Commonsense Knowledge Generation and Analysis using Deep Learning Models

by

Navid Rezaei Sarchoghaei

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering

University of Alberta

Abstract

There has been a renewed interest in commonsense as a stepping stone toward achieving human-level intelligence. By digesting enormous amounts of data in different forms, such as visual, lingual, and sensory, humans are able to create a world model for themselves. It is hypothesized that this knowledge is the basis for commonsense, which can be defined as a collection of models of the world that know the plausibility of entities or interactions. This commonsensical world model helps humans learn new skills with very few trials, as they can predict the consequences of actions and plan and reason for the next steps.

In our work, we explore and experiment with how we can generate commonsense knowledge and how it can ultimately benefit deep learning models to gain commonsense. We also analyze the weaknesses of large language models (LLMs) in a commonsense context and provide solutions to improve LLMs in commonsensical tasks.

Inspired by how toddlers learn about their environment, we first introduce a methodology to generate commonsense knowledge using only visual input. We use knowledge graphs as the preferred method of data storage, as they are easy to access and require low time complexity to expand. We further expand the knowledge stored with plausibility weights of triples and contextual information.

As linguistics is an important next step in a toddler's mental model of the world, we experiment with transformer-based language models to expand the vision-based commonsense even further. Through experiments with language

models, we observe that larger language models, trained in an unsupervised fashion, have more embedded commonsense than their smaller counterparts. Symbolic and vetted storage of commonsense knowledge from different sources can improve commonsense capabilities in smaller language models. Resource-restricted use cases, such as smartphones or self-driving cars, benefit from offline smaller language models.

During our research, we noticed the high cost of human annotations to gather human commonsense datasets used to train language models. As a by-product of our research, we proposed a model-agnostic prompt technique to reduce costly human textual annotations for fine-tuning language models.

Lastly, we demonstrate that out-of-ordinary questions can throw the LLMs off guard. We illustrate how *negated complementary* questions adversely affect the model responses. We propose a model-agnostic methodology to improve the performance in *negated complementary* scenarios. Our method outperforms few-shot generation from GPT-3 (by more than 11 points) and, more importantly, highlights the significance of studying the response of large language models in different commonsensical scenarios.

Preface

This thesis is an original work by Navid Rezaei Sarchoghaei submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Software Engineering & Intelligent Systems. The thesis is written in paper-based format with the details as follows:

Chapter 3 of this thesis contains the article: N. Rezaei, M. Z. Reformat, and R. R. Yager, "Image-Based World-perceiving Knowledge Graph (WpKG) with Imprecision," in Information Processing and Management of Uncertainty in Knowledge-Based Systems, Springer International Publishing, 2020, pp. 415–428, ISBN: 978-3-030-50146-4. [1].

Chapter 4 of this thesis contains the article: N. Rezaei, M. Z. Reformat, and R. R. Yager, "Generating contextual weighted commonsense knowledge graphs," in Information Processing and Management of Uncertainty in Knowledge-Based Systems, Cham: Springer International Publishing, 2022, pp. 593– 606, ISBN: 978-3-031-08971-8. [2].

Chapter 5 of this thesis contains the article: N. Rezaei and M. Z. Reformat, "Utilizing language models to expand vision-based commonsense knowledge graphs," Symmetry, vol. 14, no. 8, 2022, ISSN: 2073-8994. DOI: 10.3390/sym14081715. [3].

Chapter 6 of this thesis contains the article: N. Rezaei and M. Z. Reformat, "Super-prompting: Utilizing model-independent contextual data to reduce data annotation required in visual commonsense tasks," arXiv preprint arXiv:2204.11922, 2022. [4].

Chapter 7 of this thesis contains the article: N. Rezaei and M. Z. Reformat, "Negated Complementary Commonsense using Large Language Models," submitted to ACL 2023 conference [under anonymity period]. [5].

Acknowledgements

To my supervisor: Thank you for always supporting me. I appreciate your supervisory approach, which allowed me to explore new and state-of-the-art topics. I enjoyed our weekly meetings and will very much miss our conversations. I appreciate your time, help, and support, especially during last-minute reviews, late-night meetings, or lengthy submission processes. Your positivity and good mood are so much contagious. I could have the worst day, but after meeting with you, I would be different, happy, and motivated! I always loved how you were a respectful and encouraging person and supervisor.

To my supervisory committee: Your feedback and support were valuable and insightful. The feedback helped me find the flaws in my work and resolve them as much as possible. But, more importantly, it helped me better articulate my research points and gave me the confidence to present them well.

To my dear wife: Thank you for your non-stop love, support, and trust during my Ph.D. journey. Gaining this degree would have been much more challenging without your support. It was great discussing my research with you, and I was fortunate that we were in almost the same field to understand each other's works. When the research was not going as expected, you still believed in me and encouraged me to continue.

I am glad the way we look at the world is very similar. It strengthens our bond and makes it easy to focus on what we both value the most. I hope to be as supportive as you are in your next educational and career steps. You deserve the best things available, and I know you can achieve them. I love you!

To my dear parents: I cannot forget the moment in Tehran airport when I looked at your faces from the other side of a thick plexiglass. I was talking

loudly and promising you to get my Ph.D. in engineering. But unfortunately, I could not hear you from the other side of the glass, and I think you were not hearing me too. Honestly, I had the most significant hesitance in my life. I was looking for any excuse not to leave you. How could I do that to you or myself? I was leaving all the love behind to embrace the unknown. The only glimmer of hope that I had was to make you happy.

I was blessed in my life to be raised by two superb teachers. Every moment of my life was a learning opportunity for me. I learned how to confront life's tough times by observing you. Life was not easy at the time. A cruel war was going on when I was born, and we had no say in it. However, you knew you had to raise us the best you could, even with all the sorrow of losing your family members in the war and the separations. The hard times did not end there; there was an aftermath. However, with all the difficulties, you still moved on and gave us the best life we could ever have. I am eternally thankful for all the lessons that you gave me during my lifetime. You were indeed the best life teacher that I could ever ask for.

To my dear daughter: Watching you grow and learn inspired my work. You observed the world with your beautiful eyes when you were so little. Looking at how you perceived the world inspired me to work on artificial intelligence systems that learn commonsense using only vision. Seeing how you connected words and visual perception of the world was fabulous. This beautiful combination inspired me to work on enhancing vision-based commonsense with lingual inputs. You are always a source of inspiration. I am always learning from you, and I love you!

To my dear brother and his family: You taught me the English alphabet, the capitals of cities, and many more. You were always there when I needed your help. I could depend on you and knew you always looked after me. You are the most supportive, kind, and fun big brother I could ever ask for. I am happy to have you and wish you all health, wealth, and happiness with your dear wife and daughter.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The construction of intelligent systems, especially with human-like intelligence, has been a long-term goal of many researchers, companies, and organizations. Whether it is possible to build a sentient artificial being can quickly become a more philosophical question than a scientific one. To the best of our knowledge, no proven intelligence upper-bound theory exists in artificial intelligence (AI) research, as the Shannon theorem in information theory [6], to specify the possible upper bound for achievable intelligence. However, with the current state of knowledge, we can define and synthesize intelligence to the best of our ability and scientifically improve on different factors.

Human intelligence has unique capabilities, such as learning, self-awareness, creativity, and commonsense. These categorizations may not be perfect, but they allow us to divide and conquer the critical task of creating intelligent systems. There has been much work on the learning aspects, which has been fruitful. For example, stochastic gradient descent has been a significant part of almost all modern deep-learning systems. Research on other aspects of intelligence is, however, in its infancy. Therefore, we narrow our focus to commonsense in AI systems. We try to answer these questions: What are some excellent methods to extract and represent commonsense from these models? Do modern deep learning models have inherent commonsense? How can the automatically-extracted commonsense benefit humans and deep learning models alike? A good definition paves the way to answer these questions better.

1.2 Research Overview

In this work, we focus on different aspects of commonsense and their relevance with deep learning models. We first focus on generating commonsense knowledge graphs from visual data using deep learning vision models. We then expand these vision-based commonsense knowledge graphs with language models. We also discuss and introduce methodologies to reduce the required annotation data in visual commonsense tasks. We finally analyze the existence of commonsense in large language models and specifically focus on the concept of *negated complementary* commonsense.

1.2.1 Commonsense Definition

Commonsense is trivial yet challenging as people rarely talk or write about it. As early as toddler years, humans can understand if something is out of place without even a developed language ability. Language can deepen the commonsense understanding and its formalization.

Yann LeCun, an inventor of convolutional neural networks, believes that a collection of world models representing what is likely, plausible, or impossible makes our commonsense [7]. John McCarthy classifies human commonsense into two categories of knowledge and ability. The commonsense ability is the action based on the gained commonsense knowledge [8].

The human experience gained through different senses, such as vision or touch, is the base for constructing commonsense knowledge. For example, knowing that fire is hot is common knowledge gained through touch and vision. Therefore, avoiding touching fire, not burning, is a commonsense ability based on this knowledge.

Commonsense knowledge is context-dependent. For instance, the people who live in the earth's northern hemisphere know the month of July to be a hot summer month, while the people in the southern hemisphere observe it as a cold winter month. Assuming no interaction between the people of the two hemispheres, the commonsense of these two groups is different about a specific month. Furthermore, context is not limited to physical or geographical loca-

tions but can also include temporal aspects. For example, it is more common to see more formally-dressed people in the 1960s than in the 2020s.

Commonsense knowledge is inherently uncertain. Moreover, the degree of correctness of commonsense knowledge depends on the joint group of observers. For instance, it is a common occurrence to see the sun covered by clouds but less common to be covered by the moon.

Commonsense can also be classified into different categories, such as physical interactions, order of events, and social dynamics. An example of a physical commonsense is observing desks in a classroom. Regarding the order of events, lighting a match to start a fire makes sense. Finally, in social constructs, saying thanks after receiving a gift is commonsense.

In the study of commonsense, we should also be alert to biases. For example, the existence of more women nurses or men construction workers should not result in biases towards genders, especially when developing AI systems that could one day have impactful decision-making.

1.2.2 Commonsense, Deep Learning, and Knowledge Graphs

Commonsense knowledge seems evident and natural for a human being, but this knowledge is challenging to be acquired by a machine. The gap in learning that type of information is filled out by techniques and methods linked to collecting and representing commonsense knowledge and self-supervised learning in the pre-training phase of models. New models, such as large language models, appear to have emerging capabilities, such as commonsense, while being scaled up. These newly-formed abilities are still under research and experimentation.

The existence of stand-alone commonsense databases is shown in the literature to benefit commonsense teaching to AI models ([9], [10]). Humans and automated methods can help improve the accuracy and knowledge in these commonsense databases. Both traditional and deep learning models can use explicit or acquired commonsense to do commonsensical reasoning.

There are usually two types of deep learning models, goal-specific (fine-tuned) and generic (pre-trained). While (smaller) goal-specific models can be

used to improve commonsense knowledge databases, larger generic models can directly exhibit commonsense. Furthermore, it is shown in different works, such as [11], that the larger the models, the better they are at more complex tasks, such as commonsense. Please note that pre-training is a specific task in itself, such as image classification in vision models or next token prediction in language models. Given the pre-training nature, the process can be supervised or self-supervised. Lately, the pre-training is focused on self-supervision, which means using available unlabeled data, such as web corpus in language models, and not being trained by specifically-labeled data, such as a labeled sentiment classification dataset.

Regarding stand-alone commonsense knowledge database structure, knowledge graphs have proven to be an effective data structure ([9], [10]). We follow the same pattern in our research. In a nutshell, commonsense knowledge graphs represent facts and relations between them, characterizing *real-world* scenarios and situations. Such graphs focus on elements and aspects related to everyday activities, arrangements, and natural circumstances. Things like: *flower in vase*, *tree has trunk*, *food on plate*, *shoe is less likely made of metal*, or *arm is most likely to be able to move, bend and be strong.*

The co-existence of external commonsense knowledge and deep learning models can benefit both. Deep learning models can learn from these databases to upgrade their knowledge. The commonsense knowledge database can also benefit from automated deep learning-based methods and inherent self-supervised knowledge from some deep learning models to enhance its knowledge.

1.3 Objectives

Our research focuses on analysis, extraction, and representation of commonsense with a focus on deep learning models and knowledge graphs. The ultimate goal is to identify approaches supporting the construction and expansion of commonsense knowledge graphs using contextual information drawn from images and large language models and to improve the performance of language models in general on commonsense tasks by identifying shortcomings

and proposing appropriate solutions.

To accomplish that, we envision the following tasks:

- Development of methodologies to generate commonsense similar to how toddlers visually learn about their environment; convolutional neural networks (CNNs) are utilized to construct vision-based commonsense knowledge graphs.

- Application of transformer-based language models to expand vision-based commonsense knowledge; Fine-tuning and automated generation of prompts are explored to enable extraction of information from the models.

- Reduction of cost in human commonsense data annotation, which is a vital part of commonsense research; Model-agnostic prompting methods are analyzed to reduce the time and cost needed to annotate data.

- Improvement of commonsense abilities in large language models (LLMs); commonsense abilities are analyzed in LLMs, and methods with low computation overhead are proposed to improve the LLMs' commonsense abilities.

1.4 Outline

This thesis has been prepared in a paper-based format and is organized as follows:

Chapter 2 is a brief review of the necessary vision and language background concepts required to understand the following chapters.

Chapter 3 is titled *Image-based World-perceiving Knowledge Graph (WpKG) with Imprecision*. Knowledge graphs are a data format that enables the representation of semantics. Most available graphs focus on representing facts, their features, and the relations between them. However, from the point of view of possible applications of semantically rich data formats in intelligent, real-world scenarios, there is a need for knowledge graphs that describe contextual information regarding realistic and casual relations between items in

the real world. This chapter presents a methodology for generating knowledge graphs addressing such a need. We call them *World-perceiving Knowledge Graphs – WpKG*. The process of their construction is based on analyzing images. We apply deep learning image processing methods to extract scene graphs. We combine these graphs and process the obtained graph to determine the importance of relations between items detected on the images. The generated WpKG is used as a basis for constructing possibility graphs. We illustrate the process and show some snippets of the generated knowledge and possibility graphs.

Chapter 4 is titled *Generating Contextual Weighted Commonsense Knowledge Graphs*. There has been a renewed interest in commonsense knowledge and reasoning. To achieve artificial general intelligence, systems must exhibit not only the recognition abilities of humans but also other essential aspects of being human, such as commonsense and causality. Recent literature has shown that external commonsense knowledge graphs benefit various systems in multiple ways, including improvements in the commonsense abilities of deep learning models. This chapter investigates an auto-generation of weighted commonsense knowledge graphs representing general information, as well as graphs containing contextual information. The method leads to constructing graphs with frequency-based weights associated with nodes and relations. The proposed construction methodology has the advantage of a never-ending learning paradigm. We evaluate the constructed contextual knowledge graphs qualitatively and quantitatively. The commonsense knowledge graphs are inherently explainable and can support commonsense reasoning. Finally, we analyze commonsense reasoning approaches using contextual graphs and discuss the results.

Chapter 5 is titled *Utilizing Language Models to Expand Vision-Based Commonsense Knowledge Graphs*. The introduction and ever-growing size of the transformer deep-learning architecture have had a tremendous impact not only in the field of natural language processing but also in other fields. The transformer-based language models have contributed to a renewed interest in commonsense knowledge due to the abilities of deep learning models. Recent

literature has focused on analyzing commonsense embedded within the pre-trained parameters of these models and embedding missing commonsense using knowledge graphs and fine-tuning. We base our current work on the empirically proven language understanding of very large transformer-based language models to expand a limited commonsense knowledge graph initially generated only on visual data. The few-shot-prompted pre-trained language models can learn the context of an initial knowledge graph with less bias than language models fine-tuned on a large initial corpus. It is also shown that these models can offer new concepts that are added to the vision-based knowledge graph. This two-step approach of vision mining and language model prompts results in the auto-generation of a commonsense knowledge graph well equipped with physical commonsense, which is human commonsense gained by interacting with the physical world. To prompt the language models, we adapted the chain-of-thought method of prompting. To the best of our knowledge, it is a novel contribution to the domain of the generation of commonsense knowledge, which can result in a five-fold cost reduction compared to the state-of-the-art. Another contribution is assigning fuzzy linguistic terms to the generated triples. The process is end-to-end in the context of knowledge graphs. It means the triples are verbalized in natural language, and after being processed, the results are converted back to triples and added to the commonsense knowledge graph.

Chapter 6 is titled *Super-Prompting: Utilizing Model-Independent Contextual Data to Reduce Data Annotation Required in Visual Commonsense Tasks*. Pre-trained language models have shown excellent results in few-shot learning scenarios using in-context learning. Although impressive, the size of language models can be prohibitive to make them usable in on-device applications, such as sensors or smartphones. With smaller language models, task-specific data annotation is needed to fine-tune the language model for a specific purpose. However, data annotation can have a substantial financial and time burden for small research groups, startups, and companies. In this chapter, we analyze different prompt-based fine-tuning techniques to improve results on both language and multimodal causal transformer models. We use a dataset focusing

on visual commonsense reasoning in time to evaluate our results. Our results show that by simple model-agnostic prompt-based fine-tuning, comparable results can be reached by only using 35%-40% of the fine-tuning training dataset. The proposed approaches result in significant time and financial savings. Furthermore, as the proposed methods make minimal architectural assumptions, other researchers can use the results in their transformer models with minimal adaptations.

Chapter 7 is titled *Negated Complementary Commonsense using Large Language Models*. Larger language models, such as GPT-3, have shown to be excellent in many tasks. However, we demonstrate that out-of-ordinary questions can throw the model off guard. This work focuses on finding answers to negated complementary questions in commonsense scenarios. We illustrate how such questions adversely affect the model responses. We propose a model-agnostic methodology to improve the performance in negated complementary scenarios. Our method outperforms few-shot generation from GPT-3 (by more than 11 points) and, more importantly, highlights the significance of studying the response of large language models in negated complementary questions.

Finally, Chapter 8 concludes the thesis and introduces possible future works.

NOTE: Throughout our research, we make use of human evaluations. We use Amazon SageMaker [1] and mTurk [2] platforms to conduct human evaluations. SageMaker or mTurk selected the human evaluators based on provided criteria. Our only criterion was to exclude minors. The human evaluator pool consists of an expert workforce trained on various machine learning tasks, as claimed by Amazon.

To better understand the connections and motivation of different chapters of this thesis, please refer to Fig.1.1.

