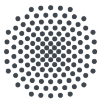


University of Stuttgart
Germany

PUI Perceptual User Interfaces

Multimodal Feature Fusion via Graph Neural Networks for Visual Dialog

Nishan Chatterjee

A Thesis presented for the degree of M.Sc. in Computational Linguistics

Examiners:

Prof. Dr. Andreas Bulling

& Jun. Prof. Dr. Carina Silberer

&

Supervisor:

Adnen Abdessaied

Institute for Visualization and Interactive Systems
Human-Computer Interaction and Cognitive Systems
University of Stuttgart
30 March 2022 - 30 December 2022

Erklärung (Statement of Authorship)

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und dabei keine andere als die angegebene Literatur verwendet habe. Alle Zitate und sinngemäßen Entlehnungen sind als solche unter genauer Angabe der Quelle gekennzeichnet. Die eingereichte Arbeit ist weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen. Sie ist weder vollständig noch in Teilen bereits veröffentlicht. Die beigefügte elektronische Version stimmt mit dem Druckexemplar überein [1].

Nishan Chatterjee

[1]Non-binding translation for convenience: This text is the result of my own work, and any material from published or unpublished work of others which is used either verbatim or indirectly in the text is credited to the author including details about the exact source in the text. This work has not been part of any other previous examination, neither completely nor in parts. It has neither completely nor partially been published before. The submitted electronic version is identical to this print version.

Abstract

One of the main challenges in Computer Vision and Natural Language Processing is enabling machines to have a conversation with humans about visual data and to understand and interpret the meaning and context in a conversational dialog. Towards this, researchers have proposed the Visual Dialog Challenge dataset which aims to create a basis on which multi-modal language and vision models can be trained and fine-tuned to reach this milestone. One promising approach apart from pre-trained language and vision models is the use of graph neural networks to model the different implicit relations between objects in an image and dialog. However, they often neglect the importance of coreference relations in dialog and the representation of an image under the context of full-question comprehension based on past dialogs. To improve this Chen et al. (2021) [1] propose a novel architecture that uses three sequential graphs to explicitly model relations between the dialog, question, and image which this thesis replicates. The GoG approach is also shown to outperform strong baseline implementations for the task. However, the explicit nature of relation modeling might not be the best approach to perform relation modeling. To address this, this thesis proposes an implicit relation modeling between the three sequential graphs using a master node using graph attention. Experimental results indicate a good starting point, however, it also advocates the need for further experimentation to confirm the validity of this novel architecture.

Contents

1 Introduction

Visual Dialog is a research area in artificial intelligence that focuses on enabling a machine to have a conversation with a human about images and videos. It involves developing algorithms that can understand and respond to natural language queries about visual content and can generate appropriate responses based on their understanding of the visual content and the context of the conversation. The goal of visual dialog is to enable a machine to have a more human-like conversation about visual content, and to improve its ability to understand and interpret visual information. It is hoped that by enabling a machine to have a visual dialog with a human, a machine will be able to improve its ability to understand and interpret visual information and assist humans more effectively in tasks that require visual content.

Das et al. (2016) [2] define the task of Visual Dialog that requires an agent to hold a meaningful dialog with humans in a natural and conversational language about the content of an image. More concretely, for a round(r) of question-answering, given an image(I), a dialog history (or simply history) consisting of a sequence of question-answer ($QA_{\text{round 1 to r-1}}$) pairs from the previous rounds and an image caption (C), the task for the machine is to answer the question(Q_r) for the current round. The task requires reasoning over both the context that has been stated in the Dialog and a sense of world knowledge that underlies it. Therefore, this task can be considered as a visual analog of the Turing Test [3].

Prior work with neural models trained in isolation [2] have shown that there is significant scope for improvement with Das et al. (2016)'s work serving as a baseline. Murahari et al. (2020) [4] highlight how transfer learning can benefit the task by using pre-trained models like ViLBERT [5] trained on related vision-language datasets (Visual Question Answering [6] and Conceptual Captions [7]). These models are transferred to Visual Dialog [2] for fine-tuning.

However, Chen et al. (2021) [1] shows in their Relation-Aware Graph-over-Graph Network for Visual Dialog showed how using three sequential graphs can result in the highest performance among all previous approaches except some pretraining-based models [8]. More specifically, the three sequential graphs that Chen et al. (2021) [1] use are: (1) a history-graph (H-graph) which aims to capture the co-reference relations in dialog history, which is then embedded into (2) the history-aware question graph (Q-graph) by the use of a history-aware attention mechanism [9] to inject relation-aware semantically rich information of the history into this question graph. This information is multi-modally embedded to form a question-aware image graph (I-graph). The three graphs are jointly

trained with a multi-modal fusion training approach [10] to perform the task.

This thesis replicates the work of Chen et al. (2021) [1] and highlights a novel method of performing Visual Dialog extending on the work of Chen et al. (2021) [1]. The novel implementation is performed with an implicit encoding style (as compared to Chen et al. (2021)'s [1] explicit embedding method) on each of the three graphs (history-graph, question-graph, image-graph) to be jointly trained using the multi-modal fusion training approach [10]. The following section 2 highlights both techniques in greater detail.

The primary goals of this thesis are to reimplement the work of Chen et al. (2021) [1] (since their code base is not open-sourced) and to implement the novel multimodal feature fusion via graph neural networks approach listed under section 2.1.3 and measure its performance against previous state-of-the-art baselines. The thesis also includes optional goals of performing ablation studies to highlight the significance of context awareness by injecting semantic information from the history graph into the question graph and from the history-aware question graph into the image graph.

The section novelty and contributions 2 explains the architectural overview of the baseline [1] along with the novel architecture proposed in this thesis. The section background and related work 3 highlights the different tasks in the field of natural language processing (NLP) related to the task of Visual Dialog along with the approaches taken to solve some of these related tasks. The materials section 4 explains the different components of the data that were used to solve the task along with providing additional analyses and insights into the data used. The methods section 5 explains in detail all the components of the architecture 2 used to solve the task of Visual Dialog. The experiment section 6 underlines the experimental setup, choices of parameters and hyperparameters, and the results from the experiments. The discussion 7 section is used to describe and provide an analysis of the experimental findings along with their significance in relation to the research questions. Finally, the future work 8 section highlights the additional experiments that can be performed in relation to the methodology and experimental observations.

2 Novelty and Contributions

The primary objective is to replicate the architecture of Chen et al. (2021) [1] since the code-base is not open-sourced, along with creating the novel master-node multimodal feature fusion approach. This section details the architectural overview of both approaches along with highlighting the key differences between them.

2.1 Architectural Overview

Figure 1 highlights the transformation of the dialog history, questions, and images from the Visual Dialog dataset [2] into feature graphs. To recap, the task is to play the role of an Answerer to a question given an image, and a dialog history. Each image contains 10 rounds of possible context-dependent questions.

The authors Chen et al. (2021) [1] propose to build a hierarchy of graphs where the lowest hierarchy contains the dialog history. This is motivated by how the Visual Dialog dataset was collected (mentioned in further detail in 4.2). To summarize, during the data collection process, for the first round of questions, the Questioners has access to only the caption, whereas the Answerer had access to the image, the question, and the caption. For the second question round, the history contains the caption, as well as the first question-answer pair, and so forth.

Since the questions contain the semantic information required to identify and answer questions about the objects in the image, it forms the second level of the hierarchy for context embedding. This also allows context mapping from coreference relations present between the question and the dialog history to be embedded into the question graph.

The dialog history and question information are then relayed onto the image graph. The two different approaches 2 and 3 highlight the process in which the contextual information is embedded into the graph hierarchy.

Finally, the three context-aware graphs are sequentially conveyed into a multimodal fusion model for training. The details of the components of the architectures have been discussed in further detail under section 5.

2.1.1 Graph Initialization

The dialog history is represented by a graph where a node refers to either a caption or a question-answer pair. The caption and {question? answer} pairs are represented by Glove embeddings [11] passed through an LSTM layer [12]. The question graph nodes refer to

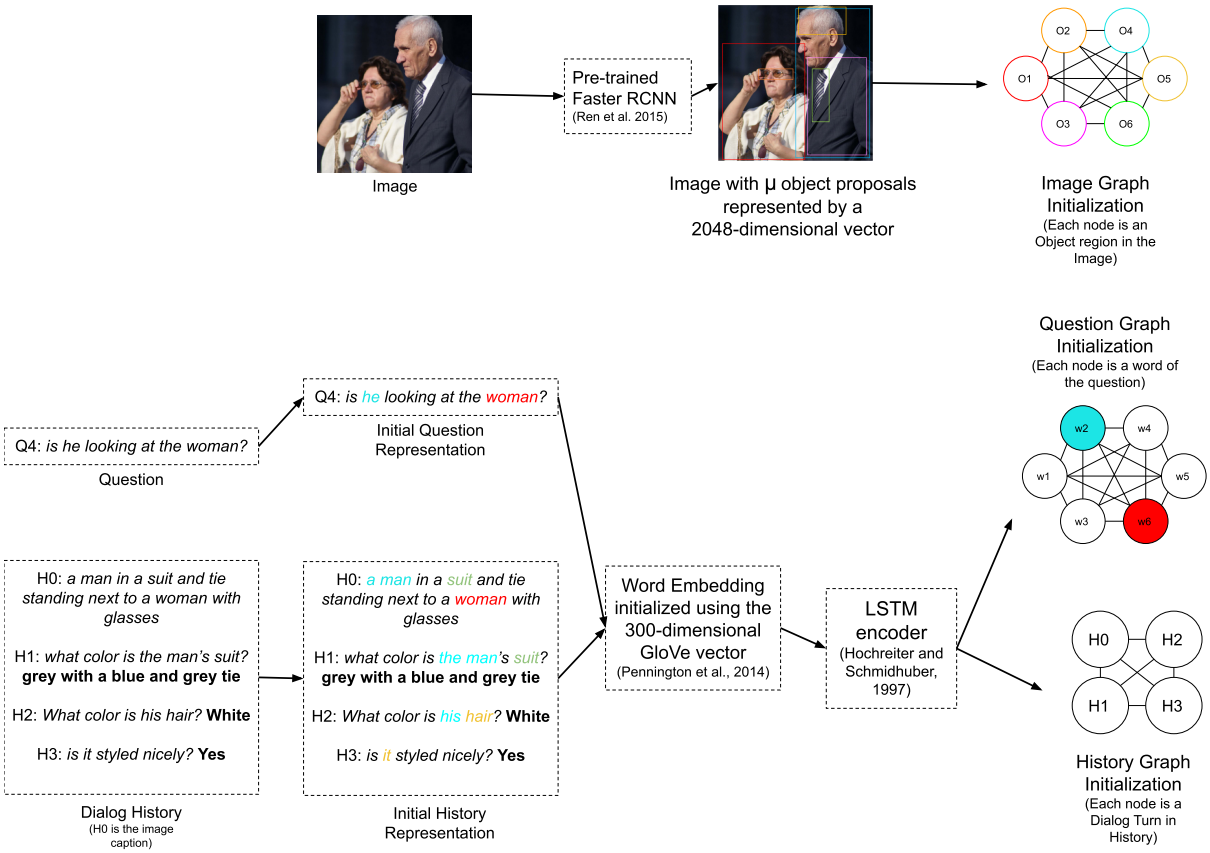


Figure 1: Feature representation of dialog history, questions, and images into feature graphs

the words present in the question. Finally, the images are represented by identifying the object proposals using a pre-trained Faster-RCNN model [13] where each node in the image graph relates to an object proposal detected from the image.

2.1.2 GoG: Relation-aware Graph-over-Graph Network

The dialog history graph is pruned using coreference resolution [14] where two nodes share an edge if a positive coreference cluster exists between the two nodes. The pruned dialog history graph is channeled through a graph attention network [15] to assign the updated node importance in relation to its neighbors. This updated graph is then propagated through a gated graph sequence neural network [16] to generate a pooled embedding representation of the history graph using global attention. This is used both in the sequential multi-modal fusion model for training, as well as the context vector to be used in the question graph.

The question graph is pruned using a neural parser [17]. Each node in the pruned question graph is concatenated with the context vector generated from the dialog history graph to create a history-aware question graph. The history-aware question graph is similarly

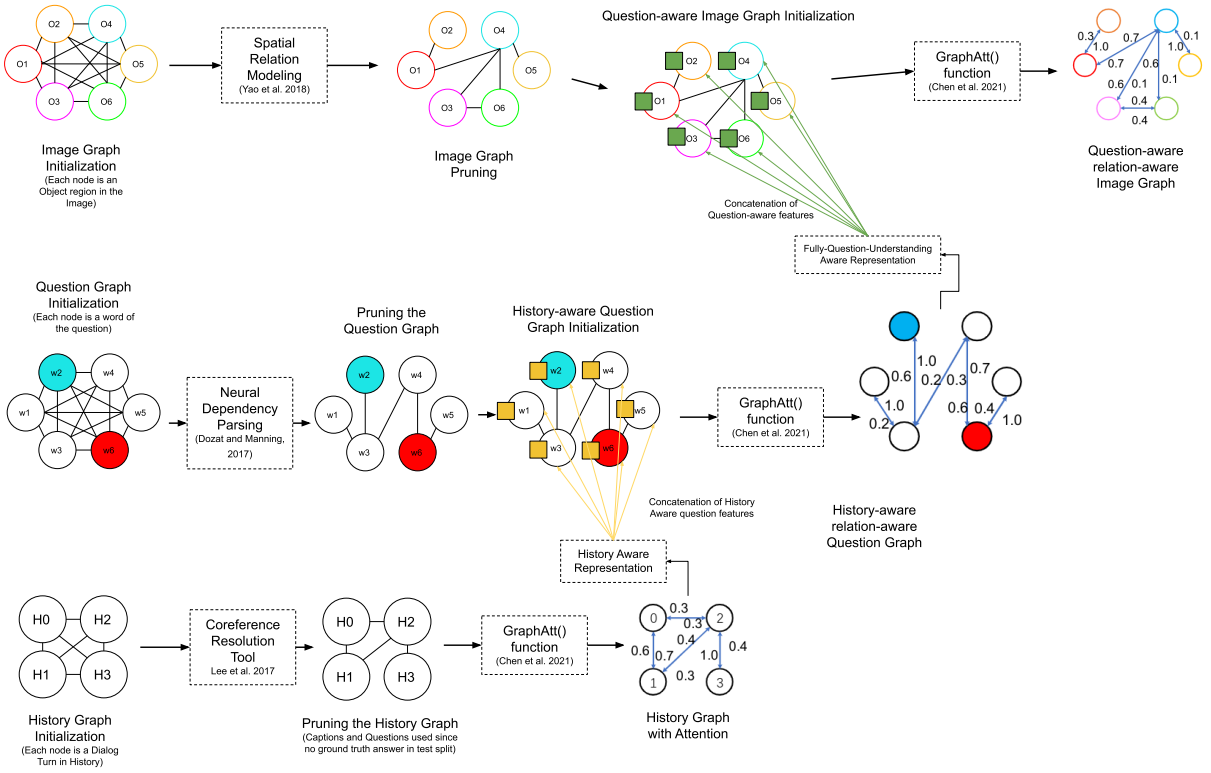


Figure 2: Relation awareness architecture for the GoG: Relation-aware Graph-over-Graph Network

channeled through a graph attention network [15] and then subsequently through a gated graph sequence neural network [16] to generate the pooled embedding for history-aware question graph using global attention. This is used both in the sequential multi-modal fusion model for the training step, as well as a context vector to be used in the image graph.

The image graph is pruned based on a spatial relation modeling [18]. Each node in the pruned image graph is concatenated with the context vector generated from the history-aware question graph to create a history-aware-question-aware image graph. The updated graph is channeled through a graph attention network [15] and then subsequently through a gated graph sequence neural network [16] to generate the pooled embedding for the image graph using global attention. This embedding is used in the sequential multi-modal fusion model for the training step. This serves as the baseline architecture for the thesis [2].

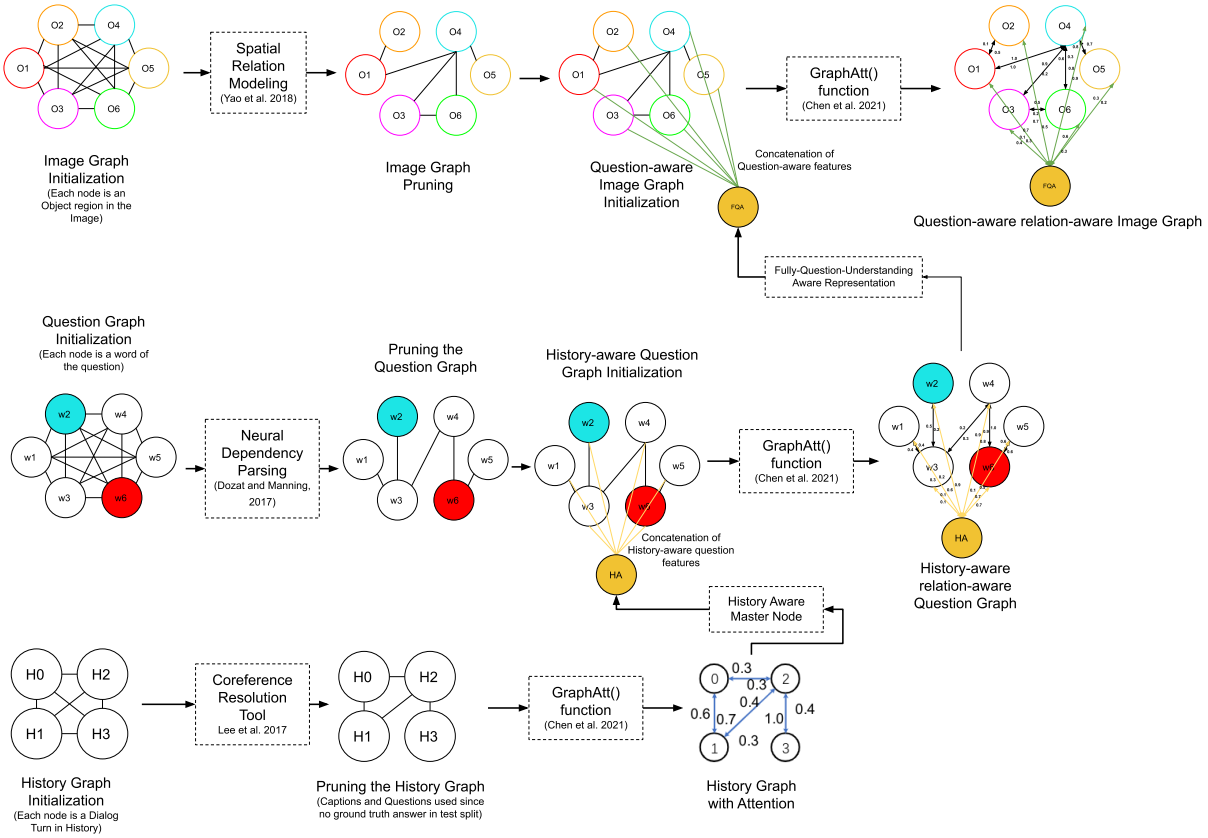


Figure 3: Relation awareness architecture for Master Node: Multimodal Feature Fusion

2.1.3 Master Node: Multimodal Feature Fusion

The key difference between the novel 2.1.3 approach when compared to the GoG baseline implementation 2.1.2 is that the context vectors are not attached explicitly to the upper hierarchy graph nodes. Instead, they are attached as a feature node to each node in the upper hierarchy graph and the graph attention network is used to assign the importance of this node with respect to all its neighboring nodes.

