increase in accuracy). We find the results to be promising, seemingly confirming the hypothesis of knowledge graph enhancement creating a better model for conversational entailment. However, it cannot be understated that this improvement does come with the steep additional cost involved from computing node embedding for each instance of dialog.

5.1 Addressing Major Challenges

Not mentioned previously is the large problem of the additional computational complexity involved with creating node embeddings. This process involves calling a coreference resolution model, a named entity recognition model, an external knowledge database for each entity, and performing random walks to finally calculate the node embeddings. Running all of this once, over the full dataset of all examples takes upwards of four hours on a A100 GPU. It quickly became apparent that it would be unrealistic to generate node embeddings for each dialog every time it is needed within the model. Instead, we chose to freeze the node embedding aspect of the model, and then store node embeddings of each dialog in a saved dictionary to save repeated computation cost. This, however, results in the downside of having frozen hyperparameters of the dimension of the node embedding, the number of references from the global knowledge graph, and frozen parameters of the random walk. This left the only trainable parameters to be that of the underlying BERT model and the linear layer for embedding augmentation, but a much faster training/testing process.

5.2 Analysis

While we did see general improvement in accuracy and coherence, they were not to the large degree we may have hoped for, such as elevating BERT to the level achieved by RoBERTA/DeBERTA as reported by Storks et al. (Storks and Chai, 2021). To understand why knowledge graph enhancement only resulted in lesser improvements of the BERT model, we performed a more in depth analysis of the results.

Our first investigation was into what percentage of data was actually being augmented by graph embeddings. Under our model, there is potential that a dialog would not create an instance specific knowledge graph, in which case there would be no graph embeddings to augment the token embeddings with. The first potential scenario is if no named entities were found from within the dialog. In this case, there are nothing to query to the global knowledge bank and so an empty instance specific knowledge graph. The second scenario would be if named entities were found by NER, but these named entities had no relationships to each other and the entities did not appear within the global knowledge bank. In this instance, the knowledge graph is not a graph, but a set of completely disjunct nodes from which node embeddings cannot be created.

We found that a large proportion of dialogs within the dataset fell under one of these two categories. In fact, only 37% and 46% of instances from the training and test set respectively had node embeddings generated. This sharply reduced the potential improvement of our model to only be under these minority of scenarios where node embeddings are relevant.

Our second investigation was into the potential impact of the node embeddings in the augmented embeddings. We noticed that due to the length of node embeddings (768) created from a limited sized graph, that most values within node embeddings were very small. Smaller values comparative to the token embeddings could result in less of an impact for the nodes when the tokens are combined. As such, some of these node embeddings with minimal values have potential to be "forgotten" during the augmentation process.

6 Conclusion and Future Work

Overall, our approach of enhancing BERT with node embeddings from instance-specific knowledge graphs for the purpose of conversational entailment produced some meaningful improvements in both accuracy and coherence metrics. However, the overhead computation cost of calculating the node embeddings was exponential in comparison to the base BERT model. This extra complexity suggest that the gains of our additional effort may likely do not justify the additional cost of utilizing knowledge graphs for most applications of conversational entailment. This is especially true given the fact that more complex models such as RoBERTA are able to achieve even better performance in a far lesser amount of time.

That being said, the performance boost seen is significant enough that it could warrant extra investigation for cases under which the increased coherence is worth the additional cost. Under such, future work may include tuning of hyperparameters that we set during the creation of node embeddings. This includes factors such as the dimension of node embeddings, which we believe adjusting could increase the relevance of node embeddings within the augmentation process. Also, future work into improvement of named entity recognition/relationship extraction and an increase to the limit of references from the global knowledge graph would likely create more useful instance-specific knowledge graph and therefore improved node embeddings.

7 Work Division

All 4 team members were involved equally in each step of the implementation process, with individuals leading different aspects of the project. Team Member 1 lead the entity and relationship extraction from dialogue data using NER and relationship extraction tools. Team Member 2 lead the construction of instance-specific knowledge graphs for each dialogue. Team Member 3 lead the process of generating node embeddings for entities in the instance-specific knowledge graphs. Finally, Team Member 4 lead the integration of node embeddings with BERT and model training and tuning. All team members contributed towards the creation of the paper and presentation.

8 Code Repository

https://github.com/jack2kiwi/NLP-595-Conversational_Entailment_Instance_Specific_KG

References

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. *Preprint*, arXiv:1607.00653.

Imed Keraghel, Stanislas Morbieu, and Mohamed Nadif. 2024. Recent advances in named entity recognition: A comprehensive survey and comparative study. *Preprint*, arXiv:2401.10825.

Ciyuan Peng, Feng Xia, Mehdi Naseriparsa, and Francesco Osborne. 2023. Knowledge graphs: Opportunities and challenges. *Preprint*, arXiv:2303.13948.

Shane Storks and Joyce Chai. 2021. Beyond the tip of the iceberg: Assessing coherence of text classifiers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3169–3177, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. Graphgpt: Graph instruction tuning for large language models. *Preprint*, arXiv:2310.13023.

Chen Zhang and Joyce Chai. 2010. Towards conversation entailment: An empirical investigation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 756–766, Cambridge, MA. Association for Computational Linguistics.

A Appendix

Model	Batch	LR	Epochs
BERT	32	7.5e-6	8
BERT+KG	32	1e-5	8

Table 2: Training hyperparameters (batch size (batch), learning rate (lr), epochs) for baseline BERT model and *new* BERT+KG model