[1] https://aws.amazon.com/sagemaker/data-labeling
[2] https://www.mturk.com/

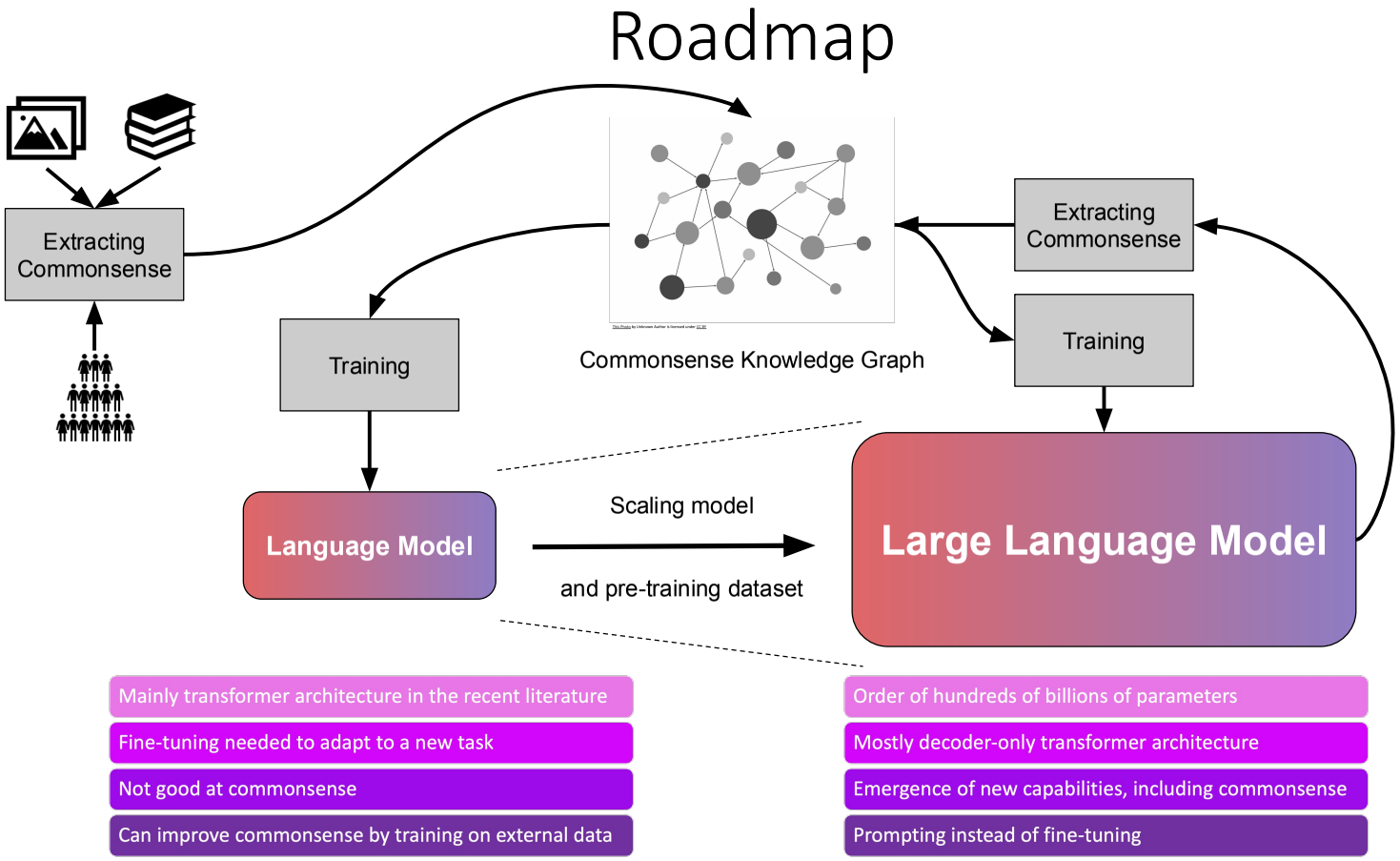


Figure 1.1: Roadmap.

Chapter 2

Background

2.1 Neural Network Basics

Each neuron in an artificial neural network imitates a biological neuron. Each neuron has a linear transformation section and an activation function, which is a non-linearity.

Each linear transformation consists of weights multiplied by each input and a bias added to the sum. The linear transformation aggregates all the input information.

Common activation functions are Sigmoid, Tanh, ReLU, and Leaky ReLU. Here is the sigmoid function as an example:

$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}. \tag{2.1}$$

By mixing these neuron units, we can create a neural network structure. A straightforward example of a neural network structure is a feed-forward network (or fully connected). In this type of network, neurons are arranged in layers, and neurons from each layer are all connected to the next and previous layers' neurons.

2.2 Computer Vision

Computer vision is the process of observing and processing visual data by machines. Visual data is usually considered part of the electromagnetic spectrum visible to humans.

Each image has different attributes that could be understood and processed by a vision algorithm. For example, if an image is described in the RGB color model (Red-Green-Blue), each pixel will have a specific value of these primary color elements between 0 and 255. This results in an image tensor of size [height, width, 3] that describes an image, where 3 is the number of color channels of RGB. For instance, the above image will be of a tensor of shape [1260, 750, 3] (2.835 million tensor elements).

Traditionally, there have been different methods to ingest and process these bits of data, e.g., edge detection, image spectrum, and other methods to find features in an image. However, unlike feature detection, data-driven approaches depend on a large set of training data, usually labeled, to recognize images and differentiate between their contents. The trained model is then evaluated on a smaller set of unseen testing data. Some examples of data-driven approaches are K-nearest neighbors, support vector machines (SVM), multi-layer perceptron neural networks, and convolutional neural networks (CNNs).

We need a task and a baseline to evaluate these models and see which one is better suited and superior in vision tasks. Image classification has been traditionally a foundational vision task as in PASCAL VOC[1] competition and the evolved ILSVRC[2] competition. Some reasons for its popularity are its vast application in different scenarios and the fact that many vision tasks, such as object recognition and image captioning, could be reduced to the image classification task. ILSVRC stands for ImageNet Large Scale Visual Recognition Challenge, which is an annual competition to do image classification on a subset of the relatively large ImageNet dataset[3]. The original ImageNet dataset is organized according to WordNet[4] hierarchy, where each word node of the tree has hundreds or thousands of image samples. Quantitatively, ImageNet has almost 22K categories and 14M image examples for them. The subset dataset of ImageNet used in the ILSVRC competition has 1,000 object classes

[1] http://host.robots.ox.ac.uk/pascal/VOC/
[2] http://www.image-net.org/challenges/LSVRC/
[3] http://www.image-net.org/
[4] https://wordnet.princeton.edu/

and almost 1.4M images, compared to only 20 object classes and almost 22K images used in the PASCAL VOC challenge.

In the initial years of the ImageNet classification challenge, the state-of-the-art algorithms used were data-driven but mainly focused on traditional machine learning algorithms, such as SVM. However, 2012 was a turning point, which reduced the classification error results from around 25% to only around 16%! The model used was called SuperVision (AlexNet). The model had five convolutional layers and three fully-connected layers. The computation was made possible using GPUs. It was trained on two NVIDIA GPUs for about a week. The model had 650K neurons, 60M parameters, and 630M connections. The final feature layer had a dimension of 4096. The winning models in the following years followed almost the same architecture but deeper. For example, 2014 winners GoogleNet and VGG had 22 and 16 deep CNN layers, respectively.

2.2.1 Convolutional Neural Networks (CNNs)

In the previous part, we reviewed a brief history of the state-of-the-art models based on their performance in the ImageNet image classification competition. We will go deeper into the recent deep convolutional models in future sessions. One notable breakthrough was the use of deep convolutional networks and GPUs that resulted in better-performing models, starting from AlexNet in 2012. In this part, we discuss further what a convolutional neural network is, and we will dissect its structure.

Convolutional Neural Networks, otherwise called CNNs or ConvNets, are similar to the basic neural network architecture in that it is made of neurons, connections, and non-linearities.

Multi-layer perceptrons (fully-connected neural nets) do not scale well with the size of the images. The reason is the vast number of connections between neurons of each layer with the next, where one neuron from the next layer is connected to all neurons from the previous layer. All the extra connections create more computation requirements and may even lead to overfitting due to many parameters.

Unlike a fully-connected neural network, a CNN accepts a 3-dimensional image tensor as input and, in the case of an image classification task, creates an n-dimensional vector with n being the number of classes. A simple CNN consists of the following layers:

- **Input**: This is the input image, e.g. in case of CIFAR-10, it is 32x32x3. 32 by 32 pixels is the size, and 3 represents three RGB channels.
- **Convolution Layer**: This layer uses a fixed-size filter to sweep through the image to calculate convolutions between the filter values and the image pixel values underneath. We will further describe this step as the most important concept in a CNN.
- **Non-linearity layer**: As in a normal neural network, different non-linearities could be used. For example, AlexNet uses ReLU non-linearity.
- **Pool layer**: The pooling layer is specific and sometimes optional for a CNN. This layer performs downsampling along the spatial dimensions.
- **Fully-Connected layer**: This is the last layer, which has an output of the size we need. For instance, in the case of CIFAR-10, the output will be of size [1, 1, 10], where each vector element represents a class score.

Convolution

To better understand the convolutional layer of a CNN, we go through some background, including continuous and discrete convolutions, as well as some applications of convolutions independent of neural networks.

One-Dimensional Continuous Convolution This procedure resembles the mathematical convolution, where a function is a slide through another function. At each point, the multiplication of one function by the reversed and slide version of the other is calculated and summed to calculate the value.

Mathematically, the formula is represented as (continuous functions):

$$\begin{aligned} f * g &= \int_{-\infty}^{+\infty} f(\tau) g(t-\tau)\, d\tau \\ &= \int_{-\infty}^{+\infty} g(\tau) f(t-\tau)\, d\tau \end{aligned} \tag{2.2}$$

Convolution is similar to a cross-correlation function with the difference that one function is reversed in convolution and then slides over the other function. In some sense, convolution is roughly a measure of the correlation between two functions. As we can see in the animations, this heuristics is almost true.

Two-Dimensional Discrete Convolution Two-dimensional discrete convolution is similar to one-dimensional discrete convolution but is applied to two dimensions. This convolution type is a good mathematical tool for image processing as images usually consist of two dimensions, and digital images are represented discretely. The mathematical representation is similar to the one-dimensional case:

$$\begin{aligned} (f * g)[m,n] &= \sum_{d_m=-\infty}^{+\infty} \sum_{d_n=-\infty}^{+\infty} f[m,n] \cdot g[d_m - m, d_n - n] \\ &= \sum_{d_m=-\infty}^{+\infty} \sum_{d_n=-\infty}^{+\infty} g[m,n] \cdot f[d_m - m, d_n - n] \end{aligned} \tag{2.3}$$

Convolutional Layer in CNN

In contrast with traditional methods, where filters are pre-defined for a specific task, filters in a convolutional neural network are learnable. This gives the advantage to the CNN to learn adaptive filters to accomplish the task given as a whole system by reducing the loss function defined.

The task of convolution is similar to what we discussed earlier. There are a few more parameters that are usually used when someone talks about a convolutional layer.

Stride Stride is the steps the kernel makes in the convolution process. If it is one, the kernel will move one by one, or if it is 2, it will move two steps simultaneously to sweep the input.

Padding Padding is the amount of zero-padding in edge locations of the input tensor.

Kernel Size This is something that we talked about earlier. The kernel size can be square or not.

Calculating convolution layer output The output of the convolutional layer can be calculated using the following formulas:

$$W_{new} = \frac{W - F + 2 * P}{S} + 1 \tag{2.4}$$

$$H_{new} = \frac{H - F + 2 * P}{S} + 1 \tag{2.5}$$

where H and W are the height and width of input, respectively. F is the filter size in the width or height dimension. P is the padding size. S is the stride amount.

Deep CNNs

In the previous section, we mentioned that deeper CNNs have better accuracy in image classification tasks than shallower ones. So, let us look at what happens under the hood in a deep convolutional neural network to gain more insight into CNNs and how they work.

CNNs, as part of the whole data-based methods, rely on data, cost function, and training to achieve a higher-accuracy model. Looking into different layers of CNNs from the image to the one-hot vector in the case of image classification, we see a notable specific pattern.

The first layer applies a specific number of trainable kernels to the image. For example, 16 different kernels of size 3×3 could create 16 different channels as a result. As mentioned previously, these kernels have traditionally been used to perform specific tasks on images, such as edge detection. Herefore, it is not

surprising that this layer of deep CNN architecture focuses more on lower-level features, such as edges.

Well-known Convolutional Neural Network Architectures In this section, we go through some at-the-time state-of-the-art convolutional neural network architectures that performed well on the image classification task, especially the ImageNet competition.

The ILSVRC challenge happened in 2017 until model ability in classifying ImageNet dataset images surpassed human ability on that specific task. The overall trend consistent with the winners after 2012 is an increase in the number of layers until the architectures reach a deep 152-layer architecture. Beyond 2017, the challenge is still open on Kaggle[5].

We go through some prominent or ground-breaking architectures to understand what makes them accurate or efficient.

AlexNet (SuperVision) This 2012-winner architecture significantly reduced classification error from the 2011's state-of-the-art model in the ILSVRC competition. The main novelty in this model is using a convolutional neural network in relatively deep architecture, compared to the 2011 winner.

This architecture uses two CNN layers with max pooling and normalization, followed by three more CNN layers and a max pool. Lastly, three fully-connected layers gradually reduce the 4096-wide neuron array to 1000 neurons. There are 1000 classes, and the last layer shows the class scores.

AlexNet was the first model to use ReLu as non-linearity. Normalization layers are not common anymore. A lot of data augmentation resulted in good results. The dropout probability was 0.5, and the batch size was 128. The learning rate was 0.01, with a one-tenth reduction when accuracy flattens.

VGG Compared to AlexNet, the VGG architecture is deeper (16 or 19 layers). The kernels (filters) used are also smaller (3x3) than the first CNN layer of AlexNet, which was a square size of 11. It is shown that a stack of 3x3

[5]https://www.kaggle.com/c/imagenet-object-localization-challenge/overview

convolutional layers with a stride of 1 has the same effective receptive field as one 7x7 convolutional layer. By effective receptive field, we mean the region in the input affected by an operation. The stack of three 3x3 convolutional layers also uses fewer parameters than a 7x7 convolutional layer. VGG19 is only slightly better than VGG16 but uses more memory.

Inception (GoogLeNet) The Inception model is deeper than VGG, with 22 layers. The architecture does not have any fully-connected layers. The architecture has 5 million parameters, 12 times less than AlexNet.

The idea with Inception architecture is to develop a good module and then stack it multiple times. The developed module applies convolutional filter operations in parallel in multiple receptive fields of 1, 3, and 5 and a max polling operation. All the results are concatenated depth-wise. Although this module is effective, it is very expensive computationally. Therefore, some dimension reduction is also performed.

ResNet This architecture made higher numbers of layers possible. Previously, more layers could have added more to the accuracy; e.g., as mentioned before, VGG19 did not provide significant accuracy improvement compared to VGG 16. Deeper plain architectures perform worse than shallower ones with the same number of iterations. The problem is not overfitting, as this worse performance happens on training and testing datasets. The deeper networks seem harder to optimize, and even need to remember the achieved results from shallower layers.

ResNet architecture has some bypass paths parallel to the regular path of convolutional modules, where the exact input gets added to the output of the convolutional module. This way, information from previous layers have a higher chance of being forgotten in the deeper layers because of operations applied to them.

In the ResNet model, each residual block has two 3x3 convolutional layers. For deeper models, the bottleneck approach of using 1x1 convolutions to make dimensionality reduction is used to improve efficiency, similar to the Inception

model. After every convolutional layer, batch normalization is performed. Xavier initialization of parameters is used. Stochastic gradient descent is used with a momentum of 0.9. The learning rate chosen is 0.1, with a reduction of 10 when the validation error flattens. Mini-batch size is 256. No dropouts were used. Weight decay of 1e-5 is used.

Transfer Learning

The shallower layers of a deep convolutional neural network learn more about the basic features of images, such as lines and corners. The deeper layers however are more detailed and can classify a whole eye. This understanding allows us to use some of the learnings from pre-trained models and apply them to a new dataset.

Let us say we want to classify Oxford-IIIT pet dataset[6]. This dataset has 37 pet classes, with roughly 200 images for each class. The images have large variations in scale, pose, and lighting. We can either do complete training or transfer learning. Transfer learning can be generally done with two methods of *feature extracting* and *fine-tuning*.

One way to do that is to start with a new architecture and train it from scratch. This, of course, is possible but may result in more training time and comparably low precision of results due to the limited number of data.

Another method is to use a pre-trained model on ImageNet and just adapt it for our purpose. We know that lines, corners, and other basic elements are similar in the two datasets. As ImageNet also has many animal images, even deeper layers can be reused as well. We need to replace some deepest layers to adapt the model to have an output vector with the size of the classes. We used the pre-trained model as a feature extractor in this method.

The other transfer learning method is when we want to have a fully or partly retrained network. One example is when the target dataset is different from the pre-trained model. In this case, we may freeze the initial layers that do a line or corner detections but will retrain all the deeper layers with our new dataset. We can compare this method to a good initialization for our

[6]https://www.robots.ox.ac.uk/ vgg/data/pets/

network. We usually randomly initialize the weights and biases. However, we now initialize the network closer to the possible solution (global optimum) and continue the training until we reach that optimum point.

2.2.2 Data Augmentation

When there is not much data, or we want more data to train our models, we can do data augmentation to increase the number of data samples we have artificially. By data augmentation, we mean changing images to look new or similar to the model so that it can see the samples from different angles, sizes, and even environments.

Using the best types of data augmentation depends on the dataset itself. There is also a fixed number of data augmentation types. Here we look at some of the common data augmentation techniques.

Color Changes

We can also augment images by changing their colors. Here are some examples:

- Color jitter: This method randomly changes the brightness, contrast, hue, and saturation of images.
- Random grayscale: This method randomly converts images to grayscale with a probability of p.

Flipping and Translating

Flips can be done in different orientations. For example, *random horizontal flip* and *random vertical flip* randomly flip the image horizontally and vertically, respectively.

The images can also be rotated to a specific degree or even translated and scaled using the *random affine* method.

Generation

Added to the normal methods of augmentation using the same photo, we can also generate new photos to create more augmented data. We can think of

two methods: generating new photos and using the same photos in different scenarios.

New Generated or Artificial Images Recent generative models can generate new images by looking at a training dataset and generating new similar photos. The same type of mechanism could be used to create extra training data.

Another method is to use artificial images instead of real ones to train the models. For example, this method trains self-driving cars in a simulator and then does extra training in reality.

Same Images in Different Situations Research has also shown that simply copying and pasting images in different environments could even increase the accuracy of the vision models.

2.3 Natural Language Processing

There are different common tasks in natural language processing (NLP). Some are easier than others, and some can be categorized as medium or hard difficulty.

Spell checking, keyword searching, and finding synonyms are relatively easy. Parsing information from websites, documents, and other information sources is relatively moderate. Some of the more challenging tasks in NLP are as follows:

- Translation between languages.
- Semantic analysis to find the real meaning of the text.
- Finding reference of pronouns. This task is called Coreference.
- Answering questions about a text.

The ability to work with natural language equips us to target other useful applications, such as spam detection and medical report analysis, as some examples.

2.3.1 Word Vectors

Representing words as vectors is an effective way to extract semantic knowledge from them. By vectors, we mean n-dimensional arrays that can be defined by hand or better learned from natural language. There are different ways to represent words in vectors.

One simple way is to represent each word as a one-hot-vector. By this, we mean representing each word as a vector with lots of zeros and a one. The vector's dimension in this method will be huge and equal to the number of tokens available in a specific language, e.g., English. Although we accomplish the task of representing words with vectors, the vectors have a computationally huge dimension. Furthermore, the one-hot-vectors do not embed any relationship information in themselves, e.g., no information of how man and father may be related.

Another way to find good word representations is to look at their context. As John Rupert Firth, a famous English linguist from the 1950s, says: "You shall know a word by the company it keeps".

The context can mean different things, such as the whole document or surrounding words.

Based on this idea, the goal is to have similar word vectors, otherwise called embeddings, for words that usually happen in the same context.

Two ways to calculate word embeddings are:

- Dimensionality Reduction Methods
- Iteration-Based Methods

Dimensionality Reduction Methods We can define a word's context by its surrounding words. As a hyper-parameter, the context window can be adjusted. For example, we can look at one or more surrounding words and consider the context of a specific word.

Using this context, we can go through a large text and count the co-occurrence of words. Then, these co-occurrences can be organized in a large matrix called a co-occurrence matrix. The same words can be organized in

rows and columns in this matrix, and the counts can be placed in the matrix accordingly.

With a large matrix, we can reduce the dimensionality by using a matrix dimensionality reduction method called SVD and extracting word embeddings. These embeddings benefit from smaller dimensions as well as contextual information embedded.

The co-occurrence matrix is the basis for a famous word embedding method called GloVe[7]. The gloVe has been trained on large text corpora, such as Wikipedia, Gigaword, Twitter, and Common Crawl.

Iteration-Based Methods Iteration-based methods can learn word embeddings by choosing a cost function and learning the word embeddings iteration by iteration without looking at the overall data. This process makes it computationally less intensive than co-occurrence matrix methods. One famous method to calculate word vectors using this method is called Word2Vec.

Word2Vec uses two different methods to calculate word embeddings:

- Continuous Bag-of-Words (CBOW) tries to predict the center word using the surrounding ones.
- Skip-gram does the opposite of CBOW and tries to predict the context words from the center word.

We can see that finding word embeddings has been simplified to a prediction task. Prediction is what machine learning, specifically neural networks, does well if trained well and a good model is chosen.

2.3.2 Language Models and Text Generation

By modeling a language, we mean assigning a probability to the occurrence of a set of words following each other to make a phrase. For example, given a set of words, what word can come next to complete the sentence?

In other words, each language model predicts the probability of the occurrence of the next word given an observed sequence of words. Having this

[7]https://nlp.stanford.edu/projects/glove/

probability, we can even generate a sentence by choosing one of the most probable next words.