The dialog history graph is created in the same way as the GoG implementation 2.1.2. The initial history graph is pruned using coreference resolution [14] where two nodes share an edge if a positive coreference cluster exists between the two nodes. The pruned dialog history graph is channeled through a graph attention network [15] to assign the updated node importance in relation to its neighbors. This updated graph is then propagated through a gated graph sequence neural network [16] to generate a pooled embedding representation of the history graph using global attention. This is used both in the sequential multi-modal fusion model for training, as well as the context vector to be used in the question graph.

[2]The key insight for the baseline 2.1.2 approach is that the context vectors are concatenated explicitly to the upper hierarchy graph nodes.

Similarly, the question graph is pruned using a neural parser [17]. Each node in the pruned question graph is connected with the context vector (generated from the dialog history graph) as a graph node to create a history-aware question graph. The history-aware question graph is channeled through a graph attention network [15] to assign the updated node-importance in relation to its neighbors and then subsequently through a gated graph sequence neural network [16] to generate the pooled embedding for history-aware question graph using global attention. This embedding is used both in the sequential multi-modal fusion model for the training step, as well as the context vector to be used in the image graph.

The image graph, similarly, is pruned based on a spatial relation modeling architecture [18]. Each node in the pruned image graph is connected with the context vector (generated from the history-aware question graph) as a graph node to create a history-aware-question-aware image graph. This graph similarly is channeled through a graph attention network [15] and then subsequently through a gated graph sequence neural network [16] to generate the pooled embedding of the image graph using global attention. The pooled embedding is then used in the sequential multi-modal fusion model for the training step. This serves as the novel architecture for the thesis [3].

[3]The proposed architecture stems from the reasoning that all nodes in a graph do not carry the same importance [15]. Graph attention networks help to identify the important nodes from a graph. Similarly, the context vector should in theory not be equally important to each node in a graph. So instead of an explicit context embedding, an implicit relation modeling followed by performing graph attention should improve the quality of the context being modeled.

3 Background and Related Work

The scope of the thesis can be deconstructed into two parts: the task in which the problem exists, and the approaches taken to solve it. This section highlights the domain of Visual Dialog (VisDial) and its related areas of research. Additionally, this section also investigates the different models and architectures that have been used to solve some of the related tasks [4] Visual Dialog challenge. Furthermore, the section also highlights the models that have been used for Visual Question Answering (VQA) in greater detail as research in Visual Dialog is a direct extension of VQA and therefore benefits from research undertaken in the scope of VQA.

3.1 Related Tasks

The task of Visual Dialog lies at the intersection of solving language and vision challenges. Some prominent language and vision tasks include: image captioning (describing images with syntactically and semantically meaningful sentences) [19, 20, 21, 22], video description (the automatic generation of natural language sentences that describes the contents of a given video) [23, 24, 25], coreference grounding in the scope of text-to-image (using visual scene parsing information to enrich natural lingual descriptions) [26, 27, 28, 29, 30, 31], and visual story-telling (modeling concrete descriptions, as well as figurative and social languages towards a more human-like understanding of subjective expression comprehension and grounded event structures) [32, 33]. However, these tasks involve at most a single natural language interaction step as only multi-modal information from one modality gets translated into another [5].

The task of Visual Question Answering [6] involves free-form and open-ended questions in natural language given an image, where the task is to provide an accurate natural language answer. The questions different parts of an image including the background details as well as the underlying context of an image. This requires a successful system to formulate a nuanced understanding of an image and complex reasoning abilities than simply producing generic image captions. The work done by [34, 35, 36, 37, 38, 39, 40, 41, 42] showcase how VQA can be solved. In contrast to the tasks mentioned earlier, VQA includes a natural language interaction step, however, there is still no discourse component involved. Upon a closer inspection of the questions and the answers present

[4] since the models that are used to tackle the challenge are often extensions of work in related tasks

[5] [28, 30] involve a natural language interaction step where coreference resolution takes place similar to how humans would identify a particular object in an image being referred to, for e.g., "the one in a red shirt" or "the dog on the left".

in the VQA dataset, many answers are open-ended which comprise only a few words or a closed set of answers provided in a multiple-choice format 5. In contrast, VisDial questions are longer and more descriptive: 2.9 words mean-answer-length in VisDial vs 1.1 words mean-answer-length for VQA [6].

VQA also tends to focus more on imaginative questions like "What is ...", "Is there ...", "How many ...", and "Does the ..." and less on task-driven and concrete, a limited set of visual concepts, for e.g., color or location [6].

VisDial [2] improves upon VQA [6], Baidu mQA [35], and Visual 7W [43] by accounting for the visual priming bias [44] as participants saw the image they were asking questions about in [6, 35, 43]. As analyzed by [34, 37, 42], subjects tended to ask 'Is there a clock-tower in the picture?' on pictures actually containing clock towers. This makes language-only models perform exceptionally well on the VQA challenge. This trend gets especially prominent in the analysis of questions like "Do you see a ...?" where even blindly answering "yes" without reading through the rest of the question or looking at the image associated with the question resulted in an average accuracy of 87%! This demonstrates how models trained on VQA can tend to have an inflated sense of progress [37, 42]. However, in VisDial, Questioners did not see the image which reduces this bias.

Other related works to Visual Dialog include the Visual Turing Test [45] which is a restrictive form of the Turing Test that involves answering binary questions following a fixed template. The dataset [45] also contains only street scenes that have limited variety. In comparison, Visual Dialog is a free-form, open-ended conversation-oriented dataset with two orders of magnitude larger data (approximately 140k images from the COCO dataset [46] with 10 rounds of question-answer pairs for each image as compared to approximately 2.6k images for the Visual Turing Test dataset by Geman et al. (2015) [45]).

The task of Visual Dialog is also related to other uni-modal question-answering tasks like text-based question answering [47], machine reading comprehension [48], and conversational modeling [49]. The task of Visual Dialog can be seen as an amalgamation of reading comprehension and visual question answering. In VisDial, the machine must comprehend the meaning of the past dialog history to contextualize the question and answer about what it sees in the image. The conversational nature also makes sure that a question, if present in the previous conversation, would not be repeated.

[6] During the data-collection phase of VQA, subjects were instructed to ask questions that required an image to answer, their task description being "We have built a smart robot. It understands a lot about images. It can and all the objects, it knows where the objects are, it can recognize the scene (e.g., kitchen, beach), people's expressions and poses, and properties of objects (e.g., color of objects, their texture). Your task is to stump this smart robot! Ask a question about this scene that this smart robot probably can not answer, but any human can easily answer while looking at the scene in the image." [6]

3.2 Related Models

Pre-trained language models have gained significant attention in recent years due to their ability to perform well on a variety of natural language processing tasks, including translation, summarization, and question answering [50, 51, 52]. These models are trained on large amounts of data and are able to capture the statistical patterns and regularities of the language. This allows them to generalize to new examples and achieve state-of-the-art performance on many benchmarks.

One approach to solving the Visual Dialog challenge is to use pre-trained language models (to resolve the natural language aspect of the task) to generate responses to the questions and comments of the human conversational partner. These models can be fine-tuned on a dataset of dialogues related to images, in order to better capture the language patterns and characteristics of such dialogues [53]. However, there are limitations to the capabilities of these models. One significant limitation is their ability to robustly capture the latent relationships between concepts, which is a key aspect of reasoning. Reasoning involves the ability to understand and draw logical conclusions from information, and is an essential component of intelligent behavior. It requires the ability to identify and understand relationships between concepts and to use this understanding to make informed decisions. To improve the reasoning capabilities of Large Language Models (LLM), Wei et al. (2022) [54] demonstrates the use of chain-of-thought prompting techniques (a series of intermediate reasoning steps that can be undertaken by an LLM) to significantly improve their ability to perform complex reasoning tasks.

Additionally, LLMs also struggle with reasoning tasks because they are primarily trained on large amounts of data without additional fine-tuning. As shown by the work of McCoy et al. (2019) [55], natural language inferencing (NLI) models may rely on three potentially flawed heuristics: the lexical overlap heuristic, the subsequence heuristic, and the constituent heuristic 4. According to the authors [55], models trained on the multi-genre natural language inference task, including the state-of-the-art BERT model [56] that performs poorly on the Heuristic Analysis for NLI Systems (HANS) challenge dataset [57]. This suggests these models have indeed adopted these heuristics, that there is further room for improvement, and that there is still potential for improvement in this area of research.

One of the workarounds (without the use of LLMs) to the problem of relationship modeling in unstructured data can be solved by the use of Knowledge Graphs (KG) like Freebase [58], Wikidata [59], and ConceptNet [60] that capture external knowledge between entities explicitly using triplets that model the relationship between various entities. Ren et al. (2020) [61] show the significant role KG's can play in structured reasoning and

query answering by embedding KG entities as well as the query as a closed region rather than a single point in the vector space 22.

Additionally, Yasunaga et al. (2021) [62] demonstrate how using large language models (LLMs) and graph neural networks can be used to estimate the importance of nodes in a knowledge graph (KG) relative to a given question answering (QA) context. Their approach (jointly reasoning over the QA context to form a joint graph) improves the performance of QA tasks in terms of relevance scoring.

However, it has been difficult to extend these architectures to enhance the reasoning abilities of AI agents to a much more general question-answering setting (QA), where questions and answers are expressed in natural language and cannot be easily mapped to strict logical queries. This requires the proper integration of the information and constraints provided by the QA with knowledge from a knowledge graph (KG) in order to form a complete and accurate answer. Mihaylov and Frank (2018) [63], Lin et al. (2019) [64], and Feng et al. (2020) [65] showcase a number of ways to leverage both modalities of structured and unstructured information to improve reasoning, the methods typically fuse the modalities in a shallow and non-interactive manner as both information is encoded separately and is fused at the final output step.

These methods showcase their advantages and disadvantages in uni-modal (text) data which is one part of the Visual Dialog Challenge. However, to perform Visual Dialog, an AI system must be capable of reasoning multi-modally (natural language (text) and images). Additionally, it must also reason over the complex relations among visual content, dialog history, and current questions. The VQA challenge has seen the use of relation-aware attention-based graph neural networks where each image is encoded into a graph and models the multi-type inter-object relations to learn the question-adaptive relations [66]. This outperforms all previous state-of-the-art approaches for the VQA challenge demonstrating the efficacy of using relation-aware graph data structures. Liang et al. (2021) [67] also showcase how in a language-guided graph neural network, question-answering can be translated into multiple iterations of a message-passing operation among graph nodes.

As mentioned earlier, attention-based pre-trained models have served as the backbone of previous mainstream approaches in Visual Dialog [68, 69, 70, 71, 72]. Lu et al. (2017) [68] introduced a history-conditioned image attentive encoder for representing the question, the question-attended history, and the attended image. Wu et al. (2018) [69] proposed a sequential co-attention encoder, in which each input feature is co-attended by the other two features in a sequential manner. Gan et al. (2019) [71] and Chen et al. (2020) [73] employed a multi-step reasoning process based on dual attention to answer a series of questions about an image in their respective approaches. Guo et al. (2019) [74], Agarwal

et al. (2020) [75], and Nguyen et al. (2020) [10] utilize a multi-headed attention mechanism to compute the multi-modal interactions between different inputs. However, again, these methods tend to focus on the most discriminative information, potentially overlooking other rich complementary clues such as the relationships between objects in an image [1].

This is where recent research [74, 76, 77, 78, 79] in the field of visual dialog has focused on exploring higher-level semantic representations of images and dialog history, often using graph-based structures to model these elements. While the use of graph-based structures has shown promise, these models also have not been successful in explicitly capturing complex relationships within visual content or textual contexts, or the relationships between these elements (according to Chen et al. (2021) [1]). Specifically, Chen et al. (2021) [1] claims there is a need to better understand the role of coreference resolution, the intention behind a question and its syntax and history, and the relationship between fully understanding a question and understanding an image. These are important areas that warrant further exploration. This leads to baseline architecture [1] and the novel architecture of the Master Node implementation 2.1.3 described in this thesis.

4 Materials

This section outlines the datasets that are used to perform the task of Visual Dialog. The section also includes insights into the different aspects of the datasets used, provides analyses on the quality of the features that can be extracted to solve the task, and some of the problems that may occur during feature engineering.

4.1 Dataset - Microsoft Common Objects in Context

The Microsoft Common Objects in Context dataset (MS COCO or COCO) [46] was compiled for the advancement of the state-of-the-art object recognition with the goal of contextualizing these objects in the broader question of scene-comprehension. The images reflect complex everyday scenes containing common objects (91 object types) that can be found in their natural context. The dataset contains a total of 2.5 million labeled instances in 328k images having roughly 5 captions each. The images in the dataset are 2.5M object instances labeled with over 5.5M hours of annotations by crowd-workers using the framework provided by Bell et al. (2013) [80]. Furthermore, the dataset contains over 1.5M captions on these images with the objectives of (1) describing all the important parts of a scene, (2) omitting unimportant scene details, (3) ensuring that scene description pertain to the current image and not borrowing from scene comprehension and prediction of near-future or near-past activities, (4) including details that cannot be comprehended solely based on the information available in the picture, and (5) having at least 8 words for each caption [81].

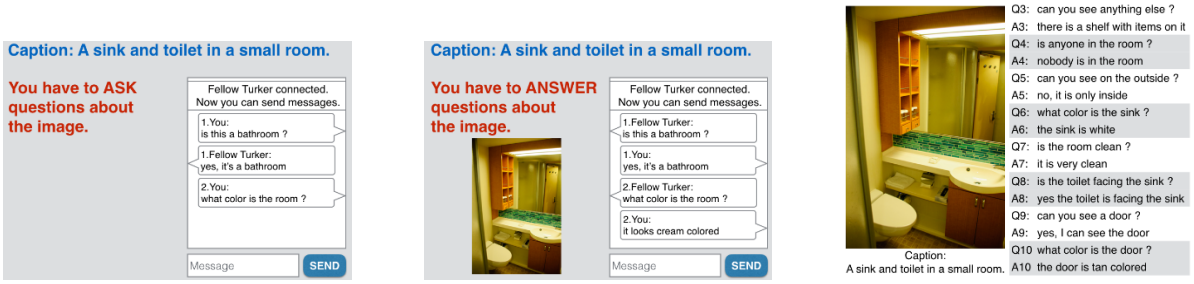
4.2 Dataset - VisDial 1.0

The dataset [2] was collected by using a chat interface hosted on Amazon Mechanical Turk (AMT) [7]. The images used for collecting the Visual Dialog data were taken from the COCO dataset [46], 4.1. This results in the VisDial dataset having more than 123k distinct images from COCO, each of which contains 10 rounds of dialogs (Question-Answer) pairs, resulting in over 1.23M Dialogs, 376k unique Questions, and 337k unique Answers.

The benchmark for good data collection required the dialogs to have a temporal sense of continuity, a distinct grounding in the image, and mimic natural exchanges in conversation. To elicit such responses, 2 crowd-workers were paired to chat with each other in

[7] Amazon Mechanical Turk is a crowd-sourcing website that businesses use to remotely hire people (also known as crowd-workers) to perform discrete on-demand tasks that today's computer systems are currently unable to do autonomously.

real-time on AMT, each crowd-worker having been assigned a specific role. The crowd worker in the role of the Questioner sees only a single line of text describing an image (the captions from COCO) with the image being hidden from the questioner. The task of the Questioner was to ask questions about this hidden image to figure out the scene more accurately. The second crowd worker who is assigned the role of the Answerer, unlike the Questioner, sees both the image and the caption to answer the question asked by the Questioner. Figure 4 shows the chat interface for the first few rounds of dialog [8].



(a) Questioner point-of-view

(b) Answerer point-of-view

(c) Example dialog entry from VisDial

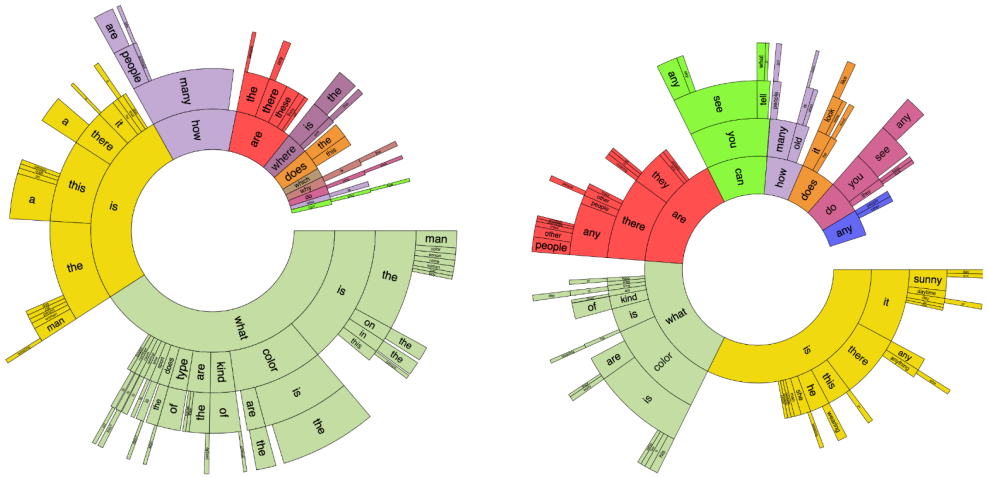
Figure 4: Data collection for VisDial via the AMT chat interface [2]

As mentioned in 3.1, when analyzing the answers collected on VisDial, an interesting category of answers emerge. These include 'I think so', 'I can't tell', or 'I can't see' answers that express a sense of doubt, uncertainty, or lack of information. The inability of the questioner to view the image leads to the possibility that some questions, despite being contextually relevant, may not be able to be definitively answered based on the image alone. This also indicates on the richness of building human-like AI that refuses to answer questions it doesn't have enough information to answer, along with being able to model uncertainty better [82].

The resulting dialog dataset captured 4 can be interpreted in the following manner: Consider an entry from the dataset for an Image with the Caption "a man in a suit and tie standing next to a woman with glasses.", where QA_1 is "what color is the man's suit? grey with a blue and grey tie", QA_2 is "what color is his hair? white", and so on until round QA_{10}. For Q_1 ("what color is the man's suit?"), dialog $\text{history}_{\text{round 1}}$ would consist of the set {caption}. Similarly, for Q_2 ("what color is his hair?"), dialog $\text{history}_{\text{round2}}$ would be

[8]The authors of Das et al. (2016)[2] also piloted an alternative setup where the Questioner had access to a blurred version of the image. However, these result in questions that are in essence an attempt at blob/blur recognition. This violated the naturalness of the questions asked, and therefore the authors chose to opt for the caption-based method for the full-scale data collection. The authors [2] also make their annotation system publicly available since the backend messaging and data-storage infrastructure necessary for a two-person live chat is not currently possible on AMT.

the set {caption, QA_1}.