It is worth mentioning that creating a language model and generating text can be seen at different levels, e.g., word, character, or sub-words. For instance, a language model can be trained on characters only and, given a set of characters seen, predict what next character suits the sequence well.

N-Gram Language Model

One method to model a language is to use n-gram models. N-grams are a group of n words, such as one (unigram), two (bigram), three (trigram), or four (4-gram).

By counting the occurrence of different n-grams in a large text, we can calculate the occurrence probability of the next word, given an observed n-gram.

The model is simple and understandable. However, its space complexity is large and keeps growing with the data size because we need to store n-grams and possible words after each n-gram with their probabilities. As a result, the text generated using this model is better than other models, such as recurrent neural networks.

Another problem with the n-gram model is the data sparsity issue. By sparsity, we mean lack of occurrence of a certain sequence of words. If the sequence does not happen, the probability of the next word cannot be calculated as well. The issue worsens when n gets larger because larger unique sequences will have lower and lower probabilities of occurrence.

Fixed-Window Neural Language Model

Another method to build a language model is looking at a fixed word or token window. Serializing the embeddings to a simple feed-forward neural network with one hidden layer.

Unlike an n-gram language model, we do not need to store the large token series occurrence statistics, which reduces the space complexity of the model substantially. The lack of a specific sequence of tokens will not be an issue;

therefore, the data sparsity issue is resolved. However, the fixed window could be more flexible and is also small. Increasing the window size results in a larger weight matrix and extra computational requirements. Another issue is the need for more symmetry in this model, as each word is treated differently by being multiplied in different trained weights.

RNN Language Model

Recurrent Neural Networks (RNNs) fix some of the shortcomings of the previous vanilla neural network model.

The previous neural language model could only process a fixed size of text. However, RNNs can process different sequence sizes. Increasing the size of the sequence does not affect the model size. In RNNs, the idea is to use the same weights for all the tokens. This results in solving the symmetry issue. RNNs can also theoretically store data from many steps back in a sequence. The issues in RNN models are related to computation speed and accessing information from steps much farther in the past. RNNs have other variants, such as LSTMs and GRUs as well.

Transformer-based Language Models

Modern language models are based on the transformer architecture introduced in [12]. The extensive pre-training in these models has become the de-facto standard for processing natural language text. We use and focus on this type of language model in this thesis.

Chapter 3

Image-based World-perceiving Knowledge Graph (WpKG) with Imprecision

3.1 Introduction

Knowledge graphs are composed of a set of triple relations, i.e. $<$*subject* – *predicate* – *object*$>$, where subjects and objects are items connected via predicates representing relations between them. The graphs are useful in representing data semantics and are employed in different applications, such as common-sense and causal reasoning [13], [14], question-answering [15], natural language processing [16], and recommender systems [17]. Some examples of existing knowledge graphs are DBpedia [18], Wikidata [19], Yago [20], the now-retired Freebase [21], and WordNet [22]. The aforementioned knowledge graphs contain information about facts, their features, and basic relations between them. They focus on people, geographical locations, movies, music, and organizations and institutions. They are missing a piece of information about everyday real-world items, their contexts, and arrangements.

From the human perspective, we can state that the visual information plays a significant role in human learning processes [23]. At the same time, the eye's information transfer rate is quite high [24] that makes a visual stimulus to be of significant importance in processes of gaining understanding about different items and how they are related to each other. Given the importance of visual data, it is appealing to develop systems that could observe, learn and create

knowledge based on such data. Additionally, traditional knowledge graphs do not provide any degree of confidence associated with relations. It is assumed that all of them are equally important.

In this paper, we look at the task of creating knowledge graphs based on visual data. The idea is to process images, generate scene graphs from them, and aggregate these graphs. Graphs constructed in such a way contain knowledge about everyday objects, their contexts and their situational information, as well as information related to the importance of common-sense relations between multiple objects in their natural scenarios.

We call such a graph *World-perceiving Knowledge Graph*, *WpKG* in short. The quality and suitability of knowledge we retrieve from images depend on the capability of tools and methods we use for image processing. Processing an image means generating a scene graph representing relations between objects/entities present on this image. Once numerous images are processed, all scene graphs are aggregated. This alone allows us to treat the process of constructing graphs via aggregation as the human-like process of learning via processing of observed images.

We also look at a process of using knowledge graphs – *WpKG*s – to construct possibility graphs reflecting conditional dependencies between sets of entities as observed in their usual environments. The information about the importance of relations allows us to build possibilistic conditional distributions. They are used for processing and reasoning about entities and relations between them in their own relevant contexts. The included case study shows an application of the presented procedure to Visual Genome (VG) dataset [25].

3.2 Related Work

Extracting information from different media to create a knowledge graph has been examined in the literature. Yet, the area of focus of these works has been different: some of them focus on images, some on text, and some on a combination of both. Also, the methods used for information retrieval can be different – automatic or manual. A brief overview is presented in subsection

3.2.1.

Possibilistic knowledge bases and graphs are important forms representing uncertainty of data and information [26], and [27]. A set of basic definitions is included in the following subsections.

3.2.1 Knowledge Graph Construction

There is a number of different knowledge graph generation methods that focus on text as the source of information, such as NELL [28], ConceptNet [9], ReVerb [29], and Quasimodo [30]. Some other published approaches, such as WebChild KB [31], [32] or LEVAN [33], extract knowledge from text and image captions or only from image objects without in-image relations. Probably, the most relevant work to our work is NEIL [34], which create a knowledge graph directly from images.

Compared to NEIL, our proposed automatic approach is capable of extracting much more types of object-to-object relations. Compared to ConceptNet, which represents an example of a semi-automatic method of retrieving knowledge from text, our proposed approach can extract common-sense relations based on only observing visual data.

3.2.2 Possibilistic Knowledge Base

A possibilistic base is a set of pairs (p, α) where p is a proposition, and α is a degree to which p is true and is in the interval $(0, 1)$ [26]. Let Ω be a set of interpretations of the real world, and possibilistic distribution π a mapping from Ω to the interval $(0, 1)$. An interpretation ω that satisfies p has $\pi(\omega) = 1$, and $1 - \alpha$ when ω fails to satisfy p. In summary:

$$\begin{aligned} \forall \omega \in \Omega, \pi_{\{p\ \alpha\}}(w) = 1 \qquad & if\ \omega \models p \\ = 1 - \alpha\ & otherwise \end{aligned}$$

From now on, we identify the base as $\sum = \{(p_i, \alpha_i), i = 1, \ldots, n\}$. Then all interpretations satisfying propositions in $\sum$ have the possibility degree of 1, while other interpretations are ranked based on the highest values of α

associated with proposition they do not satisfy, i.e., $\forall\omega \in \Omega$:

$$\begin{aligned}\pi_{\sum}(w) &= 1 && if\ \omega \models \sum \\ &= 1 - max\{\alpha_i : (p_i, \alpha_i) \in \sum\ and\ \omega \models \neg p_i\}\ otherwise\end{aligned}$$

In other words, $\pi_{\sum}$ induces a necessity 'grading' of p_i that evaluates to what extent p_i is a consequence of the available knowledge. The necessity measure *Nec* is:

$$Nec_{\pi_{\sum}}(p_i) = 1 - max\{\pi_{\sum}(\omega) \mid \omega \models \neg p_i\}$$

Based on that, we can say that (p_i, α_i) is a plausible conclusion of $\pi_{\sum}$ if

$$Nec_{\pi_{\sum}}(p_i) > Nec_{\pi_{\sum}}(\neg p_i)$$

and $Nec_{\pi_{\sum}}(p_i) \geq \alpha_i$ [35].

A possibility distribution $\pi_{\sum}$ is *normal* if there is an interpretation ω that it totally possible, i.e., $\pi_{\sum}(\omega) = 1$.

3.2.3 Possibilistic Graph

A possibility graph ΠG is an acyclic directed graph [26]. The nodes of such a graph are associated with variables A_i, each with its domain D_i; while its edges represent dependencies between elements of nodes. For the case of binary variables, i.e., when $D_i = \{a_i, \neg a_i\}$, the assignment of value to the variable is called an interpretation ω. Let us denote a set of nodes that have edges connecting them to a node A_i as its parents: $Par(A_i)$. Possibility degrees Π associated with nodes are:

for each node A_i **without a parent** $Par(A_i) = 0$ prior possibility degrees associated with a single node are $\Pi(a)$ for every value $a \in D_i$ of the variable A_i; possibilities must satisfy the normalization condition: $max_{a \in D_i} : \Pi(a) = 1$.

for each node A_j **with parent(s)** $Par(A_j) \neq 0$ possibility degrees are conditional ones $\Pi(a|\omega_{Par(A_j)})$ where $a \in D_j$, and $\omega_{Par(A_j)}$ is an element of

the Cartesian product of domains D_k of variables $A_k \in Par(A_j)$; as above, conditional possibilities must satisfy the normalization condition: $max_{a \in D_j} : \Pi(a|\omega_{Par(A_j)}) = 1$.

In our case, a conditional probability measure is defined using *min*:

$$\Pi(p|q) = 1 \qquad if\ \Pi(q \wedge p) = \Pi(q)$$
$$= \Pi(q \wedge p)\ otherwise$$

and obeys [26]:

$$\Pi(q \wedge p) = min\{\Pi(p|q), \Pi(q)\}$$

3.3 Generation of Image-based *WpKG*

We introduce a systemic approach to generate knowledge graphs given visual data. Such graphs provide us with contextual information about objects present in the world with very limited input from humans. There are unique challenges associated with the generation of this type of graph. First, we need methods able to detect objects in images, and second, we require tools to extract relations between the detected objects.

Once we have the object recognition and relation extraction processes, we execute them on a set of images. The obtained triples – <*entity – relation – entity*> are aggregated into a single knowledge graph. The strength of relations is determined by the number of co-occurrences of objects with specific relations. The overall process is shown in Fig. 3.1.

Having a trained model, the process is liberated from specific visual data and its annotations. Additionally, more visual data can be processed using the proposed methodology and comprehensive context-specific knowledge graphs could be created.

3.3.1 Detection of Objects

To detect objects and their corresponding bounding boxes, we use the Faster R-CNN model [36]. In this model, the full image is passed through a convolutional neural network (CNN) to generate image features. To detect image

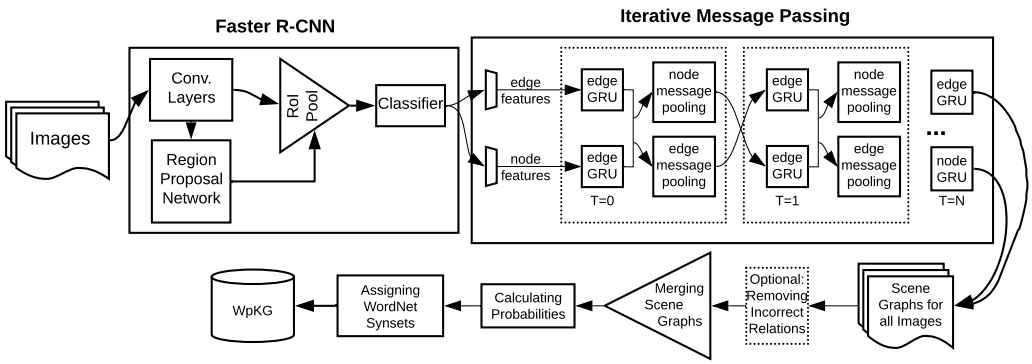


Figure 3.1: Overall procedure for generation of a knowledge graph from images

features, usually a pre-trained CNN, such as VGG network [37], trained on ImageNet [38] is used. Given the image features as input, another neural network, called Region Proposal Network (RPN), predicts regions that may contain an object and their corresponding bounding boxes. This learning network is the principal contribution of the Faster R-CNN model compared to the Fast R-CNN model [39]. This results in an improvement of performance in both training and inference. The regions of interest (RoIs) are then mapped into the image feature tensor, and via application of a process called RoI Pooling the regions are downsampled to be fed to the next neural network. This allows for the prediction of image classes and their correct bounding boxes. Given the error losses from the classification and bounding box predictions, the entire network is trained end-to-end using backpropagation and stochastic gradient descent (SGD) [40]. An illustration of the process can be found in Fig. 3.1.

3.3.2 Identification of Relations between Objects

Determining relations between objects is required to generate scene graphs and it can be done in several ways. There has been several publications that propose such methods as Iterative Message Passing [41], Neural Motifs [42], Graphical Contrastive Losses [43], and Factorizable Net [44]. In our work, we use the Iterative Message Passing model.Â

The Iterative Message Passing model predicts relations between objects de-

tected by the Faster R-CNN model. Mathematically, a scene graph generation process means finding the optimal $\mathbf{x}^* = \arg\max_{\mathbf{x}} \Pr(\mathbf{x}|I, B_I)$ that maximizes the following probability function:

$$\Pr(\mathbf{x}|I, B_I) = \prod_{i \in V} \prod_{j \neq i} \Pr(x_i^{cls}, x_i^{bbox}, x_{i \to j}|I, B_I). \tag{3.1}$$

where I is an image, B_I represents proposed object boxes, $\mathbf{x}$ is a set of all variables, including classes, bounding boxes and relations ($\mathbf{x} = \{x_i^{cls}, x_i^{bbox}, x_{i \to j} | i = 1 \ldots n, j = 1 \ldots n, i \neq j\}$), with n representing the number of proposed boxes, x_i^{cls} as a class label of the i-th proposed box, x_i^{bbox} as the offset of bounding box relative to the i-th proposed box, and $x_{i \to j}$ as a predicate between the i-th and j-th proposed boxes.

3.3.3 Aggregation of Scene Graphs

The process of amalgamating generated image scene graphs that results in a single knowledge graph has a number of challenges: 1) establishing a unique identifier for each entity; 2) identifying the importance of connections; 3) dealing with missing values and incorrect data; and 4) keeping the knowledge graph updated in presence of new data.

In the specific case of the Visual Genome dataset, we use *synsets* from WordNet to identify nodes and relations, as well as different meanings of a specific word. There are various methods to identify the uniqueness of words, such as using words occurring in natural language, grouping similar words with the same meaning, or trying to assign words to their specific synsets. Yet, another way is to keep words and phrases as they are and let their occurrence numbers show the importance of connections and nodes. Such a simple approach provides a good indication which relations are more likely to occur.Â

Another challenge is to mitigate missing or incorrect information. For example, the used methods/models could incorrectly label objects/relations and the processes could fail to find unique words or synsets. Even the hand-annotated data in the Visual Genome (VG) dataset [45], which is used for training, has missing and incorrect data [46]. The unknowns are reduced by

relying on the information already present, such as recovering a missing synset based on an already-known name to synset relation or WordNet.Â

3.4 Image-based *WpKG*: Experimental Studies

The Iterative Message Passing model [41] is trained on the VG dataset. It contains 108,077 images that capture everyday scenarios. For evaluation, only the most common 150 object categories and 50 predicates are used.

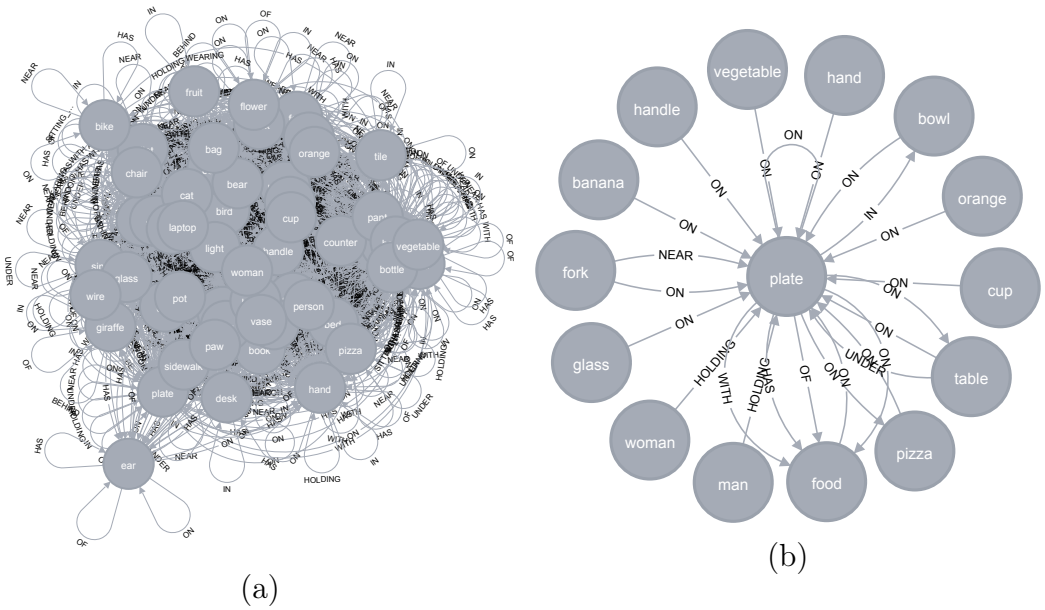
The Faster R-CNN model that is applied to detect objects and their bounding boxes is pre-trained on MS-COCO dataset. This dataset has 80 object categories. The training set is of size 80k images. Validation and test sets are 40k and 20k images, respectively. Around thirty percent of the VG dataset (test set) is used to detect objects and predict predicates. The subset has around 30,000 images. Running the process described in Section 3.3, a *WpKG* with 138 nodes and 7,287 relations is generated.Â

Neo4j [47] software is used to store and analyze the generated graph. It allows us to store object and relationship names and synsets, as well as occurrence numbers. Also, it visualizes a structure composed of triples *subject-predicate-object*.

One advantage of the generated *WpKG* is the existence of common sense relations occurring in the actual world extracted during the processing of visual data. The most important entities related to the entity of interest can be found by inspecting the strength of connections between them. One way to accomplish this is to measure how often these objects are associated with each other.

As an example, the entity *plate* together with the related entities is shown in Fig. 3.2 (a). As we can see, removing non-frequent relations leads to identification of tightly related objects relevant to the *plate*, Fig. 3.2 (b).

A sample of relation occurrence statistics is shown in Table 3.1. Based on the analysis of visual data, we can find out about some common-sense knowledge, such as places where a *vase* can be placed, and what can be put into it. Most of the relations, such as flower-in-vase, make sense and agree with



(a) (b)

Figure 3.2: Relationships to/from *plate* entity: with at least one instance and interrelationships between associated items (a); and at least 10 instances for each relationship and without the interrelationships (b).

the crowd-sourced VG dataset. However, some relations, such as vase-in-vase, may not make sense. This could be a shortcoming of the method/model used for prediction of relations. Besides a better model, processing more images and detecting more types of relations and objects may improve the results.

The comparison of our method, which is based on image processing, with other relevant automatic and semi-automatic methods is demonstrated in Table 3.2.

3.5 *WpKG*-based Possibilistic Graph and Base

The generated *WpKG*s consist of an enormous amount of nodes and relations. The relations – as built via aggregation of scene graph relations – contain information about the frequency of occurrence. This means that each relation is equipped with a weight indicating its strength and importance. For practical use, *WpKG* can be further processed and a subset of nodes together with relations between them can be used to construct a possibilistic graph.

	subject	predicate	object	occurrence number
woman	woman	wearing	shirt	192
	woman	holding	umbrella	168
	woman	has	hair	141
plate	plate	on	table	388
	plate	on	plate	193
	plate	on	pizza	19
flower	flower	in	vase	173
	flower	on	table	41
	flower	on	tree	15
vase	vase	on	table	116
	vase	in	vase	44
	vase	has	flower	31

Table 3.1: Three most common relations in *WpKG* generated using Faster R-CNN and iterative message passing models to recognize objects and predict relations, respectively.

Method	**In-Image Relations**	**Input Source(s)**	**Relation Types**	**Triples**	**Automation**
NEIL [34]	**Yes**	Image	< 10	$< 10K$	**Automated**
ConceptNet [9]	No	Text	< 100	**34M**	Semi-Automated
LEVAN [33]	No	Text and Objects in Images	< 10	$< 100K$	**Automated**
WebChild KB 2.0 [32]	No	Text and Image/Video Captions	> 1000	$> 18M$	**Automated**
Quasimodo [30]	No	Text (logs and QA forums)	**Dynamic**	$2.3M$	**Automated**
Our Work	**Yes**	Image	< 50 (**Dynamic**)	$> 7K$	**Automated**

Table 3.2: Comparison of relevant generated knowledge graphs from literature. Our method and NEIL are the ones that focus on in-image relations.