(a) VQA questions (b) VisDial questions

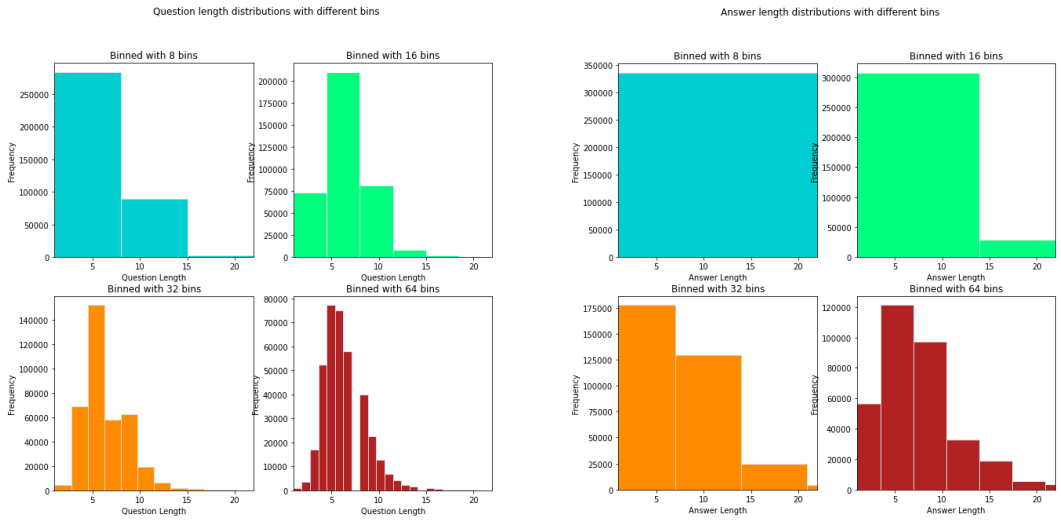
Figure 5: The distribution of first n-grams for VQA and VisDial questions. The word order starts from the center and radiates outwards, and the arc length is proportional to the number of questions containing the word mentioned in the plot [2]

Figure 5 show the distribution of the questions in VisDial when compared to the questions in VQA. In VisDial, language is the result of a conversation and therefore often includes pronouns like "he," "she," "his," "her," "it," "their," "they," "this," "that," and "those." Around 38% of questions, 19% of answers, and almost all (98%) dialogs contain at least one pronoun, indicating that a machine will need to effectively handle pronoun references in order to be successful on this task. Pronoun usage tends to be low in the initial round of a conversation and increases in frequency as the conversation continues.

Finally, figure 6 show the different groups questions and answers can be binned into, which can be used for hyper-parameter tuning towards finding the optimal question and answer lengths for padding and truncation.

4.3 Data Subsets

As previously mentioned, the implementation of both the baseline model by Chen et al. (2021) [1] and the novel approach involves the use of three toolkits to extract features that can be used to construct the three feature graphs: (1) a coreference resolution tool [14] to obtain the coreference relations between the different rounds of history, (2) a neural parser [17] to get the dependency relations of the questions, and (3) the object proposals in an image using Faster-RCNN model [13]. Two new randomized subsets of VisDial Train 1.0 were created in order to test the successful completion of the entire



(a) Question length distribution

(b) Answer length distribution

Figure 6: The figures show the average question and answer length in the VisDial training dataset

architecture: (a 1 percent and a 10 percent of VisDial Train 1.0). This allows for shorter compile times on generating the embeddings necessary to create the three feature graphs (since using the neural parser and Faster-RCNN takes significant compute time), as well as running experiments to test the performance of the fusion model [10]. The table 1 shows the number of unique images, questions, and answers each of the subsets contains. On average, the complete VisDial training dataset contains 3.05 unique questions for each image, while the 10% subset and the 1% subset contain 4.71 and 6.81 unique questions for each image respectively. The unique answer distribution (including the 100 candidate answers) follows an even greater variety with the complete VisDial training dataset 1.0, 10% subset, and 1% subset having 2.74, 27.38, and 217.37 unique answers respectively. This indicates that even training on smaller subsets should give the model enough variety to not overfit on the data. However, further analysis is required to observe how model performances translate when scaling between the subsets.

Dataset Name	Unique Images	Unique Questions	Unique Answers
VisDial 1.0 Train	123287	376082	337527
10% Subset	12329	58069	337517
1% Subset	1233	8402	268020

Table 1: A comparison of the unique images, questions, and answers across three randomized subsets. The unique answers include the list of 100 candidate answers the model has to evaluate.

4.4 Caption Analysis

While state-of-the-art text-to-image generation models [83, 84, 85, 86] are capable of generating realistic images on datasets of birds, flowers, room interiors, faces, etc., they struggle with datasets like COCO [46] due to the fact that COCO contains several objects within a single image where the subjects are not always well centered. While this makes the COCO dataset better reflect real-world use cases, it also contains gaps in knowledge. A caption for an image of a flower can usually describe most of the relevant details of the flower as can be seen in figure 7 [87].

(a) Caption

(b) Image generated from caption

Figure 7: Caption and its corresponding generated image by the StackGAN model [85]

However, the captions that are found in COCO [46] can have similar captions for very different images. Figure 8 highlights such an example where the relevant details about the foreground and background. As the authors [87] highlight, there is a possibility that two similar COCO captions could correspond to very different images, making it challenging for a model to accurately determine which objects in the image correspond to which words in the caption. This complexity, combined with the limited amount of available data that pairs images with captions, means that the model may not always be able to effectively understand the relationship between the two. As a result, a caption may not always be a reliable descriptor of an image [9]. However, as Fang et al. (2014) show, it is indeed possible to augment these human-generated image descriptors [90, 91].

The captions used in VisDial are taken randomly from the COCO [46] 2014 training and validation datasets. Each image in COCO has roughly 5 sets of alternative captions [12]. The authors of VisDial [2] choose one caption for each image at random 9 from

[9] This can also be observed in other scenarios involving round-trip translation systems (similar to the work of Huang et al. (2020)[88] and Ahmadnia et al. (2019) [89]) that cause image →[image-captioning models] →caption →[text-to-image generation models] →image to lose semantic information that human-generated captions are not able to capture.

[12] Out of roughly 123k images with captions from COCO, 324 images have 6 captions, and 4 images

(a) Caption: A flock of birds flying in a blue sky [10]

(b) A flock of birds flying in an overcast sky [11]

Figure 8: Two very different images from COCO[46] sharing a very similar caption [87]

COCO captions and preprocess them to generate the captions for the VisDial dataset. The preprocessing includes removing question marks and special characters and replacing numbers with words (for example, "a clock that's big hand is pointing at 10 while the small is on *22*" from COCO is converted into "a clock that's big hand is pointing at ten while the small is on *twenty-two*" for VisDial).

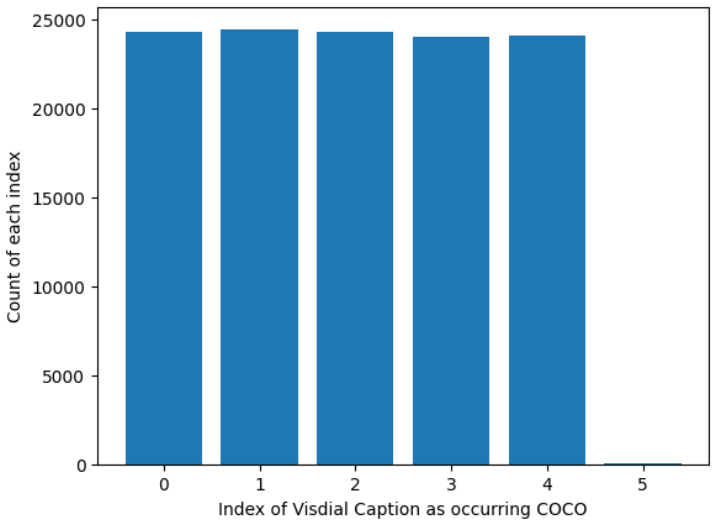


Figure 9: Index of VisDial captions in COCO

4.5 Dialog History

Agarwal et al. (2020) [75] found that co-attention models [13] that explicitly incorporate dialog history tend to perform better than those that do not, achieving state-of-the-art

have 7 captions.

[13]Co-attention models associate keywords in questions with key objects in images for visual question answering tasks [92]

results in terms of the NDCG score (72% on the validation set) [14]. According to their [75] findings, the VisDial dataset includes relatively few examples that require dialog history. In contrast, other visual dialog tasks, such as GuessWhich? by (Chattopadhyay et al., 2017) [94] and GuessWhat?! (De Vries et al., 2017) [95], take place in a goal-oriented setting, which tends to result in data that includes more natural dialog phenomena, according to Schlangen (2019) [96]. Yang et al., (2019) [97] also suggest that there is very little evidence to suggest that dialog history is important for tasks such as visual dialog, indicating that the challenge of accurately capturing visual dialog phenomena as still an open problem.

Furthermore, Agarwal et. al. (2020) [75] propose a new challenge task called VisDial-Conv, a subset of the VisDial validation set consisting of 97 dialogs where crowd-workers were asked to identify single turns with dense history annotations that explicitly required historical information.

[14]Normalized Discounted Cumulative Gain (NDCG) is a measure of the ranking quality of a set of items. It is commonly used in information retrieval and recommendation systems to evaluate the performance of a ranking model. NDCG is calculated by first ranking a set of items according to some criterion, and then computing a score based on the relevance of each item to the query or task at hand. The score for each item is discounted based on its position in the ranking, with higher-ranked items receiving a higher discount. This is done to reflect the fact that people generally value items that are ranked higher more highly than those ranked lower. The NDCG score is then calculated by dividing the discounted cumulative gain by the maximum possible discounted cumulative gain, which is the discounted cumulative gain that would be obtained if the items were perfectly ranked according to their relevance. This normalization step ensures that the NDCG score is between 0 and 1, with higher values indicating better-ranking quality. NDCG is often used in combination with other evaluation metrics, such as precision and recall, to provide a more comprehensive view of the performance of a ranking model [93].

5 Methods

The task of Visual Dialog requires the modeling of two modalities of information: Text and Images. To solve this, three graphs are generated: a history graph, a question graph, and an image graph. These graphs are designed to capture different aspects of the conversation and the image and can be used to help the dialog system understand and respond to the user's input.

The history graph captures the context of the conversation up to a given point in time. It can be used to keep track of the questions that have been asked and the responses that have been given, as well as any other relevant information about the conversation. The question graph represents the current question being asked by the user. It can be used to understand the meaning of the question and identify any relevant information in the image that might help to answer it. The image graph represents the content of the image itself. It can be used to identify objects, scenes, and other visual elements in the image, and to understand their relationships to one another.

This section provides a detailed explanation of all the components used for creating the baseline and the novel implementation mentioned under section 2, along with additional models used for training and experimentation. This includes the creation of the three feature graphs, their initialization process, how to generate the node encodings, and how to prune the graphs using coreference (for the history graphs), neural parsing (for the question graphs), and image segmentation (for the image graphs) to create sparse feature rich graphs which are going to be used for downstream training.

5.1 Graph Neural Networks

Graph neural networks (GNNs) are a type of deep learning model that is specifically designed to operate on graphs. They are particularly useful when the entities represented by the nodes in the graph can be identified and labeled. A graph is a data structure that consists of a set of nodes (also called vertices) and edges that connect these nodes. Each node in a graph can represent an entity (such as a person) and each edge can represent a relationship between two entities (such as a man "riding" a horse).

GNNs are advantageous in deep learning because they can effectively capture the relationships between the nodes in a graph and use this information to make predictions or classify nodes. This is particularly useful in cases where the relationships between the nodes are complex and cannot be easily captured by traditional machine learning algorithms.

GNNs have been used to solve a wide range of tasks, including node classification, link prediction, and graph classification. They have also been applied to a variety of domains, including natural language processing, social network analysis, and biology [98]. GNNs are especially useful in situations where traditional machine learning algorithms are not sufficient for capturing the complex relationships between the nodes. PyTorch Geometric [99] is used to encode the features extracted from VisDial into feature graphs (node features and edge features).

5.2 Text Processing

Text processing is a key aspect of natural language processing (NLP) that involves converting unstructured text data into a form that can be understood and analyzed by machine learning algorithms. This subsection discusses the methods by which dialog history and questions from VisDial have been encoded into creating the different feature graphs.

5.2.1 Text Encoding

This section describes how the dialog history sentences (caption and QA pairs) and questions can be converted into vectors to be used as node values in the dialog history graph and the question graph using Global Vectors (GloVe) [11] and Long Short-Term Memory (LSTM) [12].

5.2.1.1 Global Vectors

Global Vectors (GloVe)[11] is a method for creating word embeddings, which are numerical representations of words that capture the meaning and context of the words in a continuous, low-dimensional space [15]. GloVe embeddings are created by training a model on a large dataset of co-occurring word pairs and their probabilities, and then learning a set of word vectors that best explain the observed relationships in the data.

To create GloVe vectors, the following steps are followed:

- Tokenize the text corpus into individual words and create a vocabulary of all the unique words in the corpus.
- For each unique word in the vocabulary, create a co-occurrence matrix that counts the number of times the word appears in the same context as every other word in

[15]Since the statistics are captured at a global level directly by the model, it is named as 'Global Vectors' model.

the vocabulary. The context of a word can be defined as a window of words around it, such as a window of size 2 to the left and right of the word.

- The co-occurrence matrix is then transformed into a matrix of word vectors by applying singular value decomposition (SVD). SVD is a matrix factorization technique that decomposes the co-occurrence matrix into three matrices, which can be used to reconstruct the original matrix.

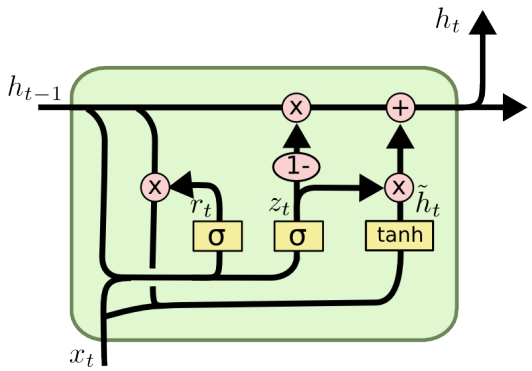
The resulting word vectors are then used to represent words in a continuous, low-dimensional space, where semantically similar words are located close to each other and dissimilar words are far apart. It should be noted that there are various variations of the GloVe algorithm, such as adding biases to the co-occurrence matrix or using different weighting schemes to down-weight less frequent word co-occurrences. For the feature representation, the Glove dictionary available from Pennington et al. (2014) [11] is directly used without modification, while the multi-modal fusion training model [10] adds an additional set of handcrafted weights to the GloVe embeddings by Pennington et al. (2014) [11].

5.2.1.2 Long Short-Term Memory

Long Short-Term Memory (LSTM) [12] is a type of recurrent neural network (RNN) that is able to learn and remember long-term dependencies between input sequences. They are able to do this through the use of a special type of neuron called a memory cell, which is able to store information over a long period of time and selectively expose it to the rest of the network when needed. LSTMs are explicitly designed to avoid the long-term dependency problems that RNNs suffer from (vanishing and exploding gradients).

Mathematically, an LSTM network can be represented as a combination of four functions: an input gate, a forget gate, an output gate, and a cell state update. These functions are used to control the flow of information in and out of the memory cell, as well as to update the cell state based on the input and previous cell state, allowing the LSTM cell to remember or forget certain information.

The input gate determines which parts of the input should be added to the cell state. It is defined as a sigmoid function applied to the dot product of the input and a weight matrix, followed by the dot product of the hidden state and another weight matrix. The forget gate determines which parts of the previous cell state should be retained. It is defined in a similar way to the input gate, but with a different set of weight matrices. The output gate determines which parts of the updated cell state should be exposed to the rest of the network. It is also defined in a similar way to the input and forget gates, but with yet another set of weight matrices. The cell state update combines the input and



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 10: LSTM Cell [100, 12]

forget gates to update the cell state based on the input and previous cell state, defined as the element-wise product of the input gate and the tanh of the dot product of the input and a weight matrix, plus the element-wise product of the forget gate and the previous cell state. Finally, the hidden state is updated based on the output gate and the updated cell state. It is defined as the element-wise product of the output gate and the tanh of the updated cell state 10.

At each time step, the LSTM cell receives an input vector and the previous hidden state. The input vector and previous hidden state are combined and passed through the input gate, forget gate, and output gate to generate the current hidden state and output. The hidden state is used to represent the memory of the LSTM cell and is passed from one-time step to the next. The output is used to generate the prediction for the current time step.

By using the gates to control the flow of information, the LSTM cell is able to remember and forget information as needed, allowing it to learn long-term dependencies between input sequences.

To encode a sentence into a single-dimensional vector using LSTM and GloVe embeddings, the following steps can be followed:

- Preprocess the sentence by tokenizing it into individual words and removing any unnecessary characters or punctuation.
- Look up the GloVe embeddings for each word in the sentence.
- Pass the GloVe embeddings for each word in the sentence through an LSTM network.
- The LSTM network can be used to process the GloVe embeddings for each word in the sentence and output a hidden state for each time step. The hidden state for the final time step is the encoded representation of the entire sentence in a single-dimensional vector.

- The encoded vector for each history item (image caption, QA1, QA2, ...) can then be used to represent a node value in the history graphs.

Similarly, to encode a word into a vector representation and generate the node values in the Question graph, we use the GloVe embeddings for each word and pass it through an LSTM network to output a hidden state for the final time step. This hidden state is the encoded representation of the word in a single-dimensional vector. It should be noted that the specific implementation details of this process, such as the size of the LSTM network and the length of the input and output sequences, vary depending on the experiments.

5.2.2 Coreference Resolution

As discussed in the earlier section 5.2.1, word embeddings have been shown to be a reliable method to encode natural language text into vector representations. However, long sentences or passages may contain subject or object pronouns which often do not translate into correct word vector representations given a lack of sufficient context.

Coreference resolution is the process of identifying and linking all mentions of the same entity in a text. It involves identifying all linguistic expressions that refer to the same real-world entity and linking them together. This is essential for text comprehension to facilitate high-level Information Retrieval tasks, especially in the domain of question-answering and text-summarization [101]. As Brack et al. (2021) [101] highlight how coreference resolution can significantly improve the quality of text comprehension on scientific papers when applied to state-of-the-art Bert-based language models.