3.5.1 Extracting Possibilistic Graph from *WpKG*

A *WpKG* is constructed with no constraints. It contains cycles, very strong and weak relations, as well as erroneous information due to the imperfection of used image processing tools. In that context, as possibilistic graph is more organized and clean. Therefore, extracting nodes and edges from *WpKG* and building a graph that satisfies rules of the possibilistic graph (Section 3.2.3) seems important steps in utilizing generated *WpKG*s.

First, a proto-possibilistic graph is constructed. It is free of cycles and contains outwards relations linked to the entity of interest. The procedure used to extract relevant entities and connections is presented as Algorithms 1 and 2. The important aspects of this process are:

Algorithm 1, line 4 the value of *Depth* identifies the allowed length of a 'relation chain' at the process of building a graph;

Algorithm 2, line 6 the procedure *randomize_createGroups*() is crucial in the construction process: 1) randomization of a sequence of entities allows to generate graphs with different paths; once this is combined with a process presented in line 8 (explained below) it prevents the existence of cycles in the generated graph; 2) grouping of relations/predicates connected to the same object, i.e., prepositions/adjectives playing the role of relations; as illustration, see entities *flower, window, table, plant*, Fig. 3.3;

Algorithm 2, line 8 this allows to solve an issue of cycles, i.e., relations between pairs of entities *flower-vase*, *plant-vase* and *table-vase*, Fig. 3.3, would lead to cyclic directed graph; however, if a connection between both entities already exist, a new one – in the opposite direction – is not created.

The application of the presented procedure leads to a graph that is acyclic and direct. It also contains occurrences associated with each connection. The last step of constructing a possibilistic graph is to determine possibility degrees. To do so, all input connections to a given node are analyzed. The maximum

Algorithm 1: Construction of Proto-Possibilistic Graph

Data: Image-based **WpKG**; Seed_Entity; Depth
Result: Proto-Possibilistic Graph

```
1  begin
2      root ← Seed_Entity;
3      d ← 1;
4      call CreateConn(root, d, Depth);
5      return;
```

Algorithm 2: Generation of Connections

```
1  CreateConn(r, d, Depth)
2  begin
3      if d <= Depth then
4          listChild ← create_list_children(r);
5          if listChild ≠ ∅ then
6              randomize_createGroups(listChild);
7              for e ∈ listChild do
8                  if e not_connected_to r then
9                      setupConnection(r, e);
10                 call CreateConn(e, d + 1, Depth);
11         else
12             return;
13     else
14         return;
15     return;
```

value is identified and is used for normalization of all other occurrence values associated with inward connections to the node. This ensures satisfaction of the requirement of maximum possibility equal to 1.0 (Section 3.2.3).

3.5.2 Construction of Possibilistic Base

The extracted possibilistic graph allows us to build a possibilistic knowledge base. Here, we follow the process presented in [26]. For that purpose, we consider the graph as a set of triples:

$$\Pi G = \{(a, P_a, \alpha) : \Pi(a|P_a) = \alpha\}$$

where a is an instance of A_i and P_a is the Cartesian product of domains D_k of variables $A_k \in Par(A_i)$. Each such triple can be represented as a formula:

$$(\neg a \vee \neg P_a, 1 - \alpha)$$

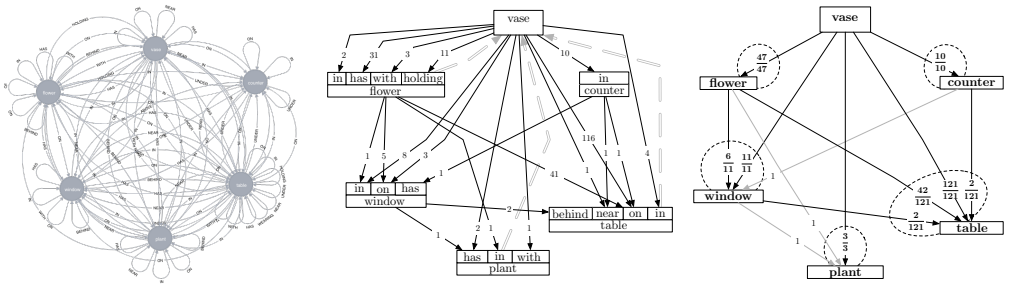
so, following [26], we have that the possibilistic knowledge base associated with ΠG defined as:

$$\sum = \{(\neg a_i \vee \neg P_{a_i} 1 - \alpha_i) \ / \ (a_i, P_{a_i}, \alpha_i) \in \Pi G\}$$

3.6 Possibilistic Graph and Base: Experimental Studies

Let us illustrate the process of building a simple possibilistic graph and a possibilistic knowledge base. We apply the procedure to build a graph of facts related to the entity *vase*, and relations between this entity and other entities from the vase's environment.

Application of Algorithm 1 to the generated *WpKG* allows us to extract entities related to the entity of interest, *vase*. The Neo4j snapshot of *WpKG* with *vase* and relations to 'relevant' entities is shown in Fig. 3.3(a). The version processed by the algorithm is shown in Fig. 3.3(b). It contains – marked as dashed lines – the pairs *flower-vase*, *plant-vase*, and *table-vase* that could result in different graphs depending on the element of randomness embedded in the procedure $randomize\_createGroups()$, Algorithm 2.



(a) *WpKG* (b) proto-possibilistic graph (c) possibilistic graph

Figure 3.3: A fragment of *WpKG* and a possibilistic graph constructed based on it.

The *WpKG* with occurrences assigned to connections allows us to determine conditional degrees. We have simplified our graph, i.e, combined all inward connections to a node into a single one, as shown in Fig. 3.3(c). This graph is further processed – the occurrence numbers are used to determine possibility values. Based on the graph in Fig. 3.3(c), we build conditional possibility degrees. All of them are presented in Tables 3.3, 3.4, and 3.5.

(a) Π(Vase)

vase	1.
¬ vase	1.

(b) Π(Counter|Vase)

Counter|Vase	vase	¬ vase
counter	1.	1.
¬ counter	1.	1.

(c) Π(Flower|Vase)

Flower|Vase	vase	¬ vase
flower	1.	1.
¬ flower	1.	1.

(d) Π(Plant|Vase)

Plant|Vase	vase	¬ vase
plant	1.	1.
¬ plant	1.	1.

Table 3.3: Possibility degrees for *Vase*, *Flower*, *Counter*, and *Plant*.

The last step of our case study is dedicated to the construction of a possi-

Window|Vase,Flower	**vase** ¬ flower	¬vase, **flower**	elsewhere
window	1.	.545	1.
¬ window	1.	1.	1.

Table 3.4: Possibility degrees for *Window* – Π(Window|Vase,Flower)

Shelf|Window,Vase, Flower,Counter	**window**, ¬ vase, ¬ flower, ¬ counter	¬ window, **vase**, ¬ flower, ¬ counter	¬ window ¬ vase, **flower**, ¬ counter	¬ window ¬ vase, ¬ flower, **counter**	elsewhere
table	.017	1.	.347	.017	1.
¬ table	1.	1.	1.	1.	1.

Table 3.5: Possibility degrees for *Table* – Π(Table|Window,Vase,Flower,Counter)

bilistic knowledge base, Section 3.5.2. As a result, we obtain:

$$\begin{aligned}\sum = \{\ & (\neg window \vee vase \vee \neg flower,\ .455),\\ & (\neg table \vee \neg window \vee vase \vee flower \vee counter,\ .983),\\ & (\neg table \vee window \vee \neg vase \vee flower \vee counter,\ .653),\\ & (\neg table \vee window \vee vase \vee flower \vee \neg counter,\ .983)\ \}.\end{aligned}$$

3.7 Conclusion

The paper focuses on the automatic construction of a knowledge graph – called *World-perceiving Knowledge Graph (WpKG)* – that contains results of the analysis of multiple images. Further, the generated *WpKG* is processed and multiple possibilistic graphs can be constructed based on it.

It is shown that using deep learning models, we can extract common-sense situational information about objects present in visual data. The trained neural networks may already know these relations implicitly, but extracting this knowledge in the form of a knowledge graph provides the ability to have this information explainable and explicit. The strength of the overall procedure depends on the capabilities of the applied learning model as well as the data it has been trained on. By improving the models themselves, the overall procedure can be improved.

Constructed *WpKG*s are contextualized by images used as an input to the presented process. A different graph will be obtained when images representing a specific geographical location are used, while a different graph will be built

based on images illustrated a specific historical event. Also, multiple different possibilistic graphs can be created to reason about the correctness of contextual utilization of specific items and relations between them.

Given the adaptability of *WpKG* to new scenarios, context-aware and even time-variant knowledge graphs can be constructed. For example, processing car images from a specific country will lead to the construction of *WpKG* representing a very specific information related to cars' details and their contextual settings. Another important aspect that can be considered is time. It can affect both occurrences of relations and meanings of words linked to the nodes.

As future work, better models can be used to improve the overall construction process, biases can be reduced by implementing procedures to diversify the input images, and prediction of unknown objects can be added.

Chapter 4

Generating Contextual Weighted Commonsense Knowledge Graphs

4.1 Introduction

There has been a renewed interest in commonsense as part of achieving human-like intelligence. Recent literature, [48] [49], has shown the importance of commonsense knowledge graphs in training artificial intelligence (AI) models with commonsense. Knowledge graphs (KGs) are a semantically rich representation of data where each piece of information is represented as a triple made of a subject (s), predicate (p), and object (o) as in $s \rightarrow p \rightarrow o$. Subjects and objects are graph nodes, while predicates are edges.

This paper introduces a methodology for generating contextual weighted commonsense KGs using vision-based deep learning models. Among different types of commonsense, such as social interactions and events, we focus on generating the physical commonsense graphs.

The choice of visual data to generate commonsense knowledge is an attempt to mimic the human way of learning. Even before developing linguistic skills, a human toddler acquires a good grasp of physical commonsense, for example, position of items in relevance to each other. This commonsense knowledge is further solidified by interacting with the real world and developing language to gain extra knowledge.

4.1.1 Defining Commonsense

First, it is imperative to have a clear definition of what commonsense is. A good definition paves the way to discuss the details better.

Commonsense, as John McCarthy puts it, can be distinguished in two different aspects: knowledge and ability [8]. The knowledge refers to the common knowledge gained, and the ability means to act based on the knowledge gained.

Yann LeCun, one of the inventors of convolutional neural networks, defines commonsense as a collection of models of the world, which can lead us to know what is likely, what is plausible, what is impossible [7].

The human perception gained through different senses, such as vision or touch, is the base for constructing commonsense KGs. This knowledge is context-dependent and can differ between individuals. For instance, the people who live in the northern hemisphere of the earth know the month of July to be a hot summer month, while the people in the southern hemisphere observe it as a cold winter month. Context is not limited to physical or geographical locations, but it can also include temporal aspects. For example, it is more common to see more formally dressed people in the 1960s than in the 2020s.

Additionally, unlike factual knowledge, commonsense knowledge is inherently uncertain. More probable phenomena are more likely part of common knowledge; while, less probable occurrences make less sense for most people.

We can also classify commonsense into different categories, including physical concepts, events, and social interactions. An example of physical commonsense is seeing desks in a classroom, and of an event one is lighting a match to start a fire. In social constructs, saying thanks after receiving a gift is commonsense.

The goal of this work is to automatically extract contextual commonsense knowledge from visual data and represent it in the format of a knowledge graph. The graph has weighted edges that illustrate levels of certainty in the extracted information. Commonsense graphs are used for many purposes, including reasoning, which is shown here.

4.1.2 Contributions

As described in the introduction, commonsense knowledge is context-dependent, whether the context is temporal or spatial. To the best of our knowledge, no recent paper has focused on generating commonsense knowledge adapted to specific contexts. Current commonsense KGs, such as ConceptNet [9], lack contextual awareness.

This paper introduces a methodology to auto-generate contextual weighted commonsense knowledge using images from different contexts. We focus on the feasibility and correctness of the results obtained instead of the size of the graph.

Commonsense is inherently uncertain and subject to the amount of processed data. In order to capture this, we propose three ways to assign weights to graph edges (triples) based on the information extracted from the images. We evaluate these weighting methods to see how they resonate with human commonsense.

Using the proposed method, we generate a context-free weighted commonsense KG, as well as contextual weighted commonsense KGs for 93 physical contexts. Given the contextual commonsense KGs, we show different reasoning examples using possibilistic theory.

The code and the data are available under `https://github.com/navidre/contextual_commonsense_kg`.

4.2 Related Work

There have been several works in the literature regarding gathering and constructing commonsense KGs.

ConceptNet [9] originated from the crowdsourced project of Open Mind Common Sense has grown into an extensive KG that includes some degree of commonsense. It comprises 36 relations focusing on taxonomies, lexical knowledge, and physical commonsense knowledge. ConceptNet v5.7 includes around 3.4 million triples. It combines crowdsourcing with other databases, such as DBPedia, WordNet, Wikitionary, and Open Cyc. As highlighted in

[48], almost 90% of the triples are related to taxonomies and lexical data.

NEIL method [34] focuses on extracting object relationships from images. It has resulted in a graph with 10,000 triples built around ten types of relations. Another process, LEVAN [33], uses a semi-automatic text processing and results in a KG with less than 100 predicate types and around 34 million triples. WebChild KB 2.0 [32] uses text and image/video captions to automatically mine commonsense knowledge. The result is a graph with over 1,000 predicate types and 18 million triples. Quasimodo [30] uses texts in logs and forums to mine commonsense. The result is 2.3 million triples, while the number of predicate types is dynamic. TransOMCS [50] mines linguistic graphs to generate a graph similar to ConceptNet with about 18.48 million triples. Such processes can be supported by other image-specific tasks [51].

World-perceiving KG (WpKG) [1] introduces a methodology to auto-generate commonsense KGs purely based on visual data. The commonsense KG includes 50 predicate types, 150 entity types, and over 7,000 triples. It can be expanded based on the vocabulary and the underlying scene graph generation models.

A recent body of work, [48] [49], focuses on annotating commonsense KGs to use them for training language models to predict commonsense results given a subject and a predicate. The human-annotated KGs are typically in the size of millions and cover social interactions, events, and entity commonsense.

4.3 Methodology

We have introduced a method for generating commonsense KGs from images using scene graph generation models in [1]. In this paper, we build on the foundations of the previously published work.

The introduced process starts with detecting objects in a set of images and predicting relationships between them. Then, based on the obtained results, it creates scene graphs (Section 4.3.1) – one per image. Finally, the method aggregates the scene graphs into a final weighted commonsense KG (Section 4.3.2). The overall process is depicted in Fig. 4.1.

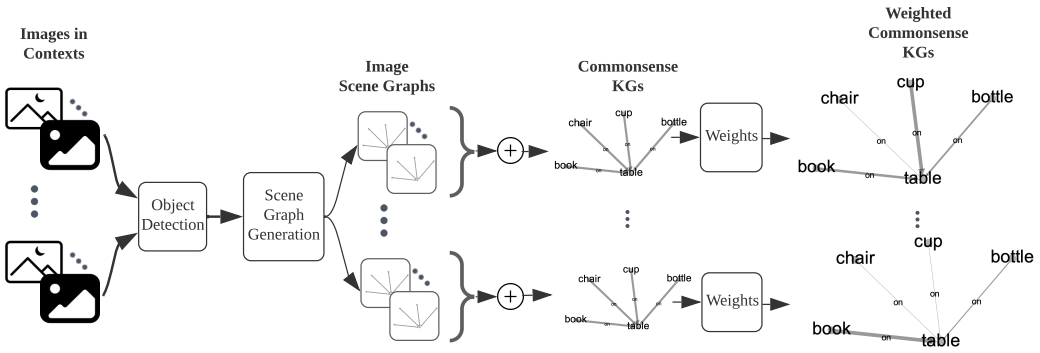


Figure 4.1: The process of generating contextual commonsense KGs from images using deep learning models. Each path focuses on a single context, and the widths of edges (the right-most graphs) represent triple weights.

The previously published method [1] is improved by utilizing state-of-the-art object detection and scene graph generation models in a context-free manner. Furthermore, after minor modifications, we use the same context-free commonsense knowledge generation process to construct contextual commonsense knowledge graphs using only images as input.

4.3.1 Processing of Images

The first stage of the process focuses on the generation of scene graphs. A scene graph is created from a single image – detected objects are represented as nodes, and the relationships between nodes are represented as directed edges. For example, to represent a person sitting on a bench, we could have *Man-2* as the subject, where 2 is the second instance of the object *Man*, *sitting on* as the predicate, and *Bench-1* as the object.

Object and Relation Detection

To detect objects a pre-trained Faster-RCNN [36] model is used. To determine the image features, ResNeXt-101-FPN convolutional neural network (CNN) is applied instead of simpler VGG-16, which results in over 7% mAP (mean average precision) score improvement on the COCO validation dataset [37], [52]–[54]. In the Faster-RCNN algorithm, a model called Region Proposal Network (RPN) uses the CNN-generated image features to predict regions containing

objects. After pooling the features with regions of interest, a classifier predicts object class scores.

Generation of Scene Graphs

In this paper, we focus on the generation of scene graphs from images with no annotations. We apply MOTIFS model [42], which has been relatively unbiased using Causal-TDE method [55]. Compared to several other models in the literature, this combination is promising. Given the predicted bounding boxes, object labels, and the regions-of-interest generated by the Fast-RCNN object detection model, the MOTIFS model uses bidirectional LSTMs (Bi-LSTMs) to predict the object features. Using these features, the MOTIFS model fine-tunes the object classes. Using pairwise object features, the MOTIFS model uses another set of Bi-LSTMs to predict the predicate (relation) labels. The Causal-Total Direct Effect (Causal-TDE) unbiasing method tries to mitigate the long-tail distribution among predicated types, where most predicate predictions comprise a few common types.

4.3.2 Formalizing Commonsense Knowledge Graph Generation

The task of aggregating scene graphs into an overall knowledge base has a few challenges of its own [1]. Some of the main challenges are: 1) quantifying and calculating the importance of scene graph triples; 2) identifying and removing incorrect data; 3) updating the graph with new knowledge; and 4) considering the different meanings of entities in different contexts.

The first task is to detect all viable objects and their relations based on the generated scene graphs. A list of the detected objects is called D_O, and a list of the detected triples is called D_T. Next, we aggregate unique triples, such as *fork-besides-plate*, to construct a KG.

Next, we calculate the weights of triples. To calculate the weights, we investigate three different methods. 1) *Detection Probability-based Method (DPbM).* We assign to each triple a probability of its detection provided by the applied method (Section 4.3.1). Then, we aggregate all the probabilities assigned to

the same triple in the form of a weighted sum. This sum is the final weight of the triple. 2) *Relative Occurrence Method (ROM).* We calculate the number of detected occurrences of each triple in all scene graphs. Then, we divide it by the number of detected occurrences of the triple's object in the same context. 3) *Weighted Occurrence Method (WOM).* This method is similar to the second one. The difference is that we count the 'effect' number of triples and objects by weighting each occurrence by its probability of detection. See Eq. 4.1 for clarification.

Detection Probability-based Method (DPbM):

$$w_{t_i} = \sum_{j=1}^{|D_T|} \delta\left(t_i, t_j\right) \cdot P(t_j) \tag{4.1a}$$

Relative Occurrence Method (ROM):

$$w_{t_i} = \frac{\sum_{j=1}^{|D_T|} \delta\left(t_i, t_j\right)}{\sum_{j=1}^{|D_O|} \delta\left(o_i, o_j\right)} \tag{4.1b}$$

Weighted Occurrence Method (WOM):

$$w_{t_i} = \frac{\sum_{j=1}^{|D_T|} \delta\left(t_i, t_j\right) \cdot P(t_j)}{\sum_{j=1}^{|D_O|} \delta\left(o_i, o_j\right) \cdot P(o_j)} \tag{4.1c}$$

where w_{t_i} is a weight of the t_i triple, $\delta(\cdot)$ is the Kronecker delta function, $P(t_j)$ represents the probability of detecting triple t_j by the scene graph generation model, which is made of a subject (s), predicate (p), and object (o). The weights are normalized by $\max\{w_{t_i} : t_i \in D_T\}$. Please note that all the calculations are done for each context independently as there is one generated knowledge graph per context.

To identify and remove incorrect data, a progressive method is applied. We use the triple weights to determine which triples are more commonsensical and sort the results based on the weights.

For the case of new images, new scene graphs are generated. As a result, the weights of the existing triples are updated, or new weighted triples are added. It means that our approach represents a never-ending learning paradigm, where the knowledge graph is easily updated in the presence of new data.

The generic nature of the method gives the ability to generate commonsense knowledge graphs based on context. The paper focuses on physical contexts,

such as restaurants or parks. We show that the same proposed method can generate contextual commonsense knowledge when applied to different contexts. We further evaluate this claim quantitatively in the results section.

4.4 Generation of Commonsense Knowledge Graphs

Firstly, we construct a commonsense KG with no contexts, which is superior to the one described in [1] in size and quality. Then, we focus on generating commonsense KGs based on different contexts and their human-based evaluation.

4.4.1 Context-Free Commonsense Knowledge Graph

To demonstrate the generation of context-free commonsense KGs, we use the testing section of the Visual Genome (VG) dataset, which has a size of 31,876 images (around 30% of the dataset). To generate scene graphs, we use MOTIFS model unbiased with Causal-TDE method that is able to recognize 150 types of entities, such as *door*, *airplane*, or *horse*, and 50 types of predicates (relations), for example, *holding*, *sitting*, or *above*. The mR@50 (mean recall at 50) is 8.2. The weights of triples are calculated using Eq.4.1.

The resulting context-free commonsense KG generation benefits from upgraded object detection and state-of-the-art scene graph generation compared to the results shown in [1]. The final context-free scene graph generated has 277,634 commonsense triples compared to around 7,000 triples in [1], which is equivalent to around 39 times improvement in size.

4.4.2 Contextual Commonsense Knowledge Graph

To generate a contextual commonsense knowledge graph, we start with 'raw' images from the Visual Commonsense Reasoning (VCR) dataset [56]. This dataset has around 110,000 images. The images are human-annotated regarding the places they represent. After a cleaning process and removing places with less than 100 images, we end up with 93 different locations. We consider these places as physical contexts, such as a restaurant or classroom.

As explained above (Section 4.3), we process images from each context to detect objects and then to predict relationships between the objects detected. The threshold of keeping the predicted relations is chosen as 0.1 and 0.6 for DPbM and ROM/WOM, respectively. It has ensured obtaining higher confidence in the identified relations. Given the object and relationship detected, we generate a single scene graph for each image. The scene graphs constructed from images belonging to a single context are aggregated to a single KG. The triple weights are assigned using the three different methods explained in Eq. 4.1. The sizes of the generated knowledge graphs varies depending on the context, but all are less than 1,000 triples, as we have tried to

retain only high-quality results.

An example of a contextual commonsense knowledge graph representing a fragment of a graph that combines multiple graphs built for single contexts is illustrated in Fig. 4.2. It shows the object *plate* connected to tables at different locations. The thickness represents the strength/weight of the relation.

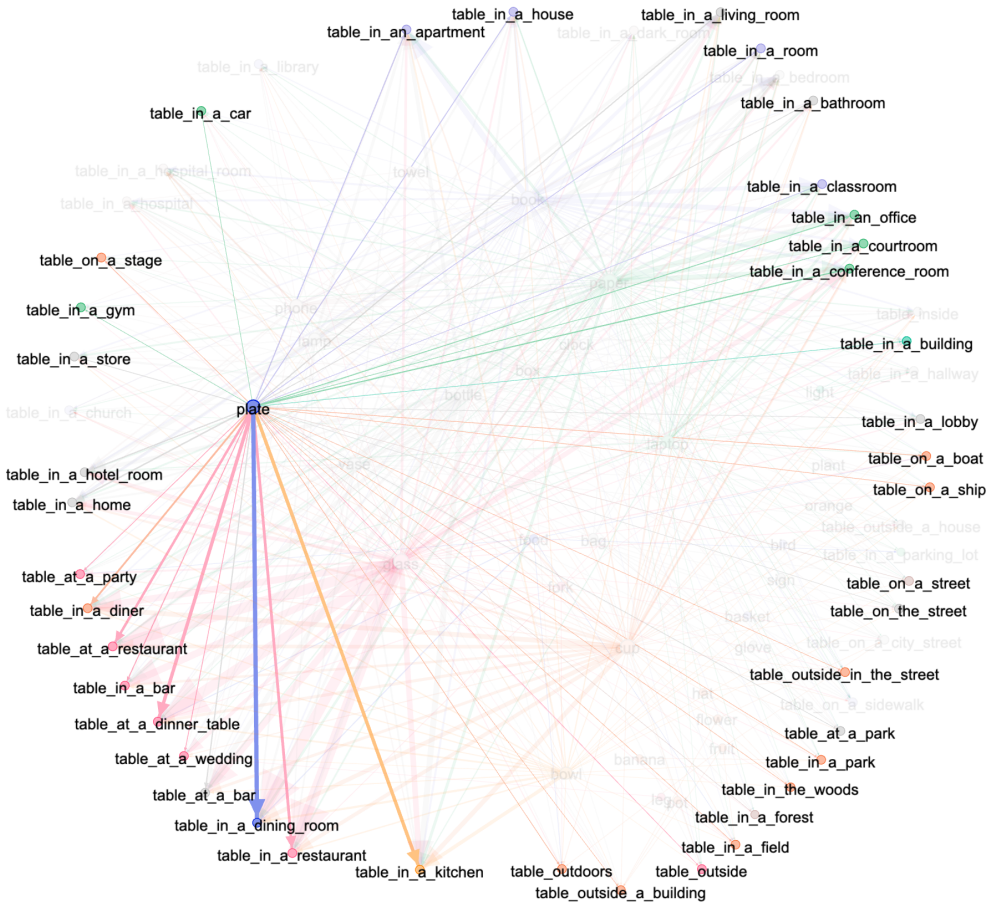


Figure 4.2: Snippet of a contextual KG illustrating the item *plate* and its relation to tables at different locations; thickness of lines indicate a strength of the connection.

To evaluate how weight assignment mechanisms correlate with human commonsense judgment, we chose the top 100 triples based on each weighting mechanism and gave them to three humans to evaluate. The three annotators have rated each commonsense triple as: "quite commonsensical", "commonsensical", "making sense, but not in this context", and "invalid/unfamiliar in any context". The first two options are considered as accepted. To illustrate the third option, let us take a triple *car having tire* – it makes general sense, but is it not commonsensical, for example, in the restaurant context. A triple *person wearing laptop* makes little sense in any context and results in the fourth option.

Our three volunteer reviewers have evaluated the triples from two contexts: *restaurant* and *classroom*. To measure the agreement among reviewers, we use Krippendorff's Alpha [57], which is a value between 0 and 1. Zero means no agreement and one means complete agreement. As seen in Table 4.1, the *Detection Probability-based Method (DPbM)* gives the highest correlation with human commonsense, while other methods still show good results.

Weighing Schema	Accept	Reject	N/A	Accuracy (%)	Alpha
DPbM	560	22	18	**93.0**	**0.78**
ROM	526	60	14	87.6	0.63
WOM	538	51	11	89.7	0.72

Table 4.1: Human evaluation of the three weighting strategies defined in Eq. 4.1. Three reviewers were given top 100 triples from each restaurant and classroom contextual commonsense knowledge graphs (total of 600 evaluations per method).

4.5 Reasoning with Contextual Commonsense Knowledge Graph

The contextual commonsense KG allows for performing reasoning tasks. To illustrate a methodology that can utilize a contextual commonsense graph for possibility theory-based reasoning, we incorporate an example with *tables* positioned at different locations and *items* placed on them.

The reasoning is based on the commonsense knowledge embedded in the auto-generated contextual knowledge graphs introduced above. The weights linked with triples are converted into occurrences of items in contexts that are considered. These are used to calculate required possibilities.

4.5.1 Extraction of Relevant Nodes and Occurrences

In the beginning, we extract – from the constructed contextual graph –the information about *tables* at three different places and items that are *on* the tables, i.e., are in the relation *on* with the tables. A snippet of the graph illustrating the information is shown in Figure 4.3. As we can see, there are several items – *paper*, *bottle*, *book*, *laptop* and *cup* – to be considered, all of which appear on *tables* at the three locations. Although the items *glass* and *plate* are 'seen' at all locations, we do not use them: a) due to ambiguity of *glass* – it can be treated as a 'cup' or as 'lenses'; and b) due to a very small number of occurrences of *plate* at tables in classrooms.

The snippet, Figure 4.3, emphasizes one item – *cup* – that has been found at all three locations to a different degree. We see that it 'showed' up 696 times on the restaurant images, but only 98 times on a table in the restaurant. Similarly, there are occurrences of a cup in classroom and bar images.

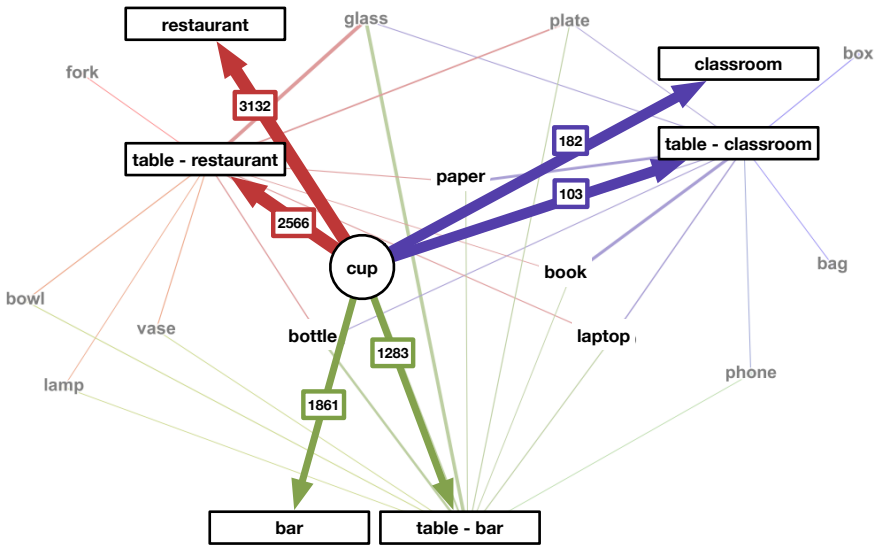


Figure 4.3: Subgraph of items *on* tables in restaurant, classroom, and bar. The occurrences of *cup* are emphasized: the number of times on table versus the total number of times.

The specific number of occurrences of individual items on *tables* at three locations are included in Table 4.2. It contains the numbers of occurrences of a given item on a *table* and the total number of occurrences of the items 'seen' at all considered images of specific locations. Additionally, the table shows the number of times a *table* appeared in pictures at each location. The numbers of pictures we have processed are 1513 of a bar, 692 of a classroom, and 2235 of a restaurant.

4.5.2 Reasoning: Inference with Uncertain Premises

In the presented reasoning process, we adopt a frequentist setting where possibility measures can be induced from the frequency of observations [58], and the extracted subgraph is a possibilistic graph [26]. We use a deductive inference with uncertain premises. Following analysis of *Modus Ponens* and *Modus Tollens* [59] [60], we apply the following reasoning schema:

$$\Pi(\ q \mid p\ ) \geq a \tag{4.2}$$

$$\Pi(\ p \mid q\ ) \geq a' \tag{4.3}$$

$$\Pi(p) \in [\ b,\ b'\ ] \tag{4.4}$$

with the consequence:

$$\Pi(q) \in \left[a * b, a' * \rightarrow b' \right] \tag{4.5}$$

location:	bar		classroom		restaurant	
item:	on table	total	on table	total	on table	total
paper	28	79	124	299	69	222
bottle	125	1562	4	52	108	634
book	20	37	164	362	47	72
laptop	4	17	47	67	37	56
cup	50	287	5	36	98	696
table	1170		837		2603	

Table 4.2: *on table*: Number of occurrences of items on table; *total*: Total number of items; Last row: Number of tables. The contexts in this case are bar, classroom, and restaurant. The observations are made in these contexts.

where $*$ is a triangular norm (t-norm), and

$$a' * \rightarrow b' \quad \text{means} \quad sup\{t \in [0,1], a' * t \leq b'\}.$$

Different selection of $*$ leads to different deduction schemas. For min, Eq. 5 becomes:

$$\Pi(q) \in \left[min(a,b), \begin{cases} 1 & if\ a' \leq\ b' \\ b' & if\ a' >\ b' \end{cases} \right] \tag{4.6}$$

and for *product*:

$$\Pi(q) \in \left[a \cdot b, \begin{cases} 1 & if\ a' = 0 \\ min(1, \frac{b'}{a'}) & if\ a' \neq\ 0 \end{cases} \right] \tag{4.7}$$

For calculating a possibility measure of an individual premise, we use:

$$\Pi(p) = \frac{number\ of\ observations\ of\ p}{number\ of\ observations}$$

and for complex premises:

$$\Pi(p_i \wedge p_j) = min(\Pi(p_i), \Pi(p_j))$$

In the presented example, the goal is to deduce a location of tables based on several different items seen on tables. Additionally, we consider two levels of the possibility of seeing these items – we are very confident that the items are on tables, and we are very uncertain about it.

Three different sets of items on *tables* are considered. We define the following premises:

$$p_A = (paper \wedge bottle \wedge book \wedge laptop \wedge cup)$$
$$p_B = (paper \wedge book \wedge laptop)$$
$$p_C = (bottle \wedge cup)$$

For the premises p_A and $q = table\_in\_X$, we have:

location X:		bar	classroom	restaurant
premise:				
p_A	*min*	[0.2353, 1]	[0.1111, 1]	**[0.6607, 1]**
	product	[0.2353, 1]	[0.1111, 1]	**[0.6607, 1]**
p_B	*min*	[0.2353, 1]	**[0.7015, 1]**	[0.6607, 1]
	product	[0.2353, 1]	**[0.7015, 1]**	[0.6607, 1]
p_C	*min*	**[0.1742, 1]**	[0.1111, 1]	[0.1546, 1]
	product	**[0.1742, 1]**	[0.1111, 1]	[0.1546, 1]

Table 4.3: $\Pi(table\_in\_X)$ for different locations and premises; possibility of premises is 1, i.e. $\Pi(p_A) = \Pi(p_B) = \Pi(p_C) = 1$.

$$\Pi(table\_in\_X|p_A) = \frac{|table \wedge paper \wedge bottle \wedge book \wedge laptop \wedge cup|}{|paper \wedge bottle \wedge book \wedge laptop \wedge cup|} \tag{4.8}$$

The numerator is a number of observations where all items of p_A and *table* occur at the same time. The denominator, on the other hand, is a number of observations where all items of p_A occur. We also have:

$$\Pi(p_A|table\_in\_X) = \frac{|table \wedge paper \wedge bottle \wedge book \wedge laptop \wedge cup|}{|table\_occurrences\_in\_X|} \tag{4.9}$$

As mentioned above, two possibility measures are used when taking into account items on tables, i.e., different values of b, b' for $\Pi(p)$, Eq. 4.4. With that, we determine $\Pi(table\_in\_X)$ for three different locations $X = bar$, $X = classroom$, and $X = restaurant$, and we use two t-norms, Eqs. 6 and 7.

The values obtained for $b = b' = 1$, i.e., $\Pi(p_{A|B|C}) = 1$, for both types of t-norm are shown in Table 4.3. As we can see, different sets of items on a table lead to different values of the lower bounds of possibility intervals. The values confirm commonsense reasoning regarding occurrence of given items on tables at specific locations. It means that the items of the premise p_A lead to the conclusion that the table is in a restaurant, while for the premise p_C points to a bar.

If we change values of b and b' to 0.1, it can be observed that low possibilities of $p_{A|B|C}$ fully 'control' possibilities of locations, Table 4.4. It becomes evident that conditional parts of the deduction schema are suppressed, seen especially when a t-norm is *min*. On the other hand, more interesting values can be observed when a t-norm is *product*. In that case, the values of the lower bounds of possibility intervals provide more meaningful – commonsense – findings.

4.6 Conclusion

Commonsense knowledge graphs are shown to benefit many AI technologies, such as transformer-based language models. Hence, their construction is gaining much attention.

location X:		bar	classroom	restaurant
premise:				
p_A	*min*	[0.1000, 1]	[0.1000, 1]	[0.1000, 1]
	product	[0.0235, 1]	[0.0111, 1]	[**0.0661**, 1]
p_B	*min*	[0.1000, 1]	[0.1000, 1]	[0.1000, 1]
	product	[0.0235, 1]	[**0.0701**, 1]	[0.0661, 1]
p_C	*min*	[0.1000, 1]	[0.1000, 1]	[0.1000, 1]
	product	[**0.0174**, 1]	[0.0111, 1]	[0.0155, 1]

Table 4.4: $\Pi(table\_in\_X)$ for different locations and premises; possibility of premises is 0.1, i.e. $\Pi(p_A) = \Pi(p_B) = \Pi(p_C) = 0.1$.

This paper presents a methodology for automatically building contextual weighted commonsense knowledge graphs based on the processing of images using deep learning models. Three methods are introduced to assign commonsense weights to the knowledge graph triples. The human evaluation confirms a high correlation between the generated contextual commonsense knowledge and human commonsense in the same context. Moreover, we show how the generated commonsense knowledge can be used to perform possibilistic reasoning.

Chapter 5

Utilizing Language Models to Expand Vision-Based Commonsense Knowledge Graphs

5.1 Introduction

There has been a renewed interest in commonsense as a stepping stone toward achieving human-level intelligence. Some of the new research has shown how important commonsense knowledge graphs can be in training artificial intelligence (AI) models, which exhibit commonsense [48], [61].

Commonsensical concepts should be symmetric to any changes in their representation. In the case of an ideal commonsense knowledge graph and an ideal language model, transforming concepts between the two representations of knowledge should not change their meaning. By an ideal language model, we mean a language model that is sufficiently large and capable that can understand language and all the concepts within. At the same time, an ideal commonsense knowledge graph is a knowledge graph that contains all correct commonsensical concepts.

The knowledge-symmetric transformation depends on the architecture of the language model and the knowledge graph, both of which are not ideal. These issues make deriving a transformation process that symmetrically maps knowledge from one to the other challenging and impractical. To compensate for this, we introduce a prompting methodology based on questions and answers to extract from the language model the knowledge missing in the knowl-

edge graph. In that way, the symmetry of concepts is preserved by mapping them between two knowledge storage paradigms.

The main building blocks of knowledge graphs used to represent commonsense knowledge are subjects, predicates, and objects. Subjects and objects are other words for the nodes of the graph. The tail of a relationship is called an object, and the head is called a subject. The directed edge connecting the two is called a predicate. Knowledge graphs are directed heterogeneous graphs in some sense.

Artificial intelligence (AI) models are reported to have limited commonsense abilities [62], [63]. Acquiring commonsense by AI systems can make the sample efficient in adapting to new environments, as proposed in [64]. Commonsense knowledge graphs can help AI systems both explicitly and implicitly: explicitly by querying the commonsense knowledge graph itself, or implicitly by knowledge transfer methods, such as the fine-tuning of language models as reported in [48]. This is similar to how a BERT model is fine-tuned on a SQuAD dataset for reading comprehension [65]. In addition, expressing commonsense knowledge in the symbolic format can help with commonsense knowledge explainability and the vetting process.

Using only vision to generate commonsense knowledge as proposed in [2], [66] has its advantages and disadvantages. By processing images and videos, we can perceive visual cues that are not usually written or spoken about, but they make our common understanding of how physical entities exist and interact. On the other hand, fine-tuned vision-based deep learning models are limited to the concept and relation vocabulary that they are trained on, which is usually limited, and are not usually capable of understanding the intricacies of natural language. An ideal self-supervised vision model, which can absorb and learn all the visual interactions, could theoretically suffice. However, the current vision models have shortcomings that we believe can be addressed via the utilization of language models. For example, scene graph generation models are limited regarding the number of detected relationship types. Increasing the number of relationship types does not provide a satisfying solution, as there is still a bias toward the frequently seen relationships in the supervised training [55].

In this paper, we explore and use the extra knowledge that language models offer to expand on the limited auto-generated vision-based commonsense knowledge graphs. We chose to use few-shot learning in larger transformer-architecture-based language models, as larger models have shown to perform well on language benchmarks without requiring further fine-tuning on a specific task. We experiment with not only adding new concepts to the vision-based commonsense knowledge graph but also new types of relationships with fuzzy-style linguistic weights.