It should be noted that the problem of coreference resolution being referred to here is a general problem of finding and resolving references in the text. However, there are other types of reference resolution tasks as well, such as anaphora resolution. Anaphora resolution involves determining the relationship between two entities in a text, where one entity refers to another entity that typically occurs *before* it [16]. The reference is called an anaphor, and the entity to which it refers is its antecedent. While both tasks involve identifying and resolving references in a text, the scope of coreference resolution is more general, as it involves identifying *all* noun phrases that refer to the same entity, while anaphora resolution specifically focuses on determining the relationship between two specific entities [102]. So although anaphora resolution is a distinct task from coreference resolution, in most cases they are equivalent, with coreference resolution having a broader scope and encompassing the majority of the use cases.

[16] Cataphora resolution is similar to anaphora resolution. The difference between them is that anaphora is the use of an expression depending on an antecedent expression, while cataphora is the use of an expression depending on a postcedent expression

This task could be interpreted under both scopes. If the dialog history is parsed turn by turn, then Anaphora Resolution becomes a suitable task for finding anaphor-antecedent pairs. However, to leverage the use of graph networks and allow the nodes to communicate with each other to improve the identification of entity-centric features [103], entity-relation identification has been focused in the scope of coreference resolution.

Among the many open-source projects on coreference resolution, the most prominent ones are NeuralCoref by HuggingFace [104] and the coreference resolution model from AllenNLP [105] [17]. For this implementation, both models are used for identifying coreference relations in the implementation.

5.2.2.1 NeuralCoref

NeuralCoref [104] works by extracting words that potentially refer to real-world entities. For each mention and mention pair, a set of features are calculated. A pairwise ranking operation is used to calculate the most likely antecedent for each mention based on these sets of features. Figure 11 shows an example of the pairwise mentions generated using the online visualization tool provided by NeuralCoref.

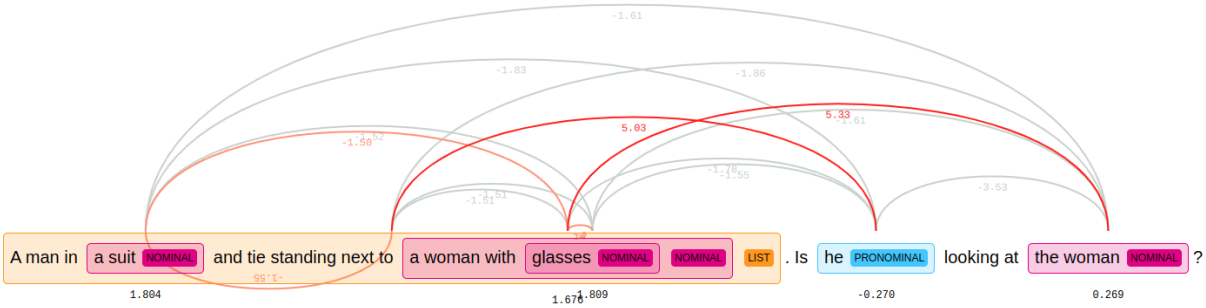


Figure 11: Pairwise mentions on a Dialog History entry from VisDial 1.0 using NeuralCoref

To generate contextualized word embeddings, NeuralCoref uses the OntoNotes Corpus [106] to capture a skeletal representation of literal meaning and coreference relations. Using these word embeddings, the context words around the mention words are extracted along with the length of the mention, speaker information, and the location of the mentions. These features are fed to two neural networks: the first neural network generates the scores for each pair of mentions and a possible antecedent, and the second scores a mention having no antecedent (since mentions sometimes represent the first reference to an entity in a text). Both these scores are compared to generate the highest score that determines

[17] The baseline implementation 2.1.2 uses the model by Lee et al. (2017) [14]. This also [14] serves as the foundation for the newer implementation by Lee et al. (2018) [105]. The newer model [105] can be found packaged by AllenNLP or at Kenton Lee's GitHub repository.

whether a mention has an antecedent and identifies this antecedent. The generated scores are based on a heuristic loss that is calculated using lack-rescaled max-margin objective [107]. NeuralCoref also fine-tunes its results based on speaker information associated with each mention. Figure 23 shows the model architecture is shown below.

The NeuralCoref pipeline offers high-speed parsing using spaCy's parser [108]. It also uses methods for predicting the embeddings of rare or unknown words based on the work of Bahdanau et al. (2017) [109].

5.2.2.2 AllenNLP coreference resolution

AllenNLP coreference resolution [105] is an improvement on the first end-to-end coreference (e2e) resolution model by Lee et al. (2017) [14] that significantly outperformed all previous work without the use of a syntactic parser or a handcrafted mentions-detector. This model directly considers all spans in a document as potential mentions, and for each of these mentions, learns to represent the possible antecedents 24. The model also computes span embeddings that combine context-dependent boundary representations and uses an attention mechanism to identify the head of the span 25. Figure 12 shows an example of the pairwise mentions generated using the online visualization tool provided by AllenNLP.

The current state-of-the-art [105] is a fully differentiable approach for resolving higher-order inferences based on the previous architecture. This approach iteratively finds the antecedent distributions from the span-ranking architecture and refines the span representations using an attention mechanism. This allows the model to make soft hops among the predicted clusters. The model is also computationally more efficient since the authors use a coarse-to-fine approach that incorporates an efficient bilinear mechanism that allows more aggressive pruning while preserving the accuracy of the model (0.2 F1-drop when compared to the non-coarse-to-fine heuristic approach).

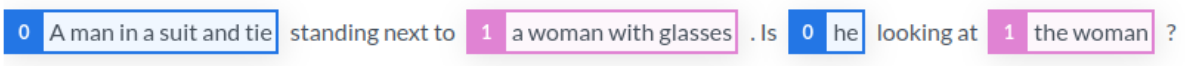


Figure 12: Pairwise mentions on a dialog history entry from VisDial 1.0 using AllenNLP coreference resolution

The model is trained with a fixed concatenation of 300-dimensional GloVe embeddings [11] and 50-dimensional Turian embeddings [110] which were then normalized to be unit vectors, and was evaluated using the OntoNotes benchmark [106].

Both the baseline implementation [1] and the novel architecture 2.1.3 use coreference

to generate pruned history graphs. An example of a generated history graph can be seen under figure 13.

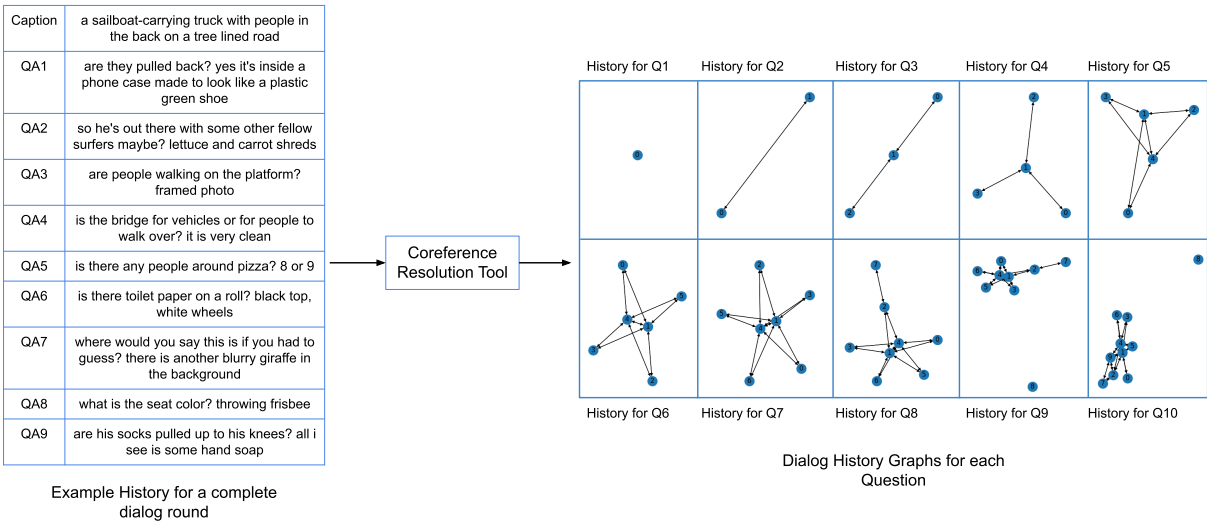


Figure 13: This figure shows how the dialog history is used with the coreference tool to generate the pruned history graphs for each round of questions.

5.2.3 Neural Parsing

In the paper "Rethinking Self-Attention: Towards Interpretability in Neural Parsing" by Mrini et al. (2019) [17], the Label Attention Layer Parser (LAL-Parser) is introduced as a neural parsing architecture that aims to improve the interpretability of self-attention mechanisms. The LAL-Parser consists of three main components: a lexical encoder, a syntactic encoder, and a self-attention layer. The lexical encoder is responsible for converting the input text into a sequence of lexical embeddings, which represent the meaning of individual words. The syntactic encoder then processes the lexical embeddings and generates syntactic embeddings, which capture the syntactic structure of the text.

The self-attention layer is the central component of the LAL-Parser 26, and it is responsible for generating a parse tree representation 14 of the input text. It does this by using self-attention to compute the importance of each word in the text relative to the other words and then constructing a parse tree based on these importance scores. One key aspect of the LAL-Parser is that it allows for the interpretation of self-attention by using a linking mechanism, which explicitly captures the relationships between words in the input text. This linking mechanism is achieved through the use of linking weights, which represent the strength of the relationships between words and can be visualized to help interpret the self-attention process 28.

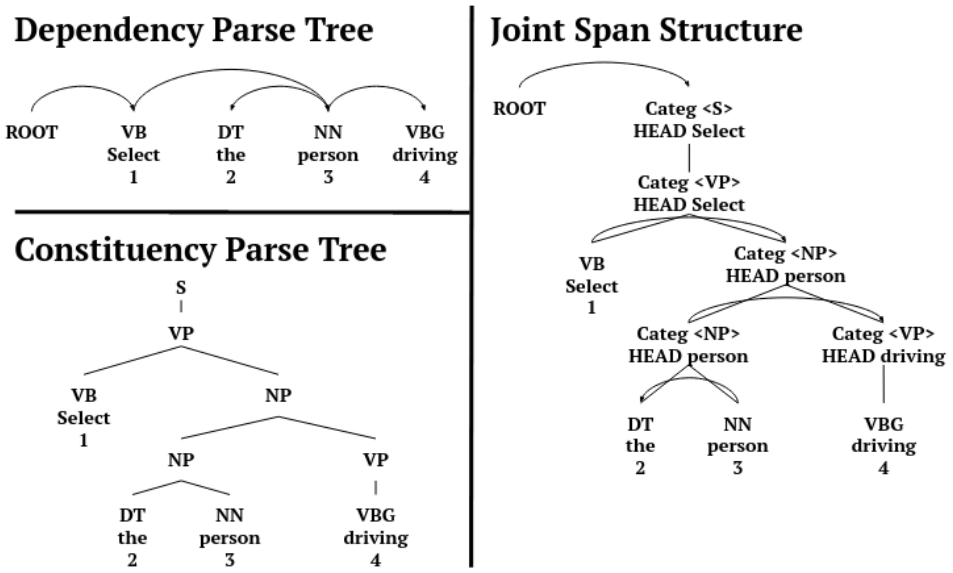


Figure 14: The parse tree representations of an example sentence "Select the person driving" has been shown in this figure [17].

The LAL-Parser is designed to perform both dependency and constituency parsing A.4. To perform dependency parsing, the LAL-Parser first converts the input text into a sequence of lexical and syntactic embeddings using the lexical and syntactic encoders. The self-attention layer then uses these embeddings to compute the importance of each word in the text relative to the other words and constructs a dependency tree based on these importance scores.

In the implementation of the baseline, [1] and the novel architecture proposed in this thesis, the dependency relations and the label attention heads are used to generate the pruned question graph. Figure 15 shows how the LAL-Parser outputs are used to generate the questions for each dialog round.

Overall, the LAL-Parser is designed to improve the interpretability of self-attention mechanisms in neural parsing by explicitly capturing the relationships between words in the input text and allowing for the visualization of these relationships. This makes it possible to improve comprehension of the dependencies between words in a sentence and to generate more accurate and interpretable dependency trees.

5.2.4 Automatic Spelling Correction

The NeuSpell model [111] is a neural spelling correction toolkit that is designed to correct spelling errors in a piece of text. The model is based on the Attention-is-all-you-need architecture [9] and uses self-attention mechanisms to correct spelling errors in the text.

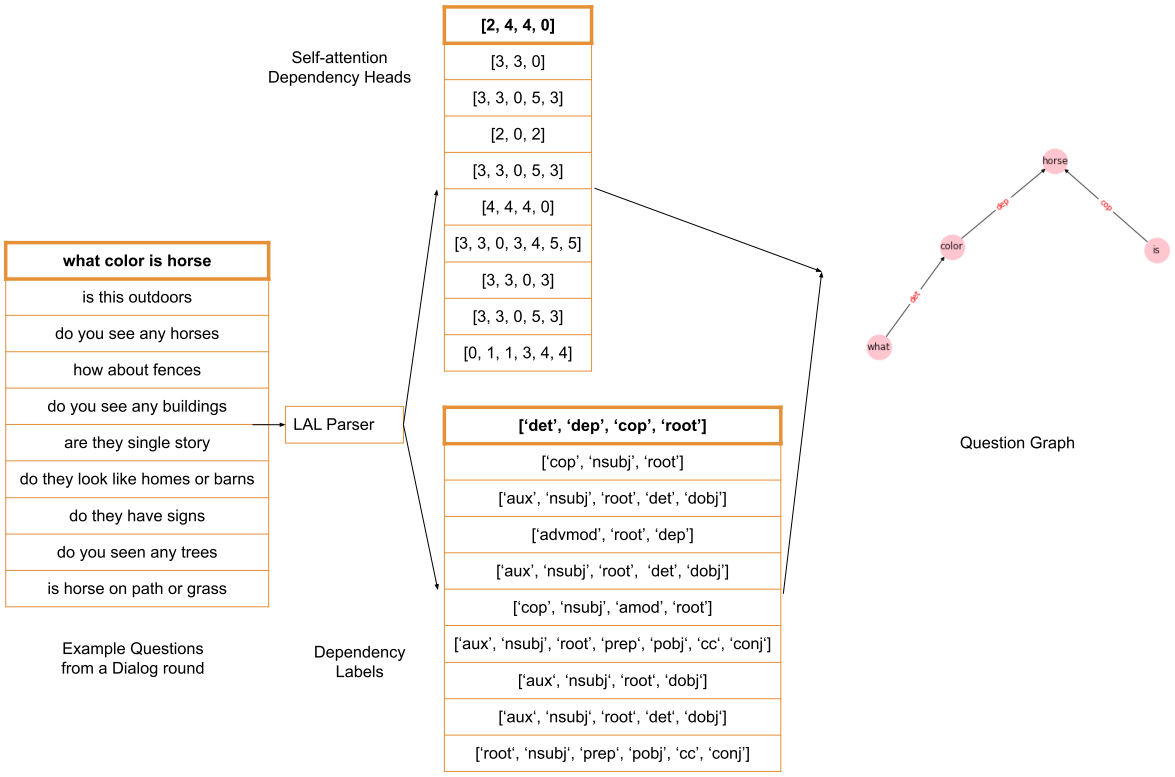


Figure 15: Using the LAL-Parser to generate the question graph. Note that the graph shown at the right is the first question graph.

The NeuSpell model works by first preprocessing the input text to identify potential spelling errors. This preprocessing step involves splitting the text into individual tokens and applying various preprocessing techniques (e.g. lower-casing and stemming) to the tokens. The model then uses Attention [9] to process sequential data and learn contextual relationships between words in a sentence. The NeuSpell model consists of several different models. The following describes its architectural pipeline:

- A preprocessing model: This model is responsible for preprocessing the input text to identify potential spelling errors. This involves splitting the text into individual tokens and applying various preprocessing techniques, such as lowercasing and stemming, to the tokens.
- A spelling correction model: This is the main model used by NeuSpell to correct spelling errors in the input text. It is based on the Transformer architecture and consists of multiple layers of self-attention mechanisms and feed-forward neural networks. The model is trained on large amounts of correctly-spelled text and uses its self-attention mechanisms to identify patterns in the data that indicate correctly-spelled words and phrases.

- A language model: This model is used to generate context-aware spelling corrections by considering the context of the input text. It is based on the Transformer architecture and consists of multiple layers of self-attention mechanisms and feed-forward neural networks. The model is trained on large amounts of text and uses its self-attention mechanisms to learn contextual relationships between words in the text.

- An error detection model: This model is used to identify potential spelling errors in the input text. It is based on a convolutional neural network (CNN) and is trained on a dataset of text with spelling errors. The model uses its CNN architecture to identify patterns in the data that indicate spelling errors.

The transformer architecture used by the NeuSpell model consists of multiple layers of self-attention mechanisms and feed-forward neural networks. The self-attention mechanism attends to different parts of the input text to identify the relationships between the tokens, while the feed-forward neural network learns the more complex patterns in the data. Once the potential spelling errors have been identified, the model corrects spelling errors and generates the corrected output sentence. The NeuSpell Model has been shown to outperform other spelling correction models on a variety of benchmarks.

This implementation uses the BertChecker model from NeuSpell (based on the BERT model [56]) that generates a fixed-length representation for each word based on its context within the sentence. Next, the BertChecker model applies the spelling correction algorithm by NeuSpell to the encoded representations of the words. Finally, the model decodes the corrected representations of the words back into the text and generates the corrected sentence. Since the VisDial dataset contains chat-based dialogs, the sentences generated often contain spelling errors. Since Glove Embeddings [11] is used in the baseline and the novel implementation 2.1, it is hypothesized that spelling correction can help in the process [18]. An example translation can be seen in figure 2.

5.3 Image Processing

Image segmentation is a task in computer vision that involves identifying and segmenting individual objects within an image or video. It is performed by dividing an image into mul-

[18]GloVe embeddings are pre-trained vector representations of words, trained using a matrix factorization technique that aims to preserve the co-occurrence statistics of the words in the training data. On the other hand, BERT embeddings are also pre-trained vector representations of words, but they are generated using a transformer-based neural network architecture.

Input	Output
I lik to play footbal	I **like** to play **football**
I will eat fish for dinner and drank milk	I will eat fish for dinner and **drink** milk

Table 2: While NeuSpell is designed to be a spelling corrector, it also corrects some grammar mistakes as can be seen in the second example sentence (past tense to future tense correction).

tiple segments or regions, each of which corresponds to a different object or background. There are three main types of image segmentation [112]:

- **Semantic segmentation** involves assigning a class label to each pixel in the image, based on the object or background it belongs to. For example, an image of a person standing in front of a building might be segmented into two classes: person and background (building).
- **Instance segmentation** goes one step further than semantic segmentation by not only labeling each pixel with a class but also assigning a unique identifier to each instance of that class. For example, in an image with multiple people, each person would be labeled as a person class, but would also be given a unique identifier to distinguish them from other people in the image.
- **Panoptic segmentation** combines both semantic and instance segmentation and aims to provide a complete segmentation of the image by labeling both the class and the instance of each object in the image. In addition to objects and background, panoptic segmentation also includes "stuff" classes, which correspond to regions of the image that do not belong to any object (e.g. sky, grass, etc.).