5.1.1 Commonsense Definition

Having a good definition of commonsense is imperative to better understand and discuss the work and results. Commonsense is simple, as almost everyone knows it, and is challenging, as no one often talks or writes about it.

Yann LeCun, an inventor of convolutional neural networks, believes that a collection of models of the world that represents what is likely, plausible, or impossible makes our commonsense [62]. John McCarthy classifies human commonsense into two categories of knowledge and ability. The commonsense ability is the action based on the gained commonsense knowledge [8].

Commonsense knowledge is inherently uncertain and context-dependent. The degree of correctness of commonsense knowledge depends on the common group of observers. For instance, the people who live in the northern hemisphere know July to be a hot summer month, while the people in the southern hemisphere observe it as a colder winter month.

Commonsense can also be classified into different topics, such as physical interactions, order of events, and social dynamics. In this paper, we mainly focus on physical commonsense, such as the usage of an object and its relative location, compared to other objects.

In a nutshell, commonsense knowledge graphs are graphs that represent facts and relations between them that are characteristic of real-world scenarios and situations. Such graphs focus on elements and aspects related to everyday activities, arrangements of things, and normal/natural circumstances, such as *flower in vase*, *tree has trunk*, *food on plate*, *shoe is less likely made of metal*,

or *arm is most likely to be able to move, bend and be strong.*

Such facts seem very obvious and normal/natural for a human being, but this knowledge is not easy to be acquired by a machine. The gap in the processes of learning that type of information is filled out by techniques and methods linked to collecting and representing commonsense knowledge.

5.1.2 Contributions

The goal is to utilize transformer-based language models to expand vision-based commonsense knowledge graphs. In this paper, we propose an extension of the methodology for constructing a commonsense knowledge graph proposed in [2], [66] with a technique based on questions and answers prompting very large language models. The new technique addresses generating prompts that are used as inputs to the language models. First, a prompt is entered into the model. Then, the obtained response that contains facts/information is added to expand a knowledge graph. The method is illustrated in Figure 5.1.

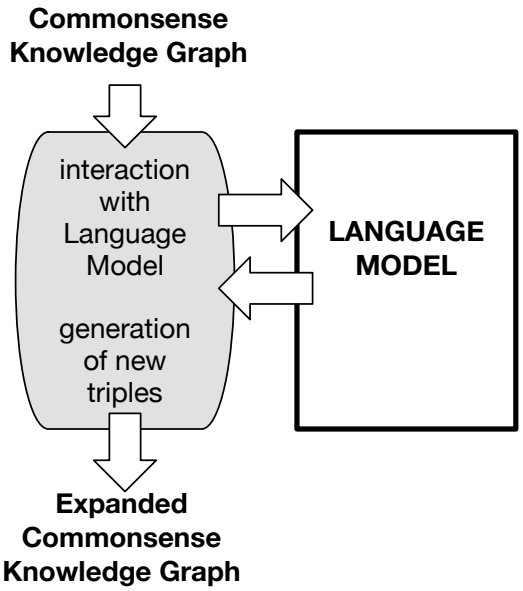


Figure 5.1: Expansion of a vision-based commonsense knowledge graph with relevant but new information.

In particular, the contributions of the paper are as follows:

- A multi-modal methodology for constructing commonsense knowledge

graphs;

- A process of generating question/answer-based prompts for language models based on triples extracted from an existing commonsense knowledge graph, or based on the input from users;
- An expansion of the standard structure of knowledge graphs by introducing an approach to add degrees of likeliness as indicators of the 'strength' of triples that are added to commonsense knowledge graphs; the degrees are expressed with linguistic terms , such as **more likely**, **less likely**;
- An evaluation process based on Amazon Mechanical Turk.

5.2 Related Work

The work presented in this paper falls into the category of tasks that focus on completion and expansion of a commonsense knowledge base. There is related literature that addresses the methods and tools to achieve these goals.

5.2.1 Expansion of Knowledge Bases

The work presented in [67], [68] focuses on link prediction between known entities within a graph. These methods are not able to expand beyond the current conceptual knowledge in the graph. They are more suited toward finding possible relations between currently known concepts.

Recent works have tried to use language models, especially transformers [12], to achieve better results in the tasks of the completion and expansion of the knowledge base. The authors of [69] use language models to construct knowledge graphs: they assume to have a subject and object and then use the language model to predict an appropriate relationship between them.

Ref. [70] indicates that using the next token prediction capability of pre-trained language models, one can use them as a factual knowledge base, e.g., to find the birthplace of a specific person. Among the language models analyzed, the largest transformer-based language model, BERT-Large [71], performed better than others. This paper confirms the overall consensus in the research

community that the larger the language models become, the more capable they become.

There has been recent works to train very large generic transformer-based language models, such as GPT-3 by OpenAI [72], Meta's OPT-175B [73], and Google's PaLM [11]. There is a common consensus among all recent findings that larger language models can potentially be more capable of performing diverse tasks. Additionally, they do not need costly fine-tuning and data collection. Yet, providing appropriate prompts to language models can be challenging.

Processes of generating prompts are a subject of recent research publications. Prompts serve as input to large language models [74] and are used to reduce the amount of data required for fine-tuning [4]. By prompt, we mean a set of tokens and a short text that constitute the input to the model. Prompts could have different purposes, such as providing context, tone, or a sample of expected responses. They are part of the few-shot-learning process and are usually used instead of fine-tuning a language model. While prompts benefit the overall performance, their design does not follow a specific rule. Some even call the process 'prompt engineering.' Question and answering tasks are improved by few-example prompts when using large language models [72]. Extra chain-of-thought language prompts that contain reasoning steps are shown to improve more complex tasks related to arithmetics and commonsense [74]. The chain-of-thought process helps find missing parts of knowledge [75]. The work is similar to unsupervised data creation [76]. However, questions and answers used in this paper serve as prompts to foundation models. They are not used directly on text for reading comprehension.

5.2.2 Construction and Expansion of Commonsense Knowledge

A body of literature [48], [61] focuses on annotating commonsense knowledge graphs to train language models for predicting commonsense information based on the given subject and predicate. The human-annotated knowledge graphs are typically in the size of millions and cover social interactions, events, and

entity commonsense. The ATOMIC-COMET work [48] is based on manually creating a commonsense knowledge graph. This graph is used to train a small language model, such as GPT-2, on the human-annotated data. Our approach is different. We focus on generating a commonsense knowledge graph automatically rather than manually. The method comprises two phases, the first based on vision and the second enriching the results using language. The manual generation of commonsense knowledge graphs can become costly, as shown in [4]. Our approach seems to be more similar to [61], where GPT-3 is utilized to generate commonsense knowledge graphs. The proposed method is different in multiple ways. One is that we use a two-step method, where the feed for GPT-3 is provided by visual data, while [61] uses human-annotated data. Moreover, [61] only generates the most probable results, while we generate both highly probable and less probable results. Our approach has a cost of roughly one-fifth of the method described in [61] when considering the linguistic generation part and using the same GPT-3 model size. The reduced cost is because our prompt method accommodates the generation of $N = 5$ triples in one pass. Another difference is that [61] is proposed to only find an object given the subject and predicate, while our approach works in both ways and can suggest an appropriate subject given the predicate and object.

Several papers have experimented with the new very large transformers, such as GPT-3 [61]. This work focuses on prompting GPT-3 with some annotated commonsense triples, then extracting GPT-3's commonsense and adding it to a graph. It assumes pre-defined predicates and does not explicitly discuss the weight of the triples. The process takes subjects and predicates and uses causal language models to predict the most suitable objects. The method we introduce in this paper can also predict the triples' subjects.

It was discussed in [77], [78] that training a model on commonsense knowledge base completion (CKBC) task suffers from low-coverage training data. Therefore, training on specific data results in the model's over-fitting and reduces its performance on novel data. Based on these observations, we focused our efforts on generically trained language models, which are large enough to accommodate few-shot learning.

A few recent works report on generating knowledge graphs from visual data. As an example, the NEIL method [34] extracts object relationships in images and results in 10,000 triples using 10 types of predicates.

One of the main physical commonsense knowledge graphs is ConceptNet [9]. As much as it can be helpful and treated as a reference, it has some drawbacks that our work can potentially resolve in the future. First, ConceptNet is mainly human annotated and cannot be continuously and cost-effectively updated. Our work suggests a methodology to continuously and automatically update the missing commonsense knowledge. Second, ConceptNet has a limited predicate related to location–the vague *AtLocation* predicate. Our method is able to enrich the commonsense knowledge with more fine-grained relations, such as *Above*, *Below*, and others. Third, ConceptNet is limited in terms of its predicate types, too. Our approach can enrich ConceptNet with new types of predicates, such as *NotIsA* or *CanEat*. An essential weakness of ConceptNet is its lack of context. For example, finding a desk in a classroom is more probable than in a bar. Our approach can potentially expand and enrich ConceptNet with weighted contextual relations. Moreover, it can be done automatically if part of ConceptNet is used as a seed commonsense knowledge graph.

5.3 Image-Based Construction of Commonsense Knowledge Graph

In our previous works, we introduced methodologies to generate a commonsense knowledge graph, called *world-perceiving knowledge graph* (*WpKG*), by only using visual data [2], [66]. Like human infants who gain commonsense details about their physical world before they learn to express them in language, the introduced process focuses on deducing commonsense knowledge by observing many images.

The *WpKG* paper [66] introduces a methodology to auto-generate commonsense using deep learning models to perform object detection and relation prediction. The final *WpKG* graph has 7000 triples using 50 predicate and

150 entity types. [2] expands on the previous work to generate contextual and weighted commonsense knowledge graph, *C-WpKG*, in 93 contexts using state-of-the-art object and relation detection models. In the following sub-sections, we describe the process of reaching these results.

5.3.1 Extraction of Scene Graphs

The first step in the process is to analyze each image individually by detecting the existing objects and extracting possible relations between the objects in the image. The resulting graph representing objects in images as nodes and their relationships as edges is called a scene graph.

A convolutional neural network (CNN) model, such as Faster-RCNN [36], is used to detect the objects. To produce image features, ResNeXt-101-FPN CNN model [79] is utilized, which is needed for the region proposal network (RPN) of the Faster-RCNN model. The output of the pre-trained object detection model includes objects in the image, together with their bounding boxes and class scores.

To predict relations between the objects and generate a scene graph for each image, the MOTIFS model [42] unbiased by the Causal-TDE method [55] is used. Then, the scene graph for each image is generated based on the object features and relations between them.

5.3.2 Fusion of Scene Graphs

Regularly observed phenomena make up collective commonsense knowledge. Similarly, we aggregate the scene graphs extracted from the images into a single knowledge graph that comprises possible commonsense relations. To differentiate between relationships to know if a phenomenon is a one-time event or a typical one, we assign weights to the links representing the relations. Different methods of assigning weights to the observations are investigated. Among them, a probability-based approach is selected. It correlates the most with human commonsense during human evaluations. This weight assignment method follows Equation (5.1).

$$w_{t_i} = \sum_{j=1}^{|D_T|} \delta\left(t_i, t_j\right) \cdot P(t_j) \tag{5.1}$$

where w_{t_i} is a weight of the t_i triple, $\delta(\cdot)$ is Kronecker delta function, $P(t_j)$ represents the probability of detecting each instance of triple t_j, which is made of a subject (s), predicate (p), and object (o). The weights are also normalized by $\max\{w_{t_i} : t_i \in D_T\}$. The list of all detected triples is represented by D_T.

Variations of the same method have been shown to work in context-free and contextual scenarios. In this paper, we only focus on context-free visual commonsense knowledge.

5.4 Expanding Knowledge Graph Using Language Model

The automatic construction of commonsense knowledge graphs requires retrieving commonsense knowledge. It seems natural—also for a human being—to start that process by analyzing images and pictures representing real-world situations. Yet, to further increase commonsense knowledge and expand knowledge graphs, other sources of information are required and beneficial. One of them is verbal, textual information.

Therefore, to diversify information embedded in vision-based commonsense knowledge graphs and further expand them, we propose a human-like method of assimilating commonsense knowledge using linguistic-based data sources.

5.4.1 Methodology

The proposed method is intuitive and straightforward. It starts with interaction with a language model using short texts created based on the commonsense knowledge graph to be expanded. Then, the obtained results, i.e., the retrieved pieces of information and facts, are added to the graph as triples. The overview of the process is illustrated in Figure 5.2. It shows the *WpKG* as a graph from which some triples are extracted. The information from these triples is used to instantiate prompt templates (Section 5.4.3) that represent

training data for a language model. The instantiated prompts are entered into the model. As a result, the obtained pieces of information are converted into new triples. These new triples are added to the *WpKG*, leading to its expansion.

5.4.2 Language Models

Larger language models, such as GPT-3, have shown promising results on diverse benchmarks with only a few examples of each task. The results are sometimes even comparable with smaller language models, which are fine-tuned on a large corpus of data. Recent research has shown the usefulness and effectiveness of large language models in automatic commonsense knowledge generation [61]. In this paper, we utilize different versions of a large language model, called GPT-3 [72], to expand vision-based commonsense knowledge graphs.

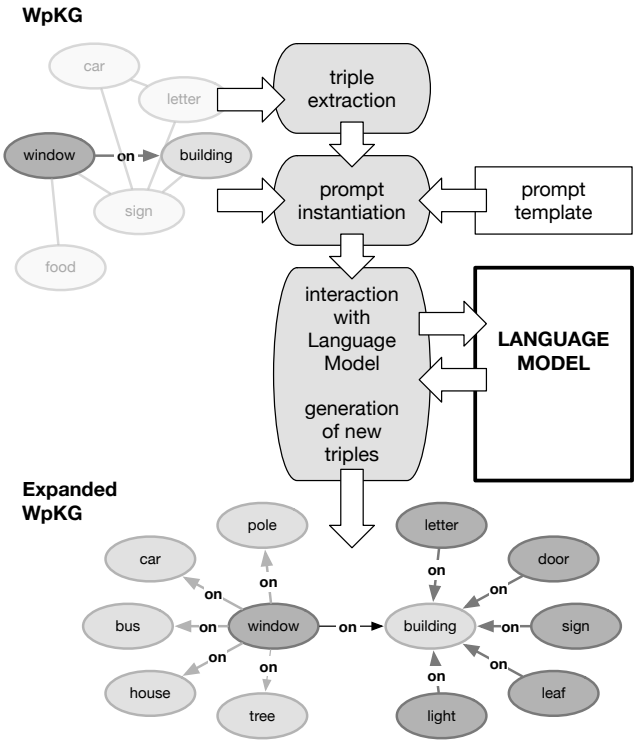


Figure 5.2: Process of expanding a graph using language model.

GPT-3 is a causal language model with almost the same yet larger architecture as previous iterations of the same model (GPT and GPT-2). The goal

of a causal model is to predict the next token given the previous tokens. The language model assigns a probability to all the tokens to decide which one could happen next.

Choosing the highest probability next token may not be the best option, given the task. In this paper, we use nucleus (top-p) sampling to generate the text responses [80] and also adjust the temperature of the sampling to reach better results.

By reducing the temperature, we basically increase the likelihood of high-probability next tokens and reduce the likelihood of low-probability next tokens. This setting results in more deterministic next tokens to be chosen when selecting the next token randomly. The temperature is implemented as a coefficient inside the softmax function. Empirically, we observed that lower temperature works better for simpler cases, while the higher temperature can work for more complex cases that need diverse results, e.g., finding objects that are less likely to exist given a subject and a predicate.

In nucleus (top-p) sampling, instead of sampling from all the tokens, the algorithm chooses from the set of tokens that their cumulative probability of occurrence next is smaller than a given probability p. In our experiments, we keep the p value equal to one to choose from the most diverse vocabulary possible.

5.4.3 Language Model Prompts

Retrieving information from GPT-3 involves prompting the model with a few examples that serve as a few-shot learning training data. The content and the structure of the responses depend on these prompts. Therefore, experimentation with different prompts to achieve the desired structure is necessary. Formally, the examples that define the structure and content of an interaction with a language model are called *prompts*.

The purpose of a prompt is to 'show' the model how to interpret and respond to an input text appended to the prompt, which in our case is a question. For example, one wants to retrieve a piece of information about the most common items found on a table in a conference room. In such a case, the following

prompt is constructed and used:

---

prompt:	*Q: What can be found on table in bar? Name five.* *A: bottle, class, cup, napkin, fork.* *Q: What can be found on table in conference room? Name five.*
GPT-3 response:	**paper, glass, laptop, phone, box.**

This example is a simple explanation of the role of the prompt. As it can be seen, the first part of the prompt—Q and A—is one-shot training data and 'teaches' the model that for a type of question like Q, a proper response looks like A. After that, the 'real' question *Q: What can be found on table in conference room?* is asked. Then, finally, the model responds with five items it 'thinks' represents the most suitable response.

Sometimes, one example is not enough, and multiple examples need to be provided to serve as few-shot learning training data. Empirically, we find that explaining the task and a well-defined question format help the model respond better.

To achieve more accurate results, we also utilize the chain-of-thought prompting method introduced in [74] for the fuzzy and the predicate expansion cases, described in Sections 5.5.2 and 5.5.3, respectively. In each example answer in the prompt, we hand-craft a reasoning that can help narrow down to the correct response. The model learns to generate a similar pattern and, as a result, generates a reasoning before answering the asked question.

5.5 Expansion of Commonsense Graph

To illustrate the benefits of using a language model for expanding the *WpKG*, we extract information from GPT-3 to construct different triples. It shows how versatile the interaction with the model can be and how different results are obtained. The presented utilization of GPT-3 involves the following scenarios:

- Asking for subjects and objects for given relations using a basic prompt

template;

- Asking for the most and least likely subjects and objects for given relations to construct fuzzy triples;
- Asking for the most and least likely objects with novel relations given by a user.

Expanding the existing graph means 'asking' the language model to provide answers that contain the most suitable pieces of information that are directly added to the graph as nodes—subjects and objects—and relations that link the existing nodes to the newly added ones.

The questions are prepared based on templates that are initialized with facts/information obtained from the *WpKG* or from a user. Three sets of templates are constructed, one for each type of defined-above scenarios.

5.5.1 Simple Triples

In the beginning, a straightforward scenario that involves adding simple triples, i.e., triples that are not associated with degrees of strength of relations between subjects and objects, is presented. In such a case, GPT-3 is asked questions that result in retrieving from the model facts that are interpreted as subjects or as objects. It means that the questions are of the format $\langle ?s,\ relation_X,\ object_X \rangle$ when subjects are asked for, or $\langle subject_X,\ relation_X,\ ?o \rangle$ when objects are asked for. The retrieved subjects and objects are added as triples with the $relation_X$ to the *WpKG*.

In a nutshell, the process—for a single $relation_X$—is as follows:

- Extract five triples with the $relation_X$ from the *WpKG*.
- Select randomly one triple from the set of five, say, triple k; it is used in the process
  of customization of a prompt template for the $relation_X$.
  - Extract a set of five most popular objects Obj_k fitting $\langle subject_k,\ relation_X,\ - \rangle$
    from the *WpKG*.

- Extract a set of five most popular subjects Sub_k fitting $\langle$ -, $relation_X$, $object_k\rangle$
  from the *WpKG*.
- Audit the instantiated prompt and make changes if necessary.

- For each extracted triple $\langle subject_i, relation_X, object_i\rangle$:
  - Put $subject_i$ and $relation_X$ into the question template and append to the prompt.
  - Put the prompt to the language model to initiate the text generation.
  - Extract the five new objects Obj_{LM} from the generated text.
  - Add five new triples $\langle subject_i, relation_X, \text{-}\rangle$ with objects from Obj_{LM} to *WpKG*.
  - Put $relation_X$ and $object_i$ into the question template and append to the prompt.
  - Put the prompt to the language model to initiate the text generation.
  - Extract the five new subjects Sub_{LM} from the generated text.
  - Add five new triples $\langle$-, $relation_i$, $object_X\rangle$ with subjects from Sub_{LM} to *WpKG*.

As it is described above, the process of asking GPT-3 involves the instantiation of prompt templates. For the simple triples case, the prompt templates for asking for both *objects* and *subjects* are shown in Table 5.1. Following the aforementioned process, it can be seen that the prompts are filled out with facts/information obtained originally from *WpKG*, and the same initialization is used for prompting GPT-3 for all other *objects* or *subjects* obtained from the randomly selected $relation_X$s. Depending on the predicate $relation_X$, different variations of the prompt templates are created to result in meaningful questions and answers.

Table 5.1: Sample template for simple triple.

	SIMPLE_TEMPLATE_A for ⟨subject⟩
prompt:	*Answer with five items separated with comma.* *Q: What is ⟨relation$_X$⟩ ⟨object$_k$⟩? Name five.* *A: elements of Sub$_k$* *Q: What is ⟨relation$_X$⟩ ⟨object$_i$⟩? Name five.*

	SIMPLE_TEMPLATE_B for ⟨object⟩
prompt:	*Answer with five items separated with comma.* *Q: What ⟨subject$_k$⟩ can be ⟨relation$_X$⟩? Name five.* *A: elements of Obj$_k$* *Q: What ⟨subject$_i$⟩ can be ⟨relation$_X$⟩? Name five.*

The templates from Table 5.1 are used with five different relations: *behind, in, has, on*, and *watching*. The instantiated prompt templates, together with the results of querying GPT-3 for the relation *on*, are shown in Table 5.2 for extracting *subjects*, and in Table 5.3 for extracting *objects*.

It can be seen that, for example, selecting $object_A = plate$, we obtain the following triples: ⟨*food, on, plate*⟩, ⟨*drink, on, plate*⟩, ⟨*utensils, on, plate*⟩, ⟨*napkin, on, plate*⟩, and ⟨*tablecloth, on, plate*⟩, Table 5.2. Similarly selecting $subject_A = hair$, we obtain the triples such as ⟨*hair, on, head*⟩, ⟨*hair, on, beard*⟩, ⟨*hair, on, eyebrows*⟩, ⟨*hair, on, eyelashes*⟩, and ⟨*hair, on, pubic*⟩ (Table 5.3). Another example, this time in a graphical form, that shows an expansion of the triple ⟨*window, on, building*⟩ is illustrated in Figure 5.3. Besides the original triple, the figure includes its extension on both *subject* and *object* sides.

Table 5.2: Query and results for ⟨-, *on*, -⟩ for *subject*.

user:	*Answer with five items separated with comma.* *Q: What is on building? Name five.* *A: letter, door, sign, leaf, light.* *Q: What is* $\langle relation_A \rangle$ $\langle object_A \rangle$*? Name five.*
where:	$relation_A$ = *on* $object_A$ = *{ building, sign, man, plate, head }*
GPT-3:	
building:	$subject_A \in$ *{letter, door, sign, leaf, light}*
sign:	$subject_A \in$ *{words, letters, numbers, shapes, colors}*
man:	$subject_A \in$ *{shirt, pants, belt, shoes, socks}*
plate:	$subject_A \in$ *{food, drink, utensils, napkin, tablecloth}*
head:	$subject_A \in$ *{hair, hat, ear, eyebrow, eyelash }*

Table 5.3: Query and results for ⟨-, *on*, -⟩ for *object*.

user:	*Answer with five items separated with comma.* *Q: What window can be on? Name five.* *A: pole, car, bus, house, tree.* *Q: What* $\langle subject_B \rangle$ *can be* $\langle relation_B \rangle$*? Name five.*
where:	$subject_B$ = *{ window, letter, hat, food, hair }* $relation_B$ = *on*
GPT-3:	
window:	$object_B \in$ *{pole, car, bus, house, tree}*
letter:	$object_B \in$ *{A, B, C, D, E }*
hat:	$object_B \in$ *{baseball, cowboy, graduation, party, winter}*
food:	$object_B \in$ *{apple, banana, orange, pear, grape}*
hair:	$object_B \in$ *{head, beard, eyebrows, eyelashes, pubic}*

Of course, not all obtained *subjects* and *objects* are correct, especially in the case of asking for *objects*. For example, triples generated for the subject *letter*, Table 5.3, are quite inferior. A human-wise evaluation was performed; see Section 5.6.3 for details.

In the prompts, we chose the *What* question word, as it is generic enough to result in diverse types of results. However, a more fine-tuned selection of the question word may result in more relevant results, as suggested in [76].

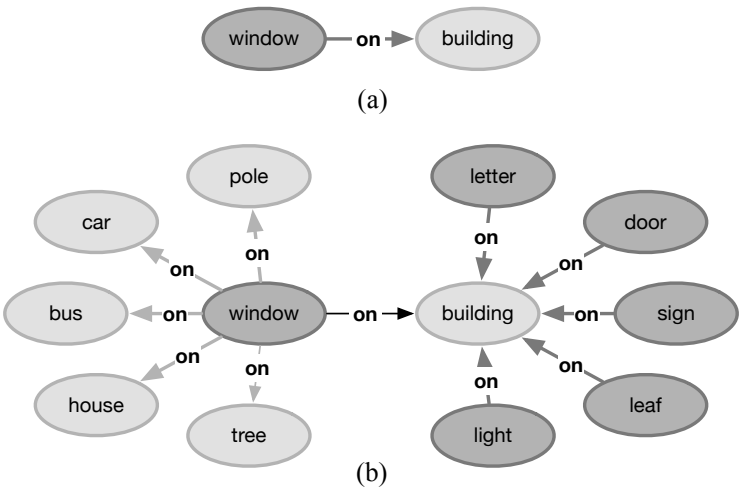


Figure 5.3: Expanded *WpKG*—simple triples: original triple (**a**); and after its extension (**b**).

We utilized the largest GPT-3 model, with 175 billion parameters, for the experiments. We started with a softmax temperature of 0.0 to obtain more deterministic results. However, we observed that the model sometimes shies away from generating text with this temperature setting and immediately generates an end token. To fix the problem, we increased the temperature to 0.7 and then to 1.0 to increase the chances for a good response.

5.5.2 Fuzzy Triples with Linguistic Terms

The remarkable abilities of GPT-3 can be utilized to extract *subjects* and *objects* when the triples need to be labeled with the degrees of the plausibility of their occurrence. Triples with such information can be added to the *WpKG* when the prompt, and its question-and-answer parts, used to query GPT-3 are constructed/designed in a specific way. The prompt templates presented in the previous section have to be modified.

To invoke responses from GPT-3 that give a quantifiable assessment of relation strength, the prompts should be more verbal to contextualize interaction with the model. The experiments with multiple approaches have led to the prompts that are the same, even if GPT-3 is asked to provide facts related to a variety of topics.

Due to the fact that two degrees of relation strength are considered, two

prompts are designed and used: one for generating triples that represent high likeliness and one for building triples that are of low likeliness. Both of them are shown in Table 5.4. A quick look at them indicates that the prompts refer to quite different domains/topics—the questions are related to window and number. Yet, they work very well with the relations we use as examples—the same as for the simple triples in Section 5.5.1.

Another interesting 'feature' of these prompts is the very little need for instantiation. Only the last questions, Q_S for *subjects* and Q_O for *objects*, Table 5.4, are initialized to reflect the relations of interest.

As an example of using the prompt templates, the results for a $relation_X$ = $relation_Y$ =*on* are included. Please note that different question templates are developed to fit various types of relations. The obtained *subjects* and *objects* are in Tables 5.5 and 5.6 for the linguistic terms **most likely** and **less likely**, respectively.

Again, not all obtained *subjects* and *objects* are correct. For example, triples ⟨*hat, (***most likely***) on, -*⟩, Table 5.5, or ⟨*hat, (***less likely***) on, person*⟩, ⟨*food, (***less likely***) on, stove*⟩, Table 5.6, are quite inferior. As before, there is also a graphical representation in Figure 5.4 of the addition of new triples with the relation *on* that have *building* as their *object*. It can be seen that the most likely *subjects* are quite reasonable, while the less likely *subjects* are a bit odd. A human-wise evaluation is performed; see Section 5.6.3 for details.

For the **most likely** case, the softmax temperature starts at 0.0 and increases to 0.7 and 1.0, in the case that no text is generated. For the **less likely** case, we observe better results if the initial temperature is set to 0.7 and increases to 1.0 if needed.

Table 5.4: Template for fuzzy triple with linguistic terms.

FUZZY_TEMPLATE_X for the linguistic term most likely
prompt: *Answer with five items separated with comma.* *Q: What* **most likely** *has window? Name five.* *A: Window is usually used to see through.* *Therefore, train, building, house, car, bus.* *Q: What number can* **most likely** *be on? Name five.* *A: Number is made of digits and can be written* *on different things for information.* *Therefore, train, sidewalk, track, street, building.* Q_S*: What is* **most likely**$\langle relation_X \rangle$ $\langle object_X \rangle$*? Name five.* Q_O*: What does/is* $\langle subject_X \rangle$ **most likely** *be/-* $\langle relation_X \rangle$*? Name five.*

FUZZY_TEMPLATE_Y for the linguistic term less likely
prompt: *Answer with five items separated with comma.* *Q: What* **less likely** *has window? Name five.* *A: Window is usually used to see through.* *Therefore, hat, drawer, vase, basket, box.* *Q: What number can* **less likely** *be on? Name five.* *A: Number is made of digits and can be written* *on different things for information.* *Therefore, window, people, rock, tree, jacket.* Q_S*: What is* **less likely**$\langle relation_Y \rangle$ $\langle object_Y \rangle$*? Name five.* Q_O*: What does/is* $\langle subject_Y \rangle$ **less likely** *be/-* $\langle relation_Y \rangle$*? Name five.*

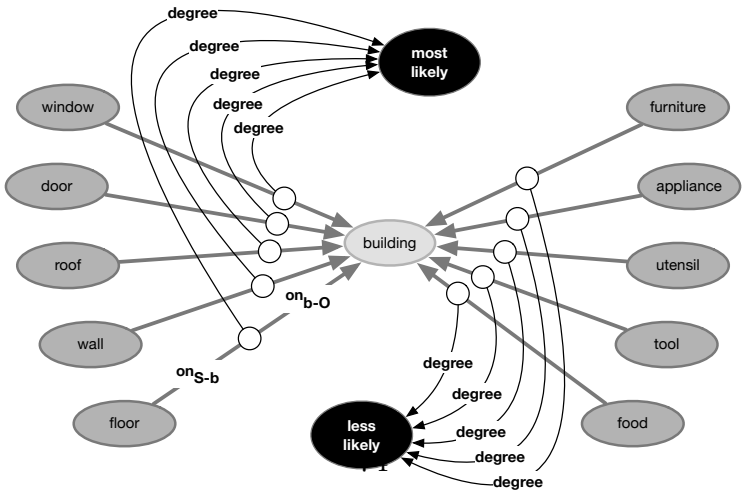


Figure 5.4: Expanded WpKG—triples with linguistic terms.

Table 5.5: Query and results for ⟨-, **most likely** *on*, -⟩ for *object*.

user:
Answer with five items separated with comma.
Q: What **most likely** *has window? Name five.*
A: Window is usually used to see through.
Therefore, train, building, house, car, bus.

Q: What number can **most likely** *be on? Name five.*
A: Number is made of digits and can be written on
different things for information.
Therefore, train, sidewalk, track, street, building.

Q_O*: What* $\langle subject_X \rangle$ *can* **most likely** *be* $\langle relation_X \rangle$*? Name five.*

where:
$subject_X$ = *{ window, letter, hat, food, hair }*
$relation_X$ = *on*
GPT-3:

window:	$object_X \in$ *{train, building, house, car, bus}*
letter:	$object_X \in$ *{train, sidewalk, track, street, building}*
hat:	$object_X \in$ *{baseball cap, fedora, beanie, cowboy hat, sun hat}*
food:	$object_X \in$ *{apple, banana, orange, grape, strawberry}*
hair:	$object_X \in$ *{person, animal, doll, toy, statue}*

5.5.3 Fuzzy Triples with Novel User-Provided Relations

The last scenario focuses on the generation of new triples that contain novel relations provided by a user. It means the user gives relations that do not exist in the initial vision-based knowledge graph. We selected three novel relations: *used for*, *made of*, and *has property*. We opted for *triples with linguistic terms* and their respective prompts instead of the *simple triples* scenario, as more information about triples is obtained. The prompt templates used here are included in Table 5.4.

The results obtained for a $subject_X$=*arm* and the user provided $relation_X \in$ *{used for, made of, has property}* are included in Table 5.7 for the fuzzy term **most likely**, and in Table 5.8 for the fuzzy term **less likely**. Graphically, the generated triples for $subject_X$ = *shoe* are in Figure 5.5. As in the previous cases, not all triples—constructed based on the obtained sets of objects—are

Table 5.6: Query and results for ⟨-, **less likely** *on*, -⟩ for *object*.

user:	*Answer with five items separated with comma.* *Q: What* **less likely** *has window? Name five.* *A: Window is usually used to see through.* *Therefore, train, building, house, car, bus.* *Q: What number can* **less likely** *be on? Name five.* *A: Number is made of digits and can be written on different things for information.* *Therefore, train, sidewalk, track, street, building.* Q_O*: What* $\langle subject_Y \rangle$ *can* **less likely** *be* $\langle relation_Y \rangle$*? Name five.*
where:	$subject_Y$ *= { window, letter, hat, food, hair }* $relation_Y$ *= on*
GPT-3:	
window:	$object_Y \in$ *{number, people, rock, tree, jacket}*
letter:	$object_Y \in$ *{number, people, rock, tree, jacket}*
hat:	$object_Y \in$ *{window, book, cat, person, wall}*
food:	$object_Y \in$ *{sink, counter, stove, refrigerator, table}*
hair :	$object_Y \in$ *{shoulder, leg, foot, arm, hand}*

satisfactory. The human evaluation results are presented in Section 5.6.3.

Table 5.7: Query and results for ⟨-, *(***most likely***) used for/made of/has property,* -⟩ for *object.*

---

user:
Answer with five items separated with comma.
Q: What **most likely** *has window? Name five.*
A: Window is usually used to see through.
Therefore, train, building, house, car, bus.

Q: What number can **most likely** *be on? Name five.*
A: Number is made of digits and can be written on
different things for information.
Therefore, train, sidewalk, track, street, building.

Q_O*: What is* $\langle subject_X \rangle$ **most likely** $\langle relation_X \rangle$*? Name five.*

where:
$subject_X$ = *arm*
$relation_X \in$ **{used for, made of, has property}**
GPT-3:
used for: $object_X \in$ *{lifting, carrying, pushing, pulling, holding}*
made of: $object_X \in$ *{human, animal, plastic, metal, wood}*
has property: $object_X \in$ *{to move, to bend, to be strong, to be flexible, to grip}*

Table 5.8: Query and results for ⟨-, *(***less likely***) used for,* -⟩ for *object.*

user:
Answer with five items separated with comma.
Q: What **less likely** *has window? Name five.*
A: Window is usually used to see through.
Therefore, train, building, house, car, bus.

Q: What number can **less likely** *be on? Name five.*
A: Number is made of digits and can be written on different things for information.
Therefore, train, sidewalk, track, street, building.

Q_O*: What is* $\langle subject_Y \rangle$ **less likely** $\langle relation_Y \rangle$*? Name five.*

where:
$subject_X$ = *arm*
$relation_X \in$ **{used for, made of, has property}**
GPT-3:
used for: $object_X \in$ *{hat, drawer, vase, basket, box}*
made of: $object_X \in$ *{metal, plastic, glass, wood, fabric}*
has property: $object_X \in$ *{number, window, glass, bottle, box}*

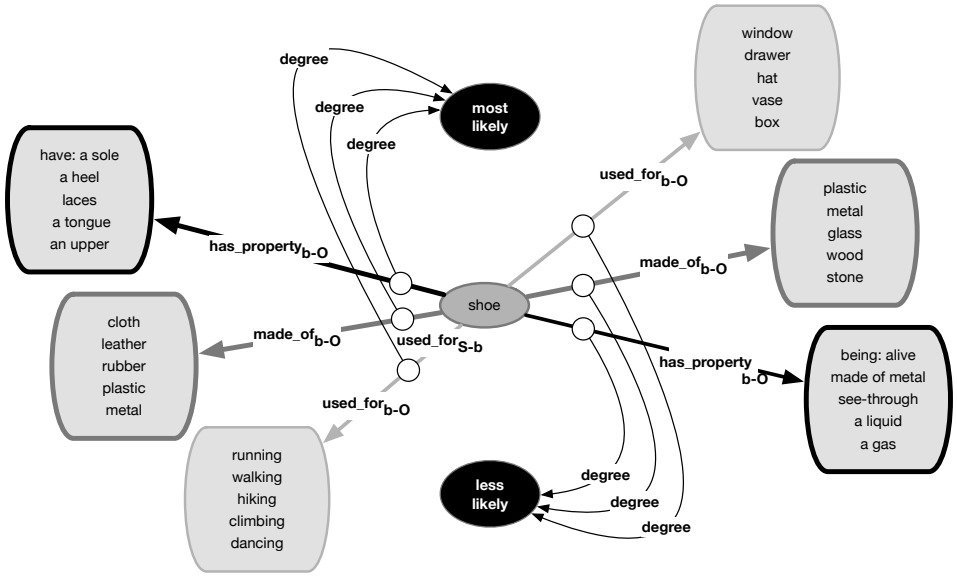


Figure 5.5: Expanded *WpKG*–fuzzy triples with *shoe* as their *subject* and user-provided relations *has_property, made_of, used_for.*

5.6 Discussion

The presented method for expanding existing commonsense knowledge graphs represents an example of a new approach to constructing knowledge graphs in a specific domain using very large language models and prompts. It can be said that these techniques are in their infancy; therefore, there are a number of aspects that need to be investigated regarding the approach itself as well as evaluation of the obtained results.

5.6.1 Vision-Based Commonsense Graph

Similar to how toddlers learn about their environment, our approach is based on two steps. First, we generate commonsense knowledge using vision models and then expand it using language models.

The evaluation of the weighted commonsense knowledge graph generated using only visual data is presented in Table 5.9 from our previous work [2], [66]. Three different approaches for determining the weights (strengths) of relations are proposed and evaluated. Depending on the weighting mechanism, the accuracy of the generated commonsense triples ranges from 87.6% to 93%. Among these, the DPbM (detection probability-based method) correlates highly with human commonsense, while other methods still show good results.

Table 5.9: Human evaluation of the three weighting mechanisms defined in [2]. Three reviewers were given top 100 triples from each restaurant and classroom contextual commonsense knowledge graphs (total of 600 evaluations per method). Alpha is Krippendorff's Alpha [57] measuring consensus among evaluators.

Weighing Schema	**Accept**	**Reject**	**N/A**	**Accuracy (%)**	**Alpha**
DPbM	560	22	18	**93.0**	**0.78**
ROM	526	60	14	87.6	0.63
WOM	538	51	11	89.7	0.72

5.6.2 Preliminary Experiments with Language Models

The high accuracy obtained using automatic vision-based weighted commonsense knowledge generation does come with some specific challenges of its own. For example, the concept and relation vocabulary is limited only to the dictionary provided to the underlying models during the supervised training of the vision models. Adding a new vocabulary requires several time-consuming and costly tasks. They include human annotation on images to label objects and relations between them and then the fine-tuning of models for object detection and scene graph generation. Even if we accept the time and cost of adding a new vocabulary, it is shown in [55] that there is a bias toward the most common relationship type. It prevents the process from effectively going beyond specific vocabulary.

To address the issue of limited vocabulary, we have investigated using language models to extend the initial vision-based commonsense knowledge graph. We opted to use very large language models, such as GPT-3, for two main reasons. One is their capability to offer new concepts beyond the known ones with acceptable precision. The other reason is the flexibility and time/cost saving of using prompts instead of fine-tuning, which usually requires large amounts of costly human-annotated data.

Our experimental results support the overfitting statement explained in [77], [78] stating that training on specific data reduces performance on novel data. We initially experimented with comparing one-shot-prompted 175-billion-parameter unsupervised-trained GPT-3 versus variations of smaller language models fine-tuned on an initial 5000-triple vision-based commonsense knowledge graph. Although the GPT-3 result accuracy was lower than a fine-tuned language model, the novelty of the vocabulary offered was much better. GPT-3 with 175 billion parameters predicted 15 times more vocabulary than the RoBERTa-large model with 355 million parameters.