There are various algorithms and techniques that can be used for image segmentation like clustering algorithms, graph-based methods, and deep learning approaches. One popular deep learning framework for image segmentation is called Detectron2 [113], a software package developed by Facebook AI that provides a suite of tools for training and evaluating object detection and image segmentation models. Instance segmentation can be performed using the following steps:

- Loading and preprocessing the input image: This includes resizing the image to a suitable size, applying any necessary transformations (e.g. cropping, flipping), and converting it to a format that can be processed by the model.

- Using a pre-trained object detection model A.6 to generate object proposals: This involves running the input image through a model that has been trained to identify and localize objects within an image. The output of this step is a set of bounding boxes that enclose the objects of interest.
- Using a pre-trained semantic segmentation model to predict class labels for each pixel within the bounding boxes: This involves running the input image (cropped to the bounding boxes) through a model that has been trained to classify each pixel within an image into one of a set of predefined classes (e.g. "person", "car", "tree" as mentioned earlier). The output of this step is a set of class labels for each pixel within the bounding boxes.
- Combining the different class labels with the bounding boxes to generate instance masks: This involves using the class labels to create a mask for each object within the bounding boxes, with each object being represented by a different color or intensity value. The output of this step is a set of instance masks that can be overlaid on top of the input image to visualize the segmentation results.

The image segments are encoded into a 2048-dimensional feature vector using Faster R-CNN by:

- Preprocessing the image by resizing it to a fixed size and normalizing it (including converting all images into RGB).
- Passing the preprocessed image through the Faster R-CNN model. This model consists of a convolutional neural network (CNN) that processes the image and extracts features from it, as well as a region proposal network (RPN) that proposes regions of the image that may contain objects.
- For each region proposed by the RPN, the CNN processes the region and extracts a feature vector from it.
- The feature vectors extracted by the CNN are passed through a fully connected (FC) layer, which maps the vectors to a fixed-length 2048-dimensional space.

The output of the FC layer is the 2048-dimensional feature vector for each image segment. This constitutes the node embeddings for the image graph.

5.3.0.1 Instance Segmentation

Based on the implementation of Yao et al. (2018) [18], relations between instances in objects are denoted by three types:

- Spatial relations: These refer to the positional relationships between objects in an image, such as "overlap", "next to", "above", or "below".
- Attribute relations: These refer to the properties or attributes of objects, such as "red", "large", or "shiny".
- Functional relations: These refer to the actions or functions performed by objects, such as "driving" or "drinking".

The authors [18] use this to model relations into an image captioning system, in order to improve the descriptive and explanatory power of the generated captions. However, for the baseline [1] and novel implementation, only the spatial relations are considered to keep edges between the graph nodes for generating the pruned image graph. For example, if two object regions are located at a significant distance from one another, it is likely that there is no relation between them. The authors [1] also use this to prune irrelevant nodes to obtain a sparse pruned graph 16.

(a) Original image from COCO (left) and its instance segments (using Faster RCNN [13] (right)

(b) Image graph after pruning based on the spatial relationship modeling by Yao et al. (2018)[18]

Figure 16: Example of how images are converted into feature graphs

5.3.0.2 Panoptic Segmentation

As mentioned earlier, panoptic segmentation can also be used to generate feature graphs. A benefit of panoptic segmentation over instance segmentation is that it provides a much more complete representation of all regions in the image. Since the "stuff" part of the image is overlooked in instance segmentation, the hypothesis is that panoptic segmentation will yield better performance for questions that address the "stuff" in the images. Panoptic segmentation also results in much more densely connected graph structures. This comes with a few advantages [16, 114], namely:

- Densely connected graphs provide a more complete representation of the relationships between nodes in the graph. In sparse graphs, only a small number of edges are present, which can lead to an incomplete representation of the relationships between nodes. Densely connected graphs, on the other hand, contain many edges, which can provide a more comprehensive representation of the relationships between nodes.
- They can provide a more efficient representation of the graph. Densely connected graphs contain many edges, which can be encoded using a smaller number of parameters compared to sparse graphs. This can make training GNNs on densely connected graphs more efficient, as there are fewer parameters to optimize.
- GNNs are able to learn and model the relationships between nodes in the graph, and densely connected graphs provide more information about these relationships, which can lead to improved performance on tasks such as link prediction or graph classification.

Figure 17 highlights the process of how panoptic segments can be converted into densely connected graphs. Two regions encoded in graph feature nodes share an edge between them if their panoptic segmentation masks share an edge.

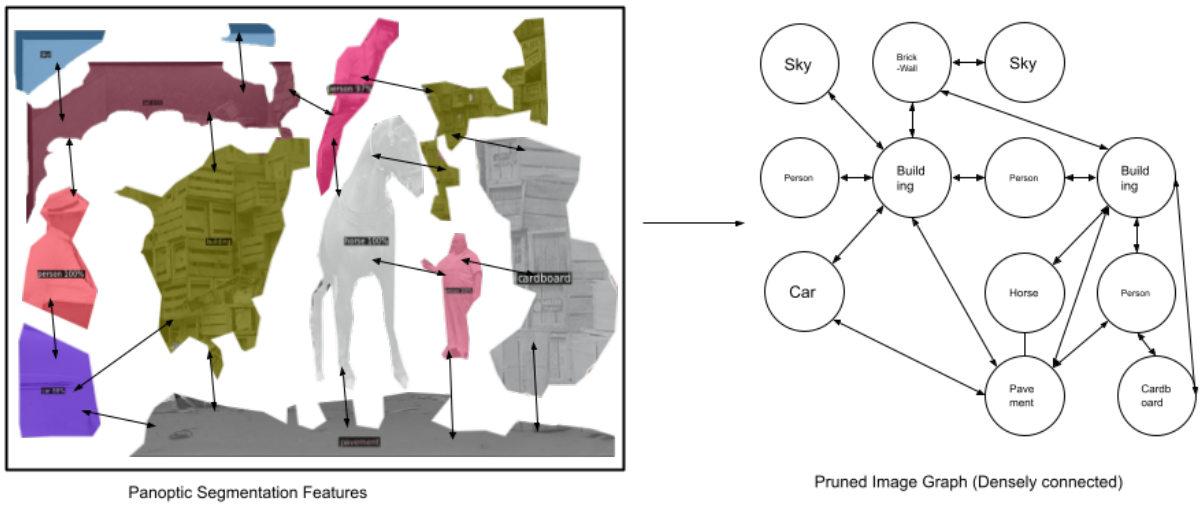


Figure 17: Example of how panoptic segments are converted into feature graphs

5.4 Graph Attention

Graph Attention Network (GAT) [15] is a type of graph neural network (GNN) that uses attention mechanisms [9] to weigh and update the importance of different nodes and edges in a graph while learning to make predictions or perform tasks. The attention

mechanisms in GATs are based on self-attention mechanisms, which allow the model to attend to different parts of the input sequence (or graph in this case) at different times.

GATs operate by first encoding the nodes and edges of the graph using a set of embedding vectors. These embedding vectors are then used to compute the attention weights for each node and edge in the graph. The attention weights are calculated using a dot product between the embedding vectors of the nodes and a set of trainable weight vectors. The attention weights are then used to calculate the new node embeddings, which are used to make predictions or perform tasks. The node embeddings are calculated using a weighted sum of the embedding vectors of the neighboring nodes, where the weights are the attention weights.

GATs are able to learn complex relationships between nodes in the graph by using multiple attention heads, which allow the model to attend to different aspects of the graph simultaneously. The output of the multiple attention heads is then concatenated and passed through a final linear layer to make the final prediction or perform the task. GATs have been shown to be effective in a variety of tasks on graphs, including node classification, graph classification, and link prediction.

The authors [1] use GATs to learn the attention weights between the different dialog turns in the history for each of the three graphs (dialog history graph, question graph, and image graph), which allows the GoG model to capture the sequential dependencies between the different turns and better understand the overall context of the dialog. This helps the model to generate more coherent and accurate responses to the questions in the visual dialog task.

5.5 Global Attention Pooling

In the paper "Gated Graph Sequence Neural Networks" by Li et al. (2015) [16], the authors propose a method for converting a graph into an embedding using a process called global attention pooling. This process involves calculating a weighted sum of the node embeddings in the graph, where the weights are determined by the attention mechanism.

To calculate the weighted sum, the attention mechanism [9] first computes a set of attention weights for each node in the graph [19]. These attention weights represent the importance of each node in the graph with respect to the task at hand. The attention weights are then used to weight the node embeddings, and the weighted node embeddings are summed to produce the final graph embedding.

[19]The attention weights are calculated using a weighted sum of the node embeddings, where the weights are determined by a set of learnable parameters. These parameters are learned during the training process, allowing the model to learn which nodes are most important for the task at hand.

The final graph embedding can then be fed into a downstream model, such as a classifier or a decoder, to perform the desired task. The global attention pooling process allows the model to selectively focus on the most important nodes in the graph and produce a more informative and relevant graph embedding.

The authors [1] use this graph embedding to inject context information from the dialog history graph into the question graph and from the question graph into the image graph. Finally, they [1] convert the context-rich graphs into their pooled embedding [16] to be used in the multi-modal fusion training step [10].

5.6 Multi-modal Fusion

The Light-weight Transformer for Many Inputs (LTMI) architecture is a variant of the Transformer architecture that was proposed by Nguyen et al. in the paper "Efficient Attention Mechanism for Visual Dialog that can Handle All the Interactions between Multiple Inputs" (2020) [10]. The LTMI architecture is designed to efficiently handle interactions between multiple inputs in a visual dialog system.

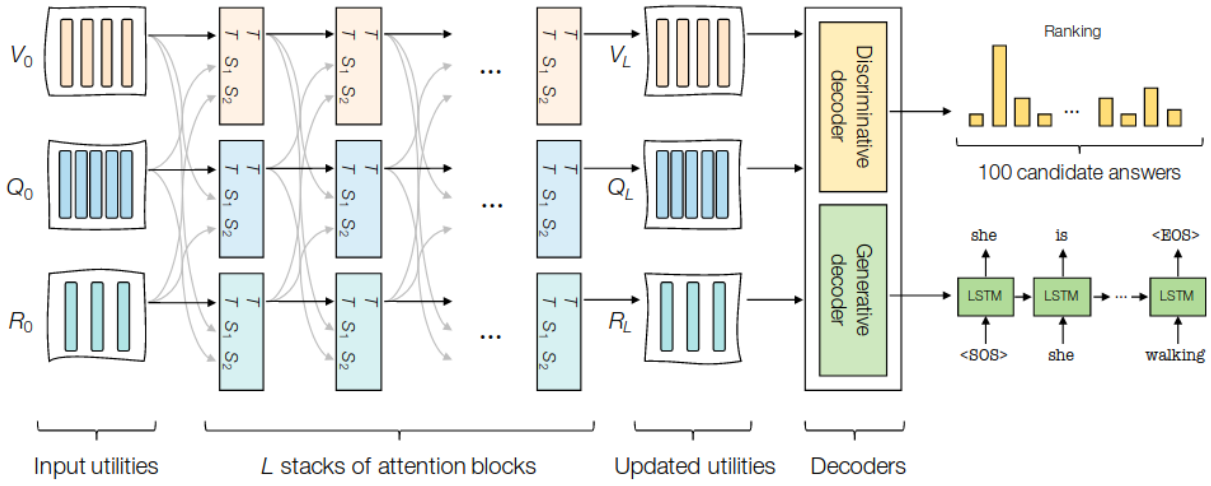


Figure 18: The complete LTMI architecture for Visual Dialog [10]. V refers to the image encoding consisting of the combined image-region level features, Q refers to the question utility features, and R refers to the different dialog round encodings where T is any given time step.

In the LTMI architecture, the image, question, and dialog history are first encoded into a set of feature vectors using separate encoder networks. These feature vectors are then stored in the memory module, which is implemented using a multi-headed self-attention mechanism. The model then uses an attention mechanism to select relevant information from the memory module and combine it with the current input (the question in the case

of visual dialog) to generate a response. The attention mechanism uses a set of weights to determine the importance of each piece of information in the memory module and combines the weighted information using a weighted sum to produce the final output. Figure 18 shows an overview of the architecture of the model and how it performs the encoding and decoding steps.

This allows the model to handle the complex interactions between multiple inputs in visual dialog by using a memory module and attention mechanism to store and retrieve relevant information from the input sequences. This makes it an ideal candidate to use with the three feature graphs.

6 Experiments

6.1 Experimental Setup

Dataset and Implementation Details To verify the approach, experiments are conducted on 1% percent of the subset of VisDial Train 1.0 [2]. The subset contains 1233 dialogs with 1233 unique images, 8402 unique questions, and 268k unique answers 1.

To represent the image regions, Mask-RCNN is used as the model A.6 which is an extension of the Faster-RCNN by Ren et al. (2015) [13] with ResNet-50 [115] [20]. A 2048-dim feature vector representing each detected image segment was generated from the last layer of the ResNet-50 model. Following Nguyen et al. (2020)'s [10] baseline implementation, 36 object proposals were detected for each image. For the question and history features, the GloVe 6B-300dim [11] vectors were used. Using an LSTM [12], the history sentences were converted into a 300-dim vector 5.2.1.2. Similarly, each word in the question was converted into a 300-dim vector 5.2.1.2. To generate the history and question graph, the number of nodes is limited to 20 nodes per graph. The universal dependency relations generated by the neural parser [17] were each assigned a unique value to initialize the edge attributes in the question graphs. NeuralCoref [104] was used to resolve coreference relations in the history graphs for the models whose performance has been noted under 3. To obtain the updated graph with multi-headed attention 5.4, 4 attention heads are used. The dimensionality of the hidden features is set to 512.

For the baseline implementation 2.1.2, the 512-dim hidden vector generated from the history graph for each round of QA using 5.4 and 5.5 and is concatenated to each node in the question graph. This generates a 300-dim (question node dimension) + 512-dim (history graph embedding) vector for each question graph node. Multi-headed attention 5.4 with 4 heads are used to update the history-aware question graph. Global Attention is used on the relation-aware graph to generate a 512-dim vector. This history-aware question graph embedding for each round of QA is similarly concatenated with each node of the image graph. This generates a 2048-dim (image node dimension) + 512-dim (history-aware question graph embedding) vector for each image graph node. Multi-headed attention is used again 5.4 with 4 heads to update the history-aware-question-aware image graph. Global Attention is used on this relation-aware image graph to generate a 512-dim vector. All three 512-dim hidden vectors (history graph, history-aware question graph, history-aware-question-aware image graph) are used in the LTMI model [10].

[20] While the project repository contains the implementation for ResNet-101 [115], experiments using this model have not been carried out yet and is proposed for Future Work 8

For the novel implementation 2.1.3, the 512-dim hidden vector generated from the history graph for each round of QA using 5.4 and 5.5 and is connected to each node in the question graph. This makes each history-aware question graph have 20+1 nodes. To match the dimensionality of the history graph embedding with the question graph nodes, the 512-dim vector is transformed into a 300-dim vector through a linear layer transforming it into a 300-dim vector. Similarly, during the creation of the history-aware-question-aware image graph with 36+1 nodes, to match the dimensionality of the image graph nodes, the 512-dim hidden vector is transformed into a 2048-dim vector using a linear layer. The final hidden output for each graph (using 5.5) is set to 512.

The model is implemented using PyTorch [116], PyTorch Geometric [99], and the Deep Graph Library [117]. In the experiments, following the implementation of Chen et al. (2021) [1] and Nguyen et al. (2020)[10], the Adam Optimizer [118] is used. For the choice of the learning rate, the warm-up strategy by Goyal et al. (2017) [119] is used [21]. The mini-batch size is kept at 8 for the training set [22]. The model is trained using an Nvidia GTX-1080 which takes about 1 hour to train on the 1% subset.

Automatic Evaluation For evaluation, three metrics have been considered. The R@K metric (R@1, R@, and R@10) and the Mean Reciprocal Rank (MRR). The scoring systems are adopted following Das et al. (2016) [2] and Chen. et al. (2021) [1].

- **Recall@k (R@k)** is calculated as the percentage of queries for which the correct result is ranked within the top K results returned by the system. For instance, R@1 is the percentage of queries for which the correct result is ranked first, and R@5 is the percentage of queries for which the correct result is ranked within the top 5 results. A higher value for R@K indicates that the system is more effective at returning relevant results for a larger percentage of queries.

  For example, if R@10 has a score of 80%, this means that it returns the correct result within the top 10 results for 80% of the queries it receives. This is considered as good performance. For the experiments, R@1, R@5, and R@10 are used.

- **Mean Reciprocal Rank (MRR)** is calculated as the mean of the reciprocal ranks of the correct results for a set of queries. The reciprocal rank of a result is defined as 1/rank, where rank is the position of the correct result in the list of results returned

[21] The learning rate is initially kept at 0.0001, which is linearly increased at each epoch till it reaches 0.0002 at epoch 4. After 10 epochs, the learning rate is decreased by 1/4 for every 2 epochs up to 15 epochs.

[22] Unfortunately, due to memory limitations, the model was not able to execute the validation step even with batch size 1.

by the system. For example, if the correct result is ranked first, the reciprocal rank is $1/1 = 1.0$. If the correct result is ranked second, the reciprocal rank is $1/2 = 0.5$.

To calculate MRR, the reciprocal ranks for each query are first computed and then the mean of these values is taken. A higher MRR value indicates that the system is more effective at returning the correct result as the top-ranked result for a larger percentage of queries. For example, if MRR has a score of 0.8, this means that the correct result is ranked first for 80% of the queries it receives.

- **Mean rank** is calculated as the mean of the ranks of the correct results for a set of queries. The rank of a result is the position of the result in the list of results returned by the system. For example, if the correct result is ranked first, the rank is 1. If the correct result is ranked second, the rank is 2.

  To calculate mean rank, the ranks for each query are first computed and then the mean of these values is taken. A lower mean rank value indicates that the system is more effective at returning the correct result closer to the top of the list of results for a larger percentage of queries. For example, if the final mean rank is 2, this means that the correct result is ranked within the top 2 results for the majority of queries it receives.

Discriminative Setting Das et al. (2016) [2] describe a discriminative setting for their visual dialog model. Under this setting, the authors use a discriminative model to predict the next dialog turn given the previous dialog history and the current image. This means that the model is trained to predict the most likely next dialog turn based on the input data, rather than generating a dialog turn from scratch.

The authors compare the performance of this discriminative model to a generative model, which is trained to generate dialog turns from scratch based on the input data. They find that the discriminative model outperforms the generative model in terms of both perplexity (a measure of the model's uncertainty) and human evaluation. Overall, the discriminative setting is one approach to modeling dialog in the visual dialog task, and the authors' results suggest that it may be a more effective approach than using a generative model.

6.2 Results

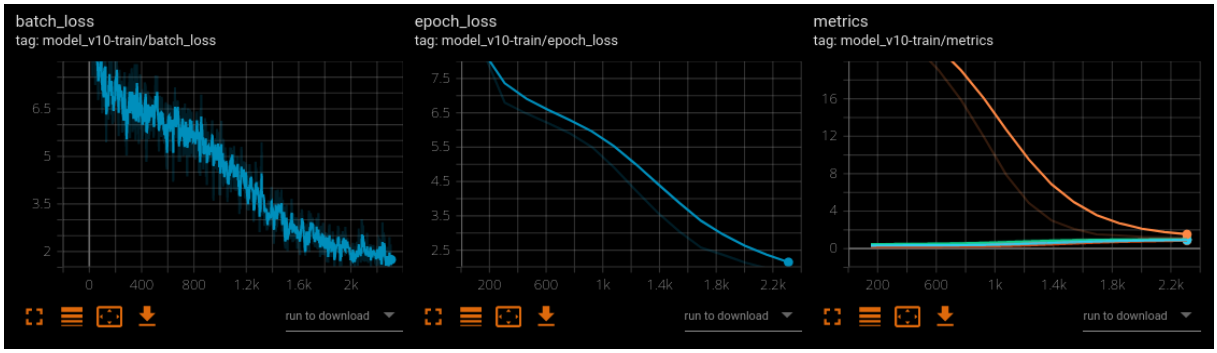
Table 3 show a comparison between the results compiled by Chen et al. (2021) [1] along with the results from the experiments carried out under the scope of this thesis. The experiments have been carried out in 4 different settings: (1) GoG-LTMI (Thesis) [1, 10]

is the baseline implementation for the thesis, (2) Master Node-LTMI (Thesis) [10] is the novel implementation proposed under the thesis, (3) Ablation-LTMI (Thesis) [10] is the control experiment where no relation-awareness feature has been transmitted across the feature graphs (i.e. no communication between the dialog history graph and the question graph and the image graph), and (4) Master Node-LTMI using Neuspell (Thesis) [10, 111] where questions and dialog history has been used spell-checked with the help of NeuSpell [111]. The results shown are performed under the discriminative setting 6.1. The results are the average scores after training the model thrice from scratch and measuring their results. The results have been analyzed in the following section 7.

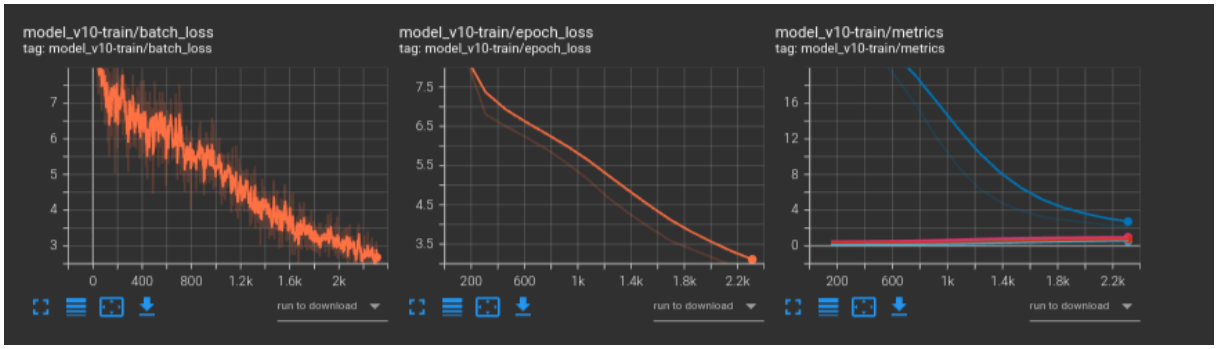
	Model	**MRR**	**R@1**	**R@5**	**R@10**	**Mean**
Attention-based Models	ReDAN [71]	64.75	51.10	81.73	90.90	3.89
	MCA [75]	37.68	20.67	56.67	72.12	8.89
Pretraining-based Models	VisualBERT [4]	50.74	37.95	64.13	80.00	6.28
	VDBERT [8]	51.17	38.90	62.82	77.98	6.69
Graph-based models	GNN-EM [76]	61.37	47.33	77.98	87.83	4.57
	DualVD [78]	63.23	49.25	80.23	89.70	4.11
	FGA [77]	66.20	52.75	82.92	91.07	3.80
	CAG [74]	63.49	49.85	80.63	90.15	4.11
	KBGN [79]	64.13	50.47	80.70	90.16	4.08
	LTMI [10]	61.20	47.08	77.78	87.60	4.88
	LTMI-GoG [1, 10]	63.13	49.88	79.65	89.05	4.39
	LTMI-GoG-Multi [1, 10]	63.52	50.01	80.13	89.28	4.31
	GoG-LTMI (Thesis) [1, 10]	**76.77**	**63.07**	**94.73**	**97.93**	**2.18**
	Master Node-LTMI (Thesis) [10]	**93.95**	**89.25**	**99.54**	**99.92**	**1.17**
	Ablation-LTMI (Thesis) [10]	**88.58**	**80.91**	**98.14**	**99.42**	**1.44**
	Master-Node-LTMI-NeuSpell (Thesis) [10, 111] [23]	**51.71**	**39.39**	**71.41**	**80.00**	**4.56**

Table 3: Main comparisons on both VisDial v1.0 datasets using the discriminative decoder including the results from Chen et al. (2021) [1]. The section in bold denotes the model performances from the thesis. It should be noted that the results are on a very small subset of the VisDial 1.0 dataset where the scores are from the training, not validation [24]. The arrows next to the score metrics indicate if higher or lower is better.

Figures 19 show the batch loss, epoch loss, and the metrics (Mean, MRR, R@1, R@5, and R@10) generated while training the GoG-LTMI (Thesis) [1, 10] and Master Node-LTMI (Thesis) [10] implementations. While figure 20 shows the learning rate for each step of these two models under the experimental setup hyperparameter settings 6.1.

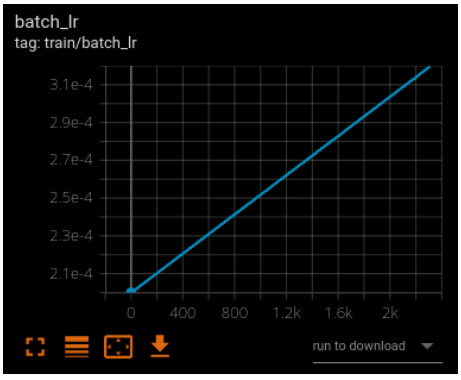


(a) Training plots for the baseline (GoG) implementation

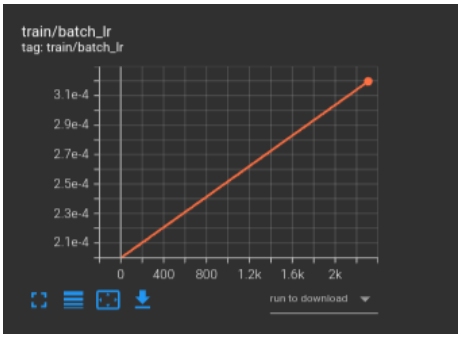


(b) Training plots for the master node implementation

Figure 19: The figure shows the batch loss, epoch loss, and metrics (Mean, MRR, R@1, R@5, and R@10) for the baseline and master node implementations



(a) Learning rate for the baseline (GoG) Implementation



(b) Learning rate for the master node implementation

Figure 20: Plots of the learning rates for the two implementations

7 Discussion

Model Results The three models GoG-LTMI (Thesis), Master Node-LTMI (Thesis), and Ablation-LTMI (Thesis) outperform all previously mentioned models along the metrics described under section 6.1. However, since these scores are the metrics on the training data, it is difficult to foresee how much the models have actually learned without their performance on the validation set. This can also be concluded while comparing the results of the GoG-LTMI [1] implementation on the complete dataset vs the results on the smaller subset (GoG-LTMI (Thesis)) as the subset model outperforms the work by Chen et al. (2021) [1] by a significant margin.

Even though the variation in the subset of the training data used includes a large variation 1, the dataset still is too small in comparison to the VisDial 1.0 training data. Chen et al. (2021) [1] concludes from their Ablation Study that while all three graphs are crucial for Visual Dialog, the image graph is the most important. From the smaller subset used, the set of images used is the least (compared to unique questions and dialogs, and answers). This is also a strong indicator that the trend of model performances cannot be extended without experimenting on the complete or at least a significantly larger subset. Similarly, while the GoG-LTMI (Thesis) model and Ablation-LTMI (Thesis) both outperform Master Node-LTMI (Thesis) under the scope of the smaller dataset, further experimentation is required with different hyperparameter settings to verify the novel architecture's performance against the baseline.

Furthermore, while the GoG-LTMI (Thesis) outperforming the Ablation-LTMI (Thesis) indicates the effect of the relation-awareness architecture proposed by Chen et al. (2021) [1], it still needs to be verified on the complete dataset to confirm whether the approach holds under the slightly different hyperparameter settings used 6.1.

However, the model scores are still a good starting point and provide a strong foundation for further experimentation. By continuing to train and fine-tune the model with more data and under different settings, there is an opportunity to improve the performance of each of the architectures used and gather more accurate and reliable results.

Training Plots The batch loss is the loss or error of the model on a batch of training data, and the epoch loss is the loss or error of the model on the entire training dataset after a single epoch of training. Loss is a general measure of how well a model is able to make predictions on the training data, with a lower loss indicating better performance in both cases.

The plots indicate that the two models GoG-LTMI (Thesis) and Master Node-LTMI

(Thesis) are learning, however, it is difficult to conclude this with certainty without the validation loss as the models could be overfitting.

Coreference Resolution The architecture also uses NeuralCoref [104] and the AllenNLP coreference model [105] to perform coreference resolution. While experiments haven't been conducted in full to test the reliability and performance of these models to perform coreference resolution on the final training, their advantages and disadvantages should still be noted [25].

NeuralCoref [104], which is packaged with spaCy, is easy to use and faster at resolving coreference relations, but the scores it generates are not normalized and it often fails to locate clusters. On the other hand, AllenNLP [105] has obscured similarity details for its generated coreference clusters and tends to detect very long clusters, making it more reliable for large spans but struggling with commonsense reasoning problems and contractions. While Huggingface tends to locate fewer clusters and substitute mentions less often, AllenNLP seems to replace mention pairs more aggressively due to its tendency to find more clusters [14].

Image Features Spatial relationship modeling based on the implementation by Yao et al. (2018) [18] is used to carry out image graph pruning. However, for this implementation, the scores used are primarily overlapping scores between the different instances that are detected by the Mask R-CNN model. However, other spatial relationship features as suggested by Yao et al. (2018) [18] can be used to update the edge attributes. It should be noted that this can result in more fully-connected graphs which might lead to noise in the model as noted in the Ablation Study by Chen et al. (2021) [1].

[25]The conclusions are observed while creating the history graph and manually testing whether the function worked reliably. This isn't an absolute metric, however, serves as a beneficial guide.

8 Future Work

This section gives an overview of what could follow the work done under the scope of the thesis.

Dataset As noted under 7, the performances of the models used indicate a good starting point and provide a strong foundation for further experimentation. Replicating experiments on a larger subset of the VisDial data and scaling it up to the complete dataset is certainly the first step to take. While hardware limitations might remain an issue, workarounds can be made with smaller batch sizes and more efficient GPU memory management.

Hyperparameter tuning The complete architecture of the baseline implementation 2.1.2 and the novel implementation proposed under this thesis 2.1.3 contain a number of hyperparameters that can be adjusted to fine-tune the models generated.

Chen et al. (2021) [1] in their experiments use 100 image proposals, while the baseline implementation by the LTMI model [10] use 36 image proposals. Both settings have been shown to generate state-of-the-art comparable scores on the VisDial Challenge. Using panoptic segmentation instead of instance segmentation can also help in capturing all parts of the image (stuff and things) instead of only the "thing" part of the image. The choice of coreference tools and identifying the validity of the clusters detected also remains an important task that needs further investigation.

Furthermore, it is necessary to identify the scalability of the models fine-tuned on the VisDial dataset to perform the general task of visual dialog as pointed out earlier in 4.5 and 4.4. As 4.5 mentions, further dataset analysis needs to be carried out in order to understand the performance of different methods of training as well as pre-trained models that ground natural language into images, the naturalness of the questions generated [120].

Finally, additional data augmentation methods can be implemented to improve the quality of the captions for VisDial as shown by Fang et al. (2014) [20]. Similarly, Jiang et al. (2019) [121] also show how caption quality can be automatically evaluated while choosing among COCO captions [81].

Optional Architecture An optional research goal can also be carried out as discussed under the proposal of the thesis where the goal is to investigate the performance of a

different setting where the image and the History dialog states graphs are embedded into a master node each and then fed into the question graph.

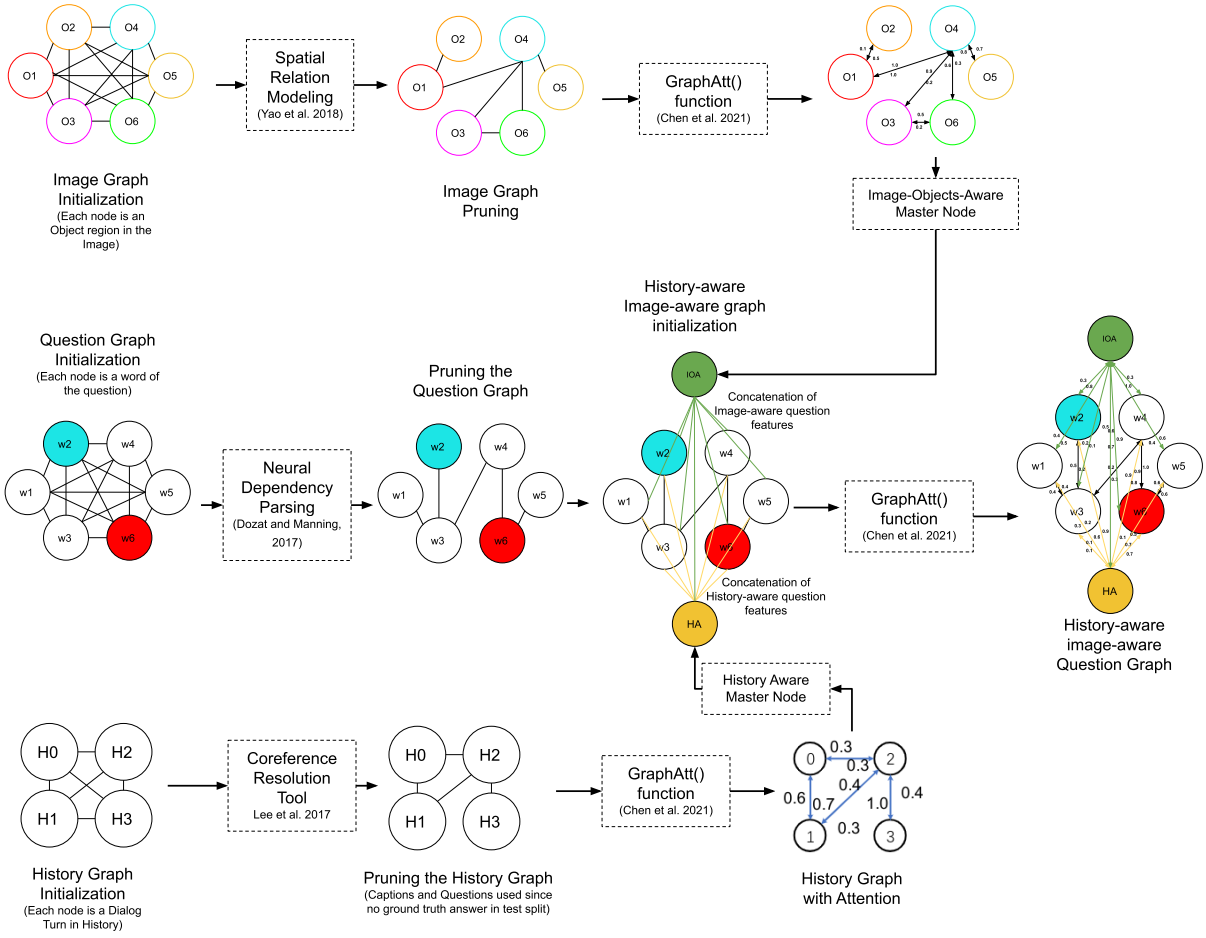


Figure 21: Optional Implementation from Proposal: Injecting both dialog history and image graph features as a master node in the question graph to form a fully question-aware representation

9 Conclusion

Researchers have been trying to enable machines to have conversations with humans about visual data and understand the context and meaning of the conversation. Natural language is complex and ambiguous, and it can be difficult for machines to accurately parse and understand the nuances of human communication. Additionally, visual data can be complex and high-dimensional, and it can be difficult for machines to accurately interpret and understand the relationships and contexts depicted in the data. To this end, the Visual Dialog Challenge [2] dataset has been developed to allow the training and fine-tuning of multi-modal language and vision models.

The task of Visual Dialog is an extension of a number of problems like image captioning, video description, and visual question answering, but it involves more than one natural language interaction step and includes a discourse component. In contrast to VQA, which tends to focus on imaginative questions and closed-ended answers, VisDial questions are longer and more descriptive, and the answers are open-ended.

Previous research has shown the benefits of using pre-trained language and vision models which are trained on related vision-language datasets and then fine-tuned on the visual dialog dataset. However, these approaches suffer from a few limitations. Since these models are typically trained on large datasets, they may not be directly relevant to the visual dialog task. This can lead to a lack of fine-grained understanding of the specific details and context of the visual dialog task, which can negatively impact their performance. Additionally, pre-trained models may be less flexible and may not be able to effectively adapt to different domains or tasks without additional fine-tuning. Finally, pre-trained models may be more difficult to interpret and explain, which can be a challenge when trying to understand how they are making decisions or generating responses in the visual dialog task. Attention-based models have also found success in solving the challenge. However, they tend to suffer from computational complexity issues, a lack of interpretability, and context limitation problems.

Another approach undertaken to solve this task makes the use of graph structures. As highlighted by Chen et al. (2021) [1], graph neural networks have been used to model the different implicit relations between objects in an image and dialog. However, these models fail to model the different coreference relationships between the dialog history and the dependency relations between words in a question. Furthermore, dialog information is not communicated effectively to extract meaningful context from images. To address this problem, the authors [1] use three sequential graphs to form a fully-question-aware image graph representation. Their approach has been replicated under the scope of this thesis and

the experimental results indicate that this relation-awareness mechanism shows promise in building feature-rich graphs that are able to outperform strong baseline implementations using solely attention-based models.

However, as pointed out by Conte et al. (2013) [122], explicit relation modeling might not always yield the best results. To make better use of graph attention networks [15], this thesis highlights a new approach to perform an implicit relation method to inject semantic information from one graph into another using a master node that captures all information of a graph. Initial experimental results indicate this to be a good starting point, however, it also advocates the need for further experimentation with alternate settings to confirm the validity of this new architecture.

A Appendix

A.1 Syntax Heuristics

Heuristic	Definition	Example
Lexical Overlap	Assume that a premise entails all hypothe ses constructed from words in the premise	textbfThe doctor was **paid** by **the actor**. [Wrong] The doctor paid the actor.
Subsequence	Assume that a premise entails all of its contiguous subsequences.	The doctor near **the actor danced**. [Wrong] The actor danced.
Constituent	Assume that a premise entails all complete subtrees in its parse tree.	If **the artist slept**, the actor ran. [Wrong] The artist slept.

Table 4: The heuristics that were targeted by the HANS dataset with the examples showing the incorrect entailment predictions that these heuristics would lead to [55]

A.2 Reasoning over Knowledge Graphs in Vector Space

A.3 Coreference Resolution

A.4 Dependency and Constituency Parsing

Dependency parsing is a type of syntactic parsing that aims to identify the grammatical relationships between words in a sentence and represent them in a directed acyclic graph (DAG) called a dependency tree. Dependency parsing is used to analyze the grammatical structure of a sentence and to identify the dependencies between the words in the sentence. In a dependency tree, each word in the sentence is represented as a node, and the dependencies between the words are represented as directed edges connecting the nodes. The head of the dependency is the word that is being modified by the dependent word, and the dependent is the word that is modifying the head. For example, in the sentence "The cat sat on the mat," the word "cat" is the head of the dependency with "sat" as the dependent, and "mat" is the head of the dependency with "on" as the dependent.

Constituency parsing is a type of syntactic parsing that aims to identify the grammatical structure of a sentence and represent it in a tree-like structure called a parse tree. Constituency parsing is used to analyze the grammatical structure of a sentence and to identify the hierarchical relationships between the words in the sentence. In a parse tree, each word in the sentence is represented as a node, and the relationships between the words are represented as edges connecting the nodes. The root of the parse tree represents the main clause of the sentence, and the branches of the tree represent the subordinate clauses and phrases that make up the sentence. For example, in the sentence "The cat sat

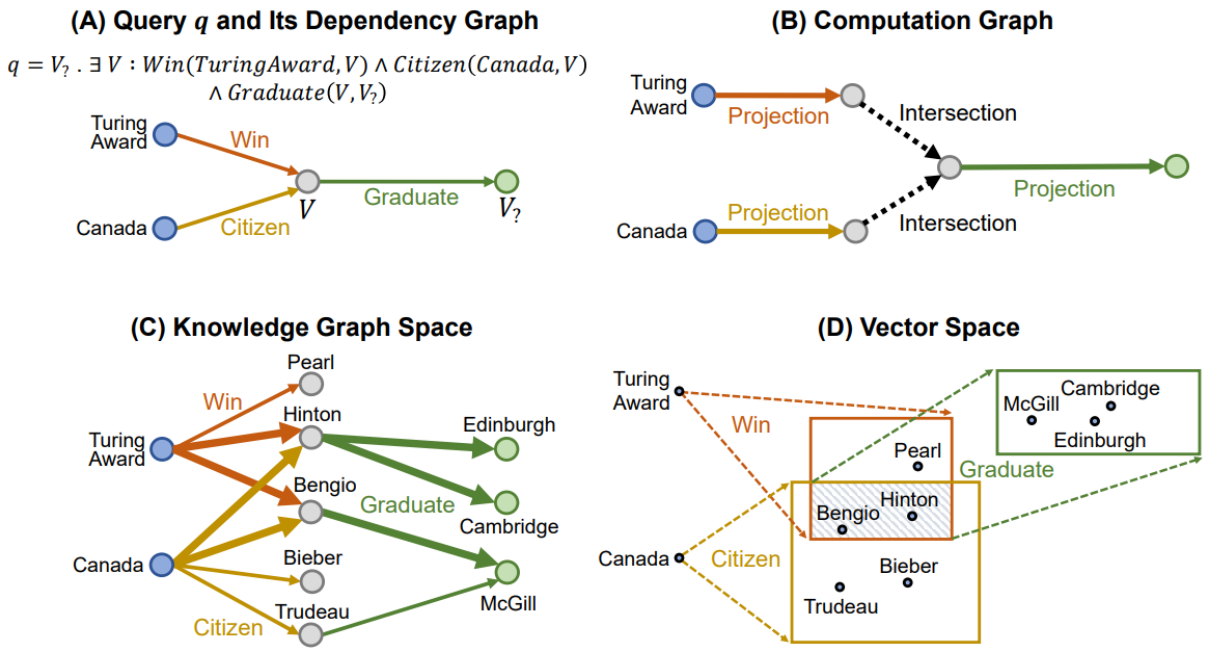


Figure 22: Answering complex queries on a knowledge graph with incomplete information can be difficult. One promising approach is to represent queries and answers as embeddings in a shared space. However, previous methods have treated queries as single points, which does not accurately reflect the fact that each query represents a set of possible answers and that logical operations on queries are operations on sets. Our proposed method, Query2box, addresses this issue by representing queries as hyper-rectangles (boxes) in the embedding space. This approach naturally handles existential quantification and conjunction through projection and intersection, respectively [61]

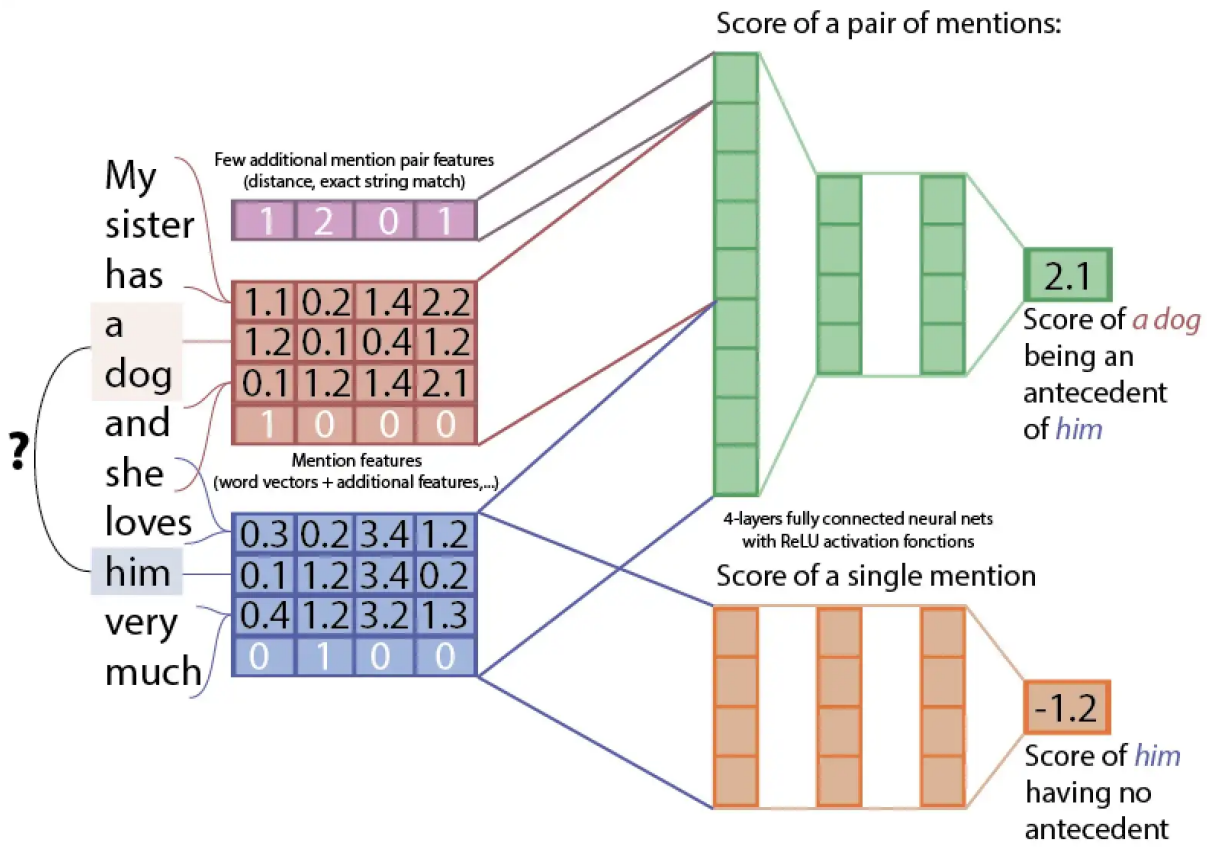


Figure 23: NeuralCoref Model Architecture [104]

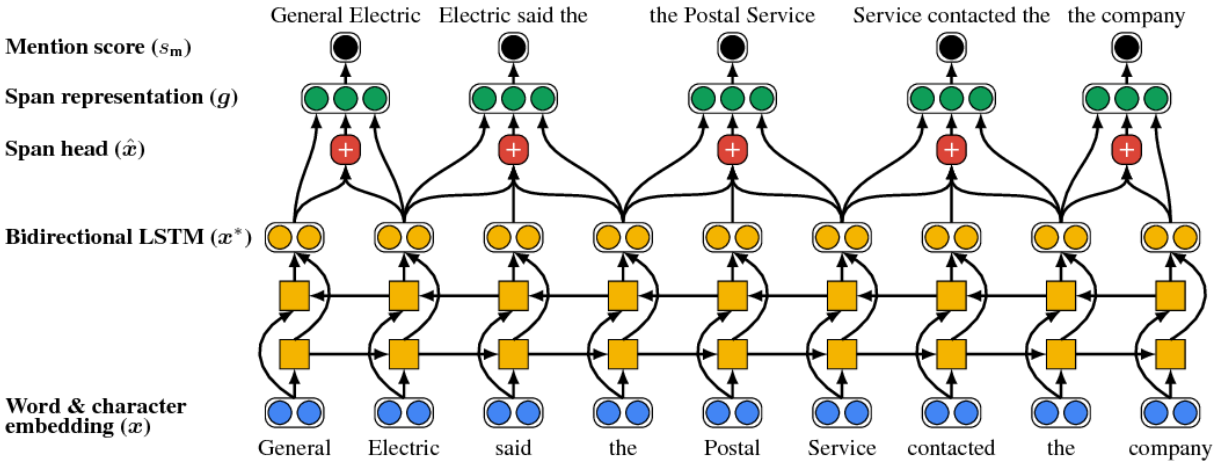


Figure 24: The e2e model's first step computes span embeddings to score potential entity mentions. Low-scoring spans are pruned to manage the number of spans being considered for coreference decisions. Only a subset of detectable spans are shown [14].

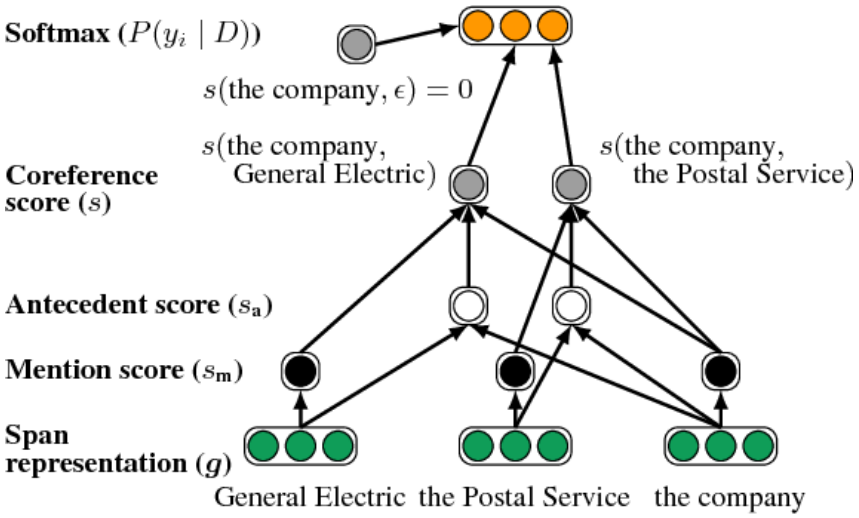


Figure 25: The second step of the e2e model calculates the antecedent scores based on pairs of span representations. The final coreference score for a pair of spans is obtained by adding the mention scores of both spans and their antecedent score as a pair [14].

on the mat," the root of the parse tree would represent the verb "sat," and the branches of the tree would represent the subject "the cat" and the object "the mat."

Both dependency parsing and constituency parsing are useful for natural language processing tasks, such as information extraction, machine translation, and text summarization. Both approaches involve analyzing the grammatical structure of a sentence and identifying the relationships between the words, which can be used to extract meaning from the text and perform various NLP tasks. However, while dependency parsing focuses on identifying the grammatical relationships between words in a sentence and representing them in a directed acyclic graph, constituency parsing involves identifying the hierarchical relationships between the words in a sentence and representing them in a tree-like structure. Both approaches allow for the analysis of the grammatical structure of a sentence and the extraction of meaning from the text, but they differ in the way they represent the relationships between the words [123].

A.5 Neural Parser

A.6 Detectron2 Models

Some of the popular object detection models available in detectron2 [113] include:

- Faster R-CNN: This model is based on a two-stage approach, where the region proposal network (RPN) is used to generate a set of candidate object bounding

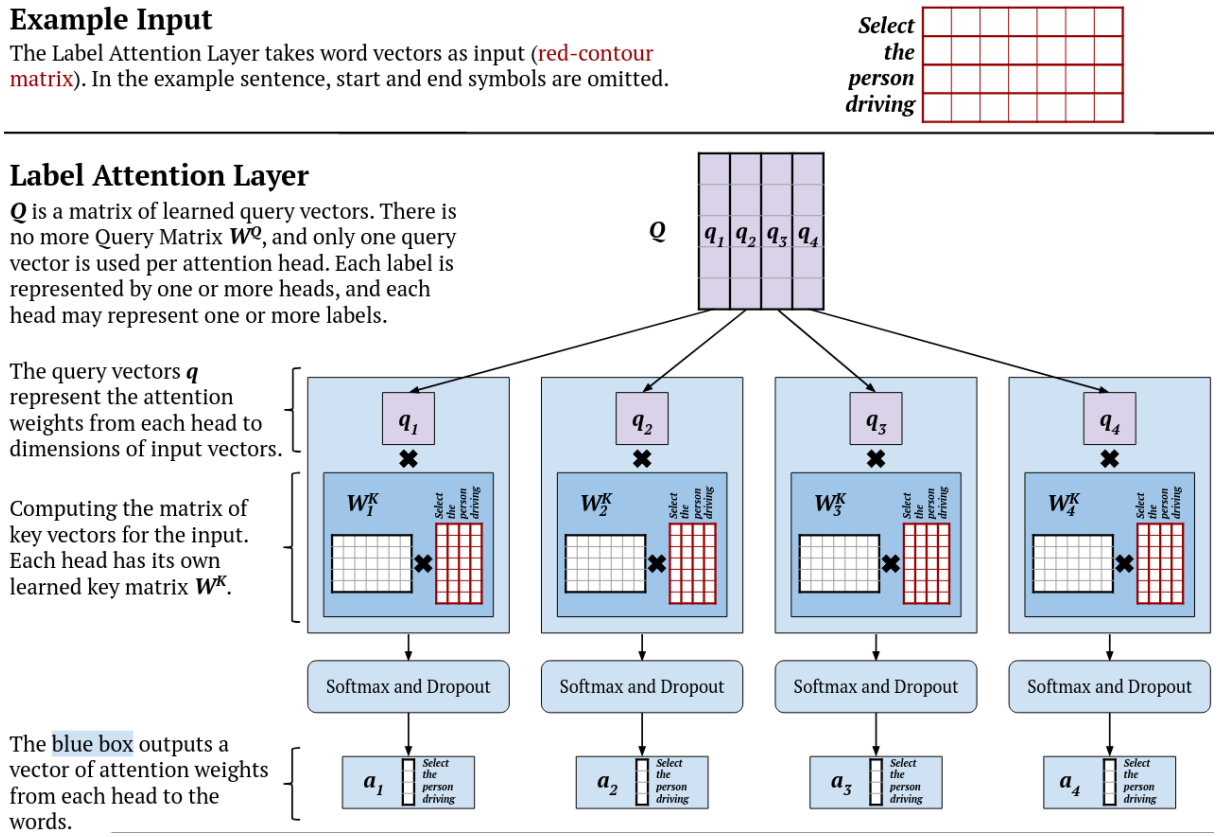


Figure 26: The architecture of the top of the Label Attention Layer from the LAL-Parser. In this figure, the example input sentence is "Select the person driving" [17].

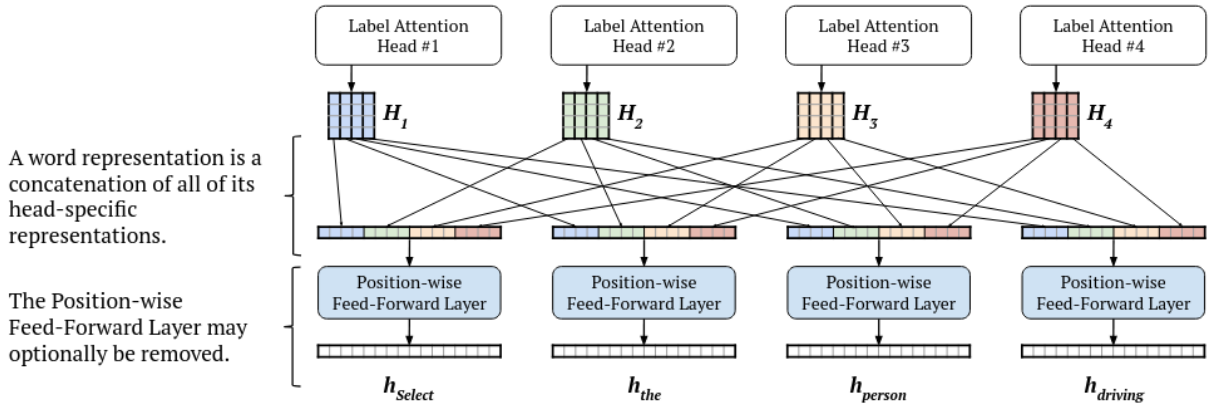


Figure 27: The head-specific word representations are redistributed to form word vectors through concatenation. Different colors are used for each label attention head to show where the head outputs go in the word representations. The vectors resulting from the position-wise feed-forward layer are not colored, as the head-specific information has been moved [17].

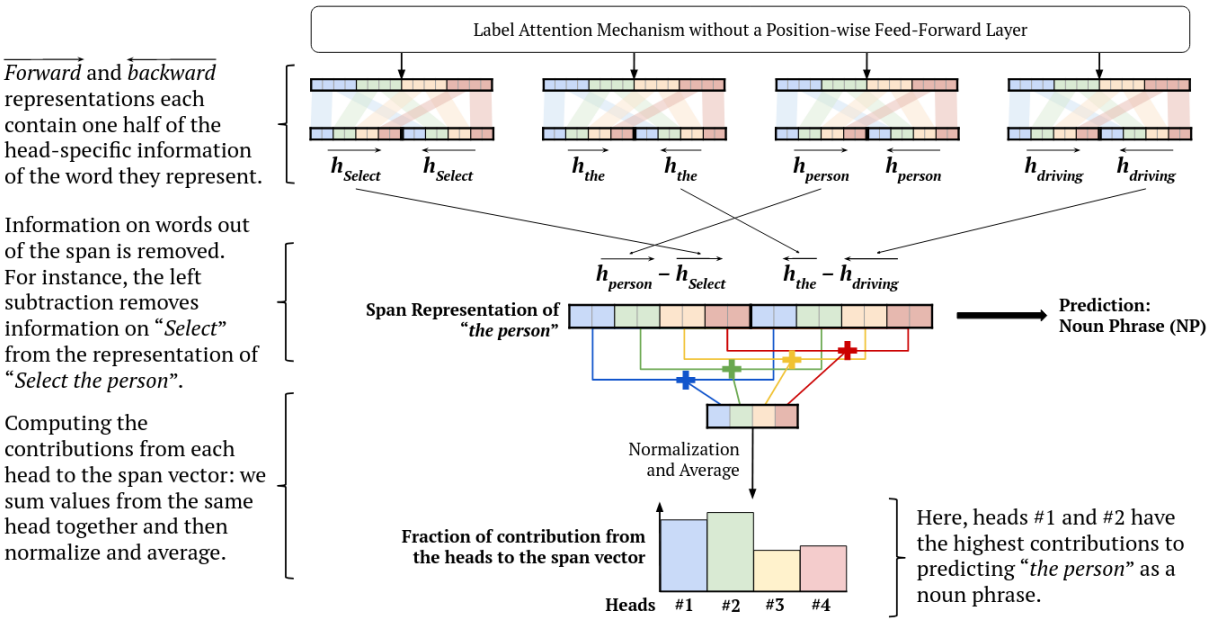


Figure 28: By removing the position-wise feed-forward layer, it is possible to calculate the contributions from each label attention head to the span representation, which allows for the interpretation of head contributions [17].

boxes, and then a separate network is used to classify and refine these bounding boxes.

- Mask R-CNN: This model is an extension of Faster R-CNN, which includes an additional branch for predicting an object mask in addition to the bounding box and class label.
- RetinaNet: This model is based on a one-stage approach, where the object detection task is performed directly in a single step using a dense set of anchor boxes.
- DensePose-RCNN: This model is used for instance segmentation of human bodies in images, where the goal is to predict a dense set of correspondences between the image and a 3D human body model.
- Panoptic FPN: This model is used for panoptic segmentation, which involves both instance segmentation and semantic segmentation of an image. It uses a combination of feature pyramid networks (FPN) and a bottom-up top-down attention mechanism to perform the segmentation.

References

[1] Feilong Chen, Xiuyi Chen, Fandong Meng, Peng Li, and Jie Zhou. Gog: Relation-aware graph-over-graph network for visual dialog. *CoRR*, abs/2109.08475, 2021.

[2] Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M. F. Moura, Devi Parikh, and Dhruv Batra. Visual dialog. *CoRR*, abs/1611.08669, 2016.

[3] A. M. TURING. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236):433–460, October 1950.

[4] Vishvak Murahari, Dhruv Batra, Devi Parikh, and Abhishek Das. Large-scale pre-training for visual dialog: A simple state-of-the-art baseline. *CoRR*, abs/1912.02379, 2019.

[5] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[6] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: visual question answering. *CoRR*, abs/1505.00468, 2015.

[7] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[8] Yue Wang, Shafiq R. Joty, Michael R. Lyu, Irwin King, Caiming Xiong, and Steven C. H. Hoi. VD-BERT: A unified vision and dialog transformer with BERT. *CoRR*, abs/2004.13278, 2020.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[10] Van-Quang Nguyen, Masanori Suganuma, and Takayuki Okatani. Efficient attention mechanism for visual dialog that can handle all the interactions between multiple

inputs. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*, pages 223–240. Springer, 2020.

[11] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[13] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[14] Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. End-to-end neural coreference resolution. *CoRR*, abs/1707.07045, 2017.

[15] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2017.

[16] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks, 2015.

[17] Khalil Mrini, Franck Dernoncourt, Trung Bui, Walter Chang, and Ndapa Nakashole. Rethinking self-attention: An interpretable self-attentive encoder-decoder parser. *arXiv preprint arXiv:1911.03875*, 2019.

[18] Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. Exploring visual relationship for image captioning. *CoRR*, abs/1809.07041, 2018.

[19] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Trevor Darrell, and Kate Saenko. Long-term recurrent convolutional networks for visual recognition and description. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634, 2015.

[20] Hao Fang, Saurabh Gupta, Forrest N. Iandola, Rupesh Kumar Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. From captions to visual concepts and back. *CoRR*, abs/1411.4952, 2014.

[21] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306, 2014.

[22] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014.

[23] Anna Rohrbach, Marcus Rohrbach, Niket Tandon, and Bernt Schiele. A dataset for movie description. *CoRR*, abs/1501.02530, 2015.

[24] Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond J. Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence - video to text. *CoRR*, abs/1505.00487, 2015.

[25] Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond J. Mooney, and Kate Saenko. Translating videos to natural language using deep recurrent neural networks. *CoRR*, abs/1412.4729, 2014.

[26] Gordon Christie, Ankit Laddha, Aishwarya Agrawal, Stanislaw Antol, Yash Goyal, Kevin Kochersberger, and Dhruv Batra. Resolving language and vision ambiguities together: Joint segmentation & prepositional attachment resolution in captioned scenes. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1493–1503, Austin, Texas, November 2016. Association for Computational Linguistics.

[27] Ronghang Hu, Marcus Rohrbach, and Trevor Darrell. Segmentation from natural language expressions. *CoRR*, abs/1603.06180, 2016.

[28] Chen Kong, Dahua Lin, Mohit Bansal, Raquel Urtasun, and Sanja Fidler. What are you talking about? text-to-image coreference. 06 2014.

[29] Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *CoRR*, abs/1505.04870, 2015.

[30] Vignesh Ramanathan, Armand Joulin, Percy Liang, and Li Fei-Fei. Linking people in videos with "their" names using coreference resolution. In *Computer Vision – ECCV 2014*, pages 95–110. Springer International Publishing, 2014.

[31] Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. Grounding of textual phrases in images by reconstruction. *CoRR*, abs/1511.03745, 2015.

[32] Ting-Hao (Kenneth) Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross B. Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, C. Lawrence Zitnick, Devi Parikh, Lucy Vanderwende, Michel Galley, and Margaret Mitchell. Visual storytelling. *CoRR*, abs/1604.03968, 2016.

[33] Harsh Agrawal, Arjun Chandrasekaran, Dhruv Batra, Devi Parikh, and Mohit Bansal. Sort story: Sorting jumbled images and captions into stories. *CoRR*, abs/1606.07493, 2016.

[34] Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. Analyzing the behavior of visual question answering models. *CoRR*, abs/1606.07356, 2016.

[35] Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. Are you talking to a machine? dataset and methods for multilingual image question answering. *CoRR*, abs/1505.05612, 2015.

[36] Abhishek Das, Harsh Agrawal, Larry Zitnick, Devi Parikh, and Dhruv Batra. Human attention in visual question answering: Do humans and deep networks look at the same regions? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 932–937, Austin, Texas, November 2016. Association for Computational Linguistics.

[37] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. *CoRR*, abs/1612.00837, 2016.

[38] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. *CoRR*, abs/1606.00061, 2016.

[39] Mateusz Malinowski and Mario Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. *CoRR*, abs/1410.0210, 2014.

[40] Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. Ask your neurons: A neural-based approach to answering questions about images. *CoRR*, abs/1505.01121, 2015.

[41] Mengye Ren, Ryan Kiros, and Richard S. Zemel. Image question answering: A visual semantic embedding model and a new dataset. *CoRR*, abs/1505.02074, 2015.

[42] Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Yin and yang: Balancing and answering binary visual questions. *CoRR*, abs/1511.05099, 2015.

[43] Yuke Zhu, Oliver Groth, Michael S. Bernstein, and Li Fei-Fei. Visual7w: Grounded question answering in images. *CoRR*, abs/1511.03416, 2015.

[44] André Calero Valdez, Martina Ziefle, and Michael Sedlmair. Priming and anchoring effects in visualization. *IEEE Transactions on Visualization and Computer Graphics*, PP:1–1, 08 2017.

[45] Donald Geman, Stuart Geman, Neil Hallonquist, and Laurent Younes. Visual turing test for computer vision systems. *Proceedings of the National Academy of Sciences*, 112(12):3618–3623, March 2015.

[46] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[47] Zahra Abbasiyantaeb and Saeedeh Momtazi. Text-based question answering from information retrieval and deep neural network perspectives: A survey. *CoRR*, abs/2002.06612, 2020.

[48] Chengchang Zeng, Shaobo Li, Qin Li, Jie Hu, and Jianjun Hu. A survey on machine reading comprehension: Tasks, evaluation metrics, and benchmark datasets. *CoRR*, abs/2006.11880, 2020.

[49] Oriol Vinyals and Quoc V. Le. A neural conversational model. In *ICML Deep Learning Workshop*, 2015.

[50] Haifeng Wang, Jiwei Li, Hua Wu, Eduard Hovy, and Yu Sun. Pre-trained language models and their applications. *Engineering*, September 2022.

[51] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *CoRR*, abs/1908.03265, 2019.

[52] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019.

[53] Vishvak Murahari, Prithvijit Chattopadhyay, Dhruv Batra, Devi Parikh, and Abhishek Das. Improving generative visual dialog by answering diverse questions. *CoRR*, abs/1909.10470, 2019.

[54] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903, 2022.

[55] Tom McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy, July 2019. Association for Computational Linguistics.

[56] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[57] R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. *CoRR*, abs/1902.01007, 2019.

[58] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA, 2008. Association for Computing Machinery.

[59] Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, sep 2014.

[60] Robyn Speer, Joshua Chin, and Catherine Havasi. ConceptNet 5.5: An open multilingual graph of general knowledge. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), February 2017.

[61] Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. *CoRR*, abs/2002.05969, 2020.

[62] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. QA-GNN: reasoning with language models and knowledge graphs for question answering. *CoRR*, abs/2104.06378, 2021.

[63] Todor Mihaylov and Anette Frank. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–832, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[64] Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. KagNet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th Interna-*

tional Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 2829–2839, Hong Kong, China, November 2019. Association for Computational Linguistics.

[65] Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. Scalable multi-hop relational reasoning for knowledge-aware question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1295–1309, Online, November 2020. Association for Computational Linguistics.

[66] Linjie Li, Zhe Gan, Yu Cheng, and Jingjing Liu. Relation-aware graph attention network for visual question answering. *CoRR*, abs/1903.12314, 2019.

[67] Weixin Liang, Yanhao Jiang, and Zixuan Liu. Graghvqa: Language-guided graph neural networks for graph-based visual question answering. *CoRR*, abs/2104.10283, 2021.

[68] Jiasen Lu, Anitha Kannan, Jianwei Yang, Devi Parikh, and Dhruv Batra. Best of both worlds: Transferring knowledge from discriminative learning to a generative visual dialog model. *CoRR*, abs/1706.01554, 2017.

[69] Qi Wu, Peng Wang, Chunhua Shen, Ian D. Reid, and Anton van den Hengel. Are you talking to me? reasoned visual dialog generation through adversarial learning. *CoRR*, abs/1711.07613, 2017.

[70] Satwik Kottur, José M. F. Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. Visual coreference resolution in visual dialog using neural module networks. *CoRR*, abs/1809.01816, 2018.

[71] Zhe Gan, Yu Cheng, Ahmed El Kholy, Linjie Li, Jingjing Liu, and Jianfeng Gao. Multi-step reasoning via recurrent dual attention for visual dialog. *CoRR*, abs/1902.00579, 2019.

[72] Dan Guo, Hui Wang, and Meng Wang. Dual visual attention network for visual dialog. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, August 2019.

[73] Feilong Chen, Fandong Meng, Jiaming Xu, Peng Li, Bo Xu, and Jie Zhou. DMRM: A dual-channel multi-hop reasoning model for visual dialog. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7504–7511, April 2020.

[74] Dan Guo, Hui Wang, Hanwang Zhang, Zheng-Jun Zha, and Meng Wang. Iterative context-aware graph inference for visual dialog. *CoRR*, abs/2004.02194, 2020.

[75] Shubham Agarwal, Trung Bui, Joon-Young Lee, Ioannis Konstas, and Verena Rieser. History for visual dialog: Do we really need it? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8182–8197, Online, July 2020. Association for Computational Linguistics.

[76] Zilong Zheng, Wenguan Wang, Siyuan Qi, and Song-Chun Zhu. Reasoning visual dialogs with structural and partial observations. *CoRR*, abs/1904.05548, 2019.

[77] Idan Schwartz, Seunghak Yu, Tamir Hazan, and Alexander G. Schwing. Factor graph attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[78] Xiaoze Jiang, Jing Yu, Zengchang Qin, Yingying Zhuang, Xingxing Zhang, Yue Hu, and Qi Wu. DualVD: An adaptive dual encoding model for deep visual understanding in visual dialogue. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11125–11132, April 2020.

[79] Xiaoze Jiang, Siyi Du, Zengchang Qin, Yajing Sun, and Jing Yu. KBGN: Knowledge-bridge graph network for adaptive vision-text reasoning in visual dialogue. In *Proceedings of the 28th ACM International Conference on Multimedia*. ACM, October 2020.

[80] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. OpenSurfaces. *ACM Transactions on Graphics*, 32(4):1–17, July 2013.

[81] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO captions: Data collection and evaluation server. *CoRR*, abs/1504.00325, 2015.

[82] Yijun Xiao and William Yang Wang. Quantifying uncertainties in natural language processing tasks. *CoRR*, abs/1811.07253, 2018.

[83] Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Generating images from captions with attention, 2015.

[84] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1060–1069, New York, New York, USA, 20–22 Jun 2016. PMLR.

[85] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *CoRR*, abs/1612.03242, 2016.

[86] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. *CoRR*, abs/1711.10485, 2017.

[87] Shikhar Sharma, Dendi Suhubdy, Vincent Michalski, Samira Ebrahimi Kahou, and Yoshua Bengio. Chatpainter: Improving text to image generation using dialogue. *CoRR*, abs/1802.08216, 2018.

[88] Po-Yao Huang, Junjie Hu, Xiaojun Chang, and Alexander Hauptmann. Unsupervised multimodal neural machine translation with pseudo visual pivoting. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8226–8237, Online, July 2020. Association for Computational Linguistics.

[89] Benyamin Ahmadnia and Bonnie J. Dorr. Augmenting neural machine translation through round-trip training approach. *Open Computer Science*, 9(1):268–278, January 2019.

[90] A. BELZ, T.L. BERG, and L. YU. From image to language and back again. *Natural Language Engineering*, 24(3):325–362, April 2018.

[91] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. From captions to visual concepts and back, 2014.

[92] Zhou Yu, Jun Yu, Yuhao Cui, Dacheng Tao, and Qi Tian. Deep modular co-attention networks for visual question answering. *CoRR*, abs/1906.10770, 2019.

[93] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[94] Prithvijit Chattopadhyay, Deshraj Yadav, Viraj Prabhu, Arjun Chandrasekaran, Abhishek Das, Stefan Lee, Dhruv Batra, and Devi Parikh. Evaluating visual conver-

sational agents via cooperative human-ai games. In *Proceedings AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, October 2017.

[95] Harm de Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron C. Courville. Guesswhat?! visual object discovery through multi-modal dialogue. *CoRR*, abs/1611.08481, 2016.

[96] David Schlangen. Grounded agreement games: Emphasizing conversational grounding in visual dialogue settings. *CoRR*, abs/1908.11279, 2019.

[97] Tianhao Yang, Zheng-Jun Zha, and Hanwang Zhang. Making history matter: Gold-critic sequence training for visual dialog. *CoRR*, abs/1902.09326, 2019.

[98] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *CoRR*, abs/1901.00596, 2019.

[99] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[100] Christopher Olah. Understanding lstm networks. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015.

[101] Arthur Brack, Daniel Uwe Müller, Anett Hoppe, and Ralph Ewerth. Coreference resolution in research papers from multiple domains. *CoRR*, abs/2101.00884, 2021.

[102] Khadiga Mahmoud Seddik and Ali Farghaly. Anaphora resolution. In *Natural Language Processing of Semitic Languages*, pages 247–277. Springer Berlin Heidelberg, 2014.

[103] Lu Liu, Zhenqiao Song, and Xiaoqing Zheng. Improving coreference resolution by leveraging entity-centric features with graph neural networks and second-order inference. *CoRR*, abs/2009.04639, 2020.

[104] Huggingface. GitHub - neuralcoref: Fast Coreference Resolution in spaCy with Neural Networks. `https://github.com/huggingface/neuralcoref`, [Accessed 24-Dec-2022].

[105] Kenton Lee, Luheng He, and Luke Zettlemoyer. Higher-order coreference resolution with coarse-to-fine inference, 2018.

[106] Ralph M. Weischedel, Eduard H. Hovy, Mitchell P. Marcus, and Martha Palmer. Ontonotes : A large training corpus for enhanced processing. 2017.

[107] Kevin Clark and Christopher D. Manning. Deep reinforcement learning for mention-ranking coreference models, 2016.

[108] Adriane Boyd. explosion/spacy: v2.3.9: Compatibility with numpy v1.24+, 2022.

[109] Dzmitry Bahdanau, Tom Bosc, Stanisław Jastrzebski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. Learning to compute word embeddings on the fly, 2017.

[110] Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[111] Sai Muralidhar Jayanthi, Danish Pruthi, and Graham Neubig. Neuspell: A neural spelling correction toolkit. *CoRR*, abs/2010.11085, 2020.

[112] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *CoRR*, abs/2001.05566, 2020.

[113] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[114] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[115] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[116]

[117] Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, Ziyue Huang, Qipeng Guo, Hao Zhang, Haibin Lin, Junbo Zhao, Jinyang Li, Alexander J. Smola, and Zheng Zhang. Deep graph library: Towards efficient and scalable deep learning on graphs. *CoRR*, abs/1909.01315, 2019.

[118] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[119] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR*, abs/1706.02677, 2017.

[120] Ye Liu, Wolfgang Maier, Wolfgang Minker, and Stefan Ultes. Naturalness evaluation of natural language generation in task-oriented dialogues using BERT. *CoRR*, abs/2109.02938, 2021.

[121] Ming Jiang, Qiuyuan Huang, Lei Zhang, Xin Wang, Pengchuan Zhang, Zhe Gan, Jana Diesner, and Jianfeng Gao. Tiger: Text-to-image grounding for image caption evaluation. *CoRR*, abs/1909.02050, 2019.

[122] Donatello Conte, Jean-Yves Ramel, Nicolas Sidère, Muhammad Muzzamil Luqman, Benoît Gaüzère, Jaume Gibert, Luc Brun, and Mario Vento. A comparison of explicit and implicit graph embedding methods for pattern recognition. In *Graph-Based Representations in Pattern Recognition*, pages 81–90. Springer Berlin Heidelberg, 2013.

[123] Dan Jurafsky and James H. Martin. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Pearson Prentice Hall, Upper Saddle River, N.J., 2009.