5.6.3 Evaluation of Commonsense Knowledge Graph

To the best of our knowledge, there is limited benchmark data or a well-established method suitable for evaluating constructed commonsense knowledge graphs, especially when there are mostly novel generated concepts. There are benchmarks introduced in works such as [81], but are more related to knowledge base completion rather than expansion to new concepts. For mostly novel concepts, human evaluation of the results seems to be the preferred method, mainly in generative model scenarios, as performed in [61].

In this work, the process applied to assess the quality of the constructed commonsense knowledge graph is fully based on human evaluation using Amazon MTurk annotators. Amazon Mechanical Turk `https://www.mturk.com` (accessed on Aug. 12, 2022) (MTurk) is a crowd-sourcing marketplace that provides, among multiple services, assistance in data annotation tasks. Three sets of validation tasks are performed for simple triples (Section 5.5.1), fuzzy triples (Section 5.5.2), and fuzzy triples with user-provided relations (Section 5.5.3).

The evaluation results are shown in Table 5.10 for only the new triples that did not exist in the original commonsense knowledge graph. As it can be seen, the results are encouraging. To gain some insight into the evaluation process and to better understand the evaluation results, it should be stressed that MTurk controls who is involved in the evaluation task. To increase the confidence in results, each triple is evaluated by three independent annotators.

To make the evaluation task easier and more intuitive for the annotators, we generated sentences from triples. Based on each predicate, a manual pattern is introduced. Once a sentence is generated using a fixed pattern, it is passed through an off-the-shelf grammar correction module to fix obvious errors. The sentences are then manually vetted to make sure they are grammatically correct and are based on the original triples.

In the description given, the annotators were asked to assume visual commonsense when encountering any of these statements. For example, in the case of *It is likely to see cloud behind cow.*, we asked them to imagine that they

are in a field and they see cows. Then it makes sense to see clouds behind the cows.

Some examples of the triples and their evaluation scores are presented:

- *Shoe is used for running.* –> Correct with 0.95 confidence.
- *Shoe is not likely to be alive.* –> Incorrect with 0.95 confidence.
- *Shoe is not usually made of stone.* –> Correct with 0.65 confidence.

As we can see in the examples, finding a well-understood and easy-to-annotate verbalization of triples can affect the result. For example, in the case of *Shoe is not likely to be alive.*, the statement makes sense based on our understanding; however, it was not the case with the three annotators.

Table 5.10: Results of human evaluation of generated triples. Overall, *Likely* and *Unlikely* columns show the accuracies regarding total triples, most-likely triples, and less-likely triples, respectively. *N* represents the number of triples evaluated in each case.

Triple Type	***N***	**Overall Accuracy**	***Likely* Accuracy**	***Unlikely* Accuracy**
Simple	122	72.95%	N/A	N/A
with Linguistic Terms	287	67.94%	68.09%	67.81%
with New Relations	148	72.97%	66.22%	79.73%

A few examples are analyzed under Table 5.11 to understand the obtained results better. Triples without linguistic terms are called *Simple*. Triples *with Linguistic Terms* contain two terms, **most likely** and **less likely**. Triples *with New Relations* refer to triples with linguistic terms generated with predicates that do not exist in the initial commonsense knowledge graph. For brevity, the initial parts of the prompts are removed. Only the last part of the prompt (question) is kept. The process of generating triples *with Linguistic Terms* and *with New Relation* uses the chain-of-thought prompting methods, shown

in Sections 5.5.2 and 5.5.3, while *Simple* triples are generated using a simple question and answering prompting method, shown in Section 5.5.1.

The obtained results are compared with the results found in similar works. TransOMCS paper [82] reports an overall accuracy of 56% while focusing on the automatic mining of commonsense knowledge from linguistic graphs. The results in TransOMCS are based on 100 randomly selected tuples from the overall results set, which five Amazon mTurk workers evaluated. Another comparable work focuses on symbolic knowledge distillation from large language models, mostly about commonsense social relations, without relationship weights [61]. This work reports a human-evaluated correctness percentage of 73.3% when GPT-3 is used with prompts to complete a knowledge graph. The reported value is close to the comparable case of *Simple* triples as shown in Table 5.10. The approach used in [61] requires text completion for every subject and predicate to generate each triple. On the other hand, our approach uses prompts that generate $N = 5$ new concepts during a single run. It results in roughly one-fifth of the cost when both methods use the same model.

To further demonstrate the scalability of the proposed method, we generated 1,905 triples with linguistic terms. Triples with 13 different predicate types from our vision-based commonsense knowledge graph were used for the generation purpose. There are 1,075 triples with the linguistic term **less likely** and 830 with the term **more likely**.

All the triples were evaluated using three Amazon mTurk annotators on the Amazon SageMaker platform. The human evaluations of **more likely** triples resulted in higher accuracy of 72.15%, while the **less likely** triples resulted in an accuracy of 62.1%. We only considered triples with at least 95% evaluation confidence among the three annotators (662 triples). The evaluation of triples with different predicate types and linguistic terms resulted in different accuracies, as shown in Figure 5.6. This scaling experiment shows that the generated dataset size can expand from the initial hundreds of triples to thousands and beyond.

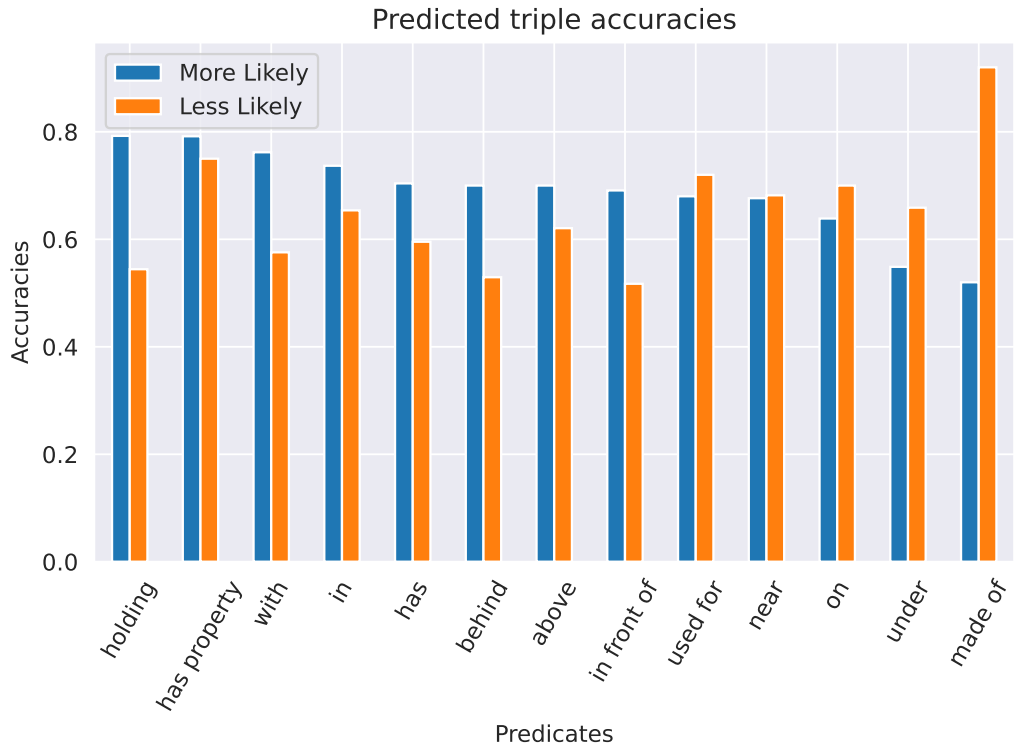


Figure 5.6: Human (mTurk) annotation accuracy of different predicate types and linguistic terms.

5.7 Conclusion

There is a growing interest and a need for collecting and storing knowledge that represents information about real-world scenarios and things and activities of everyday life. That type of information—named commonsense—becomes essential when one wants to build autonomous systems that exist around us and assist us in daily duties.

The commonsense knowledge is present in different visual and verbal forms and is learned via observations, experiences, and interaction with others.

A simple attempt to address extracting commonsense knowledge and representing it as a graph is presented here. The previous work [66] showed a method of analyzing images and constructing a commonsense knowledge graph via the fusion of multiple scene graphs extracted from images.

This paper, perceived as a continuation of the work on images, presents a

methodology for expanding existing commonsense graphs with facts retrieved from language models. The development of very large language models opens an opportunity to use them for multiple tasks involving retrieving pieces of information and facts in various domains. This capability of the models was utilized here to pull out commonsense information that is easily added to the existing knowledge graphs. Specific prompts and their templates were constructed to retrieve related information. This information was transformed into triples and added to the commonsense graph. Three different types of new triples were considered: simple ones, fuzzy ones with linguistic terms describing degrees of their likeliness, and ones with specific relations provided by the user.

A validation process of new triples was designed and executed—the Amazon service called Mechanical Turk was utilized. The obtained evaluations confirmed the usefulness of the proposed methodology for expanding commonsense graphs.

At the same time, more work is needed to construct prompts that improve the correctness of retrieved information and create triples with more subtle degrees of likeliness. Additionally, more investigation regarding the suitability of different language models is mandated. In this paper, we used the chain-of-thought prompting method [74]. While this prompting method leads to good results, it seems interesting and important to investigate other prompt methods, such as [83], to see if better and more accurate results are achievable.

Table 5.11: Examples: two correct and one incorrect for each type of generated triple. Correct parts of the response are in teal color, while the incorrect parts are in red color.

Simple	**with Linguistic Terms**	**with New Relations**
Q: What is in room? Name five.	Q: What is most likely to be in bowl? Name five.	Q: What is shoe most likely used for? Name five.
A: bed, table, chair, lamp, computer.	A: Bowl is a container and can hold different things. Therefore, rice, cereal, milk, soup, fruit.	A: Shoe is used to protect feet. Therefore, running, walking, hiking, climbing, dancing.
Q: What does face have? Name five.	Q: Where is flower less likely in? Name five.	Q: What property does sidewalk most likely have? Name five.
A: eyes, nose, mouth, ears, chin.	A: Flower is a natural thing and needs soil, water and sunlight to grow. Therefore, sink, oven, book, pencil, phone.	A: Sidewalk is a path for pedestrians. Therefore, it is usually level, made of concrete, has a curb, is smooth, and is wide.
Q: What food can be on? Name five.	Q: What hat can most likely be on? Name five.	Q: What property does arm less likely have? Name five.
A: apple, banana, orange, pear, grape.	A: Hat is worn on the head and can be made of different materials. Therefore, baseball cap, fedora, beanie, cowboy hat, sun hat.	A: Arm is a body part and usually has skin, muscle, bone, and hair. Therefore, number, window, glass, bottle, box.

Chapter 6

Super-Prompting: Utilizing Model-Independent Contextual Data to Reduce Data Annotation Required in Visual Commonsense Tasks

6.1 Introduction

Human annotation is time-consuming and is also a financial burden for research groups, startups, and companies. To put it in context, almost $240,000 has been spent on the annotation of the Visual Commonsense Reasoning in Time (VisualCOMET) dataset and this figure only includes the payment to crowd-workers from Amazon Mechanical Turk [84]. The real financial burden can be much higher when including the time value of the staff involved in the annotation process.

Although large pre-trained language models, such as GPT-3 transformer [85], are impressive at multi-task few-shot learning, their huge size can be prohibitive for different scenarios, including on-device applications. Fine-tuning still plays an important role in achieving the state of the art, even with a relatively smaller model. As an example, the two current leading models[1] (better than human baseline) on SuperGLUE task [86] are fine-tuned variants of T5 [87] and DeBERTa [88] language models, while GPT-3 is at 14th place.

[1] https://super.gluebenchmark.com/leaderboard

Our goal is to devise a model-independent process that could improve results based on fine-tuning with much less annotated training data.

6.2 Related Work

Several recent works have focused on improving fine-tuning methods in language models, such as [89], [90], [91], and [92]. The focus has been put mostly on optimization and regularization, but not on using less data for fine-tuning. The results from those studies are complementary to our work.

Some previous efforts have been put on prompt-based fine-tuning to improve classification or regression tasks in natural language processing (NLP). [93] and [94] convert textual inputs into cloze-style questions with a task description. [95] studies smaller language models for few-shot learning capability by using automatically-generated prompts for fine-tuning and by incorporating demonstrations into context.

On another topic, a group of recent research studies, including [96], [97] and [98], aim at task-dependent added parameters to adapt models to different tasks. This way, one does not need to re-train a complete model to fine-tune it to a specific task but only needs to re-train a fraction of parameters.

There is a recent body of work that utilizes inherent knowledge of language models combined with fine-tuning on specialized large-scale training datasets to infer different commonsense and causal scenarios. [14] uses generative language models to expand on ATOMIC [10] and ConceptNet [9] commonsense knowledge graphs. [48] introduces an updated knowledge graph similar to ATOMIC and uses BART [99] encoder-decoder model to generate new knowledge. [100] uses generative language models to expand on an introduced knowledge base of causal mini-story explanations.

Given the success of prompt-based fine-tuning and in-context learning in classification and regression tasks, we are motivated to assess similar principles in the context of commonsense generation using generative language models, which are fine-tuned on a commonsense knowledge graph.

6.3 Dataset

For this paper, we have selected a multi-modal commonsense knowledge graph for fine-tuning. The Visual Commonsense Reasoning in Time (VisualCOMET) dataset [84] consists of 1.4 million commonsense inferences over 59,356 images and 139,377 specific events at present. The dataset has human-annotated inferences regarding three different aspects: the intention of the person mentioned, the possible events that could happen next, and the possible preceding events. The inferences are made based on a single image. The annotators have access to short clips before and after the event, which are not part of the dataset. Each image is also annotated with event and place descriptions. There is a total amount of 1,465,704 commonsense inferences.

The images are sourced from the VCR dataset [101]. The images usually have a complex visual scene with multiple people and activities present. This dataset includes automatically-detected object bounding boxes and people are annotated with numerical tags.

6.4 Method

In this work, we focus on using generative language models and analyze how prompt-based fine-tuning and in-context learning could help to reduce the size of the data required for fine-tuning training.

As seen in Fig. 6.1, there are several scenarios where extra context could help lead the generative language model to a correct answer, but lack of correct understanding about the scene and the event text can result in incorrect results. Extra human annotations, focused on these shortcomings, could improve the results, but that comes with extra time and money expenditure.

We propose using the underutilized context already present in text and image, then transforming them to a form that is usable by most transformer models, which is a sequence. We analyze if this kind of addition helps the language model achieve better results in the case of limited annotated data available.

Assuming the added context text is represented with c and its tokenized

version with $\{c\}$, we can represent the context with $\{c\} = \{w_1^c, w_2^c, ...w_q^c\}$, where w_i^c represents each token created from tokenization of the context c. This context is merged with tokenized versions of event and place, which are represented as: $\{e\} = \{w_1^e, w_2^e, ...w_n^e\}$ and $\{p\} = \{w_1^p, w_2^p, ...w_m^p\}$, respectively. Using the merged versions of event and place texts with context, the updated sequence-to-sequence loss can be written as:

$$\begin{aligned}\mathcal{L} = &- \sum_{i=1}^{n} \log P(w_i^e | w_{<i}^e, v)) - \sum_{i=1}^{m} \log P(w_i^p | w_{<i}^p, e, v)) \\ &- \sum_{i=1}^{q} \log P(w_i^c | w_{<i}^c, p, e, v)) \\ &- \sum_{i=1}^{l} \log P(w_{hi}^r | w_{h<i}^r, c, p, e, v) \end{aligned} \tag{6.1}$$

where v represents visual features, including overall images and person-specific boxes, r represents inference prompts, which could be intent, before and after, and $w_{<i}^*$ represents past tokens for each case.

6.5 Experiments

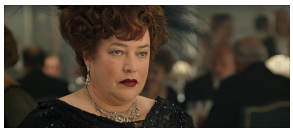
The goal of the experiments is to see how much we could reduce the annotated data and still achieve results comparable to a case where the full human-annotated data is used. We tried different contextual data, which did not require extra annotations, such as captions, facial expressions, and related concepts.

As shown in Fig. 6.1, we can intuitively see that some extra context could potentially help the language model to reach a more logical deduction of intention, past, and future events.

For each scenario, the VisualCOMET dataset provides several human annotations for comparison, each showing intent of a person, what could happen next, and what happened before. The experiments are evaluated using BLEU [102], METEOR [103] and CIDEr [104] automatic metrics to compare the generated texts for different scenarios of before, intent and after with the human-annotated texts.

(a) **Event:** Person-4 is sitting on the couch with her legs over the arm.
Place: In a living room
Annotated Intent Inferences:
1) be comfortable
2) get cozy
A Predicted Intent: Show boredom
Missing context: Happy facial expression

(b) **Event:** Person-2 is taking a tiny spoon and scooping up a heap of caviar.
Place: In a dining room
Annotated Intent Inferences:
1) live luxuriously
2) enjoy a delicious treat
A Predicted Intent: keep everything neat
Missing context: Caviar is a luxury edible.



(c) **Event:** Person-1 is sitting down staring at someone angrily.
Place: In a dining room
Annotated After Inferences:
1) choose not to express her anger verbally
2) look down in front of her
3) slam the table
4) walk out of the restaurant
A Predicted After: Fight
Missing context: Objects in a dining room and its location.

Figure 6.1: Predictions based on the fine-tuned language model introduced in [84]. Each example shows a piece of missing contextual information that could be utilized.

Method	BLEU-2	METEOR	CIDEr
GPT-2 [84]	13.81	10.85	15.37
Concept Word (NVP)	17.25	12.17	19.79
Concept Word (VP)	17.17	12.28	19.34
Concept Sent. (NVP)	14.92	11.25	16.9

Table 6.1: Effects of adding relevant concepts. Results are shown at the fourth epoch using almost 25,000 (22%) of the available annotated data. NVP: No Validation Prompt. VP: Validation Prompt.

We tried two different methods of adding relevant concepts. One method is based on converting relevant concept graphs into a readable sentence and the other method is based on only prepending concept words to the target sentence. In either method, the text is scanned for concepts, and the related concepts are extracted based on a commonsense knowledge graph such as in [9], [66]. Although sentence-based inputs perform well, they require a longer input width that may not be available given the language model. To sort relevant concepts, crowd-based scores or frequency scores are used based on the specific knowledge graph used. These triples are then converted to text with some hand-designed rules. An example of this process is shown in Fig. 6.2a. Table 6.1 shows three of the top-performing models with added conceptual contexts. They are compared with the original data, which does not have any added context. Evaluation is done on a validation dataset with a size of a hundred. Concept words added in this specific scenario are connected via HasProperty and PartOf predicates. Concept sentences use the HasProperty predicate. Adding similar information during inference time does not result in much improvement in this specific case. More comparisons can be found in the Appendix.

As seen in Fig. 6.1a, lack of the model's attention to some visual cues, such as facial expressions, could also result in errors of judgment. To fix this issue, we trained a ResNet [105] model on FER2013 [106] dataset with almost 70% accuracy. The dataset consists of human face images and emotion labels of angry, disgust, fear, happy, neutral, sad, and surprise. Only the emotion of people mentioned in the event text is processed. The results are

Method	BLEU-2	METEOR	CIDEr
GPT-2 [84]	13.81	10.85	15.37
FE (NVP)	14.45	11.27	15.7
FE (VP)	15.11	11.23	16.03

Table 6.2: Effects of adding information about facial expressions. Results are shown at the fourth epoch using almost 25,000 (22%) of the available annotated data. NVP: No Validation Prompt. VP: Validation Prompt. FE: Facial Expressions.

Method	BLEU-2	METEOR	CIDEr
GPT-2 [84]	13.81	10.85	15.37
Caption (NVP)	14.08	10.78	15.63
Caption (VP)	16.49	11.85	18.8

Table 6.3: Effects of adding image captions. Results are shown at the fourth epoch using almost 25,000 (22%) of the available annotated data. NVP: No Validation Prompt. VP: Validation Prompt.

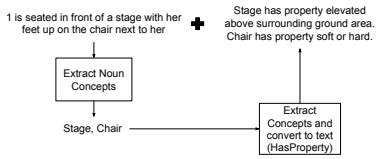
then prepended to the event text. An example of this process is shown in Fig. 6.2b. Table 6.2 shows effects of adding facial expressions as a context in the final performance of the model. Contrary to relevant concepts, adding facial expressions during inference time improves the results. Evaluation is done on a validation data size of a hundred.

Another type of automatically-generated context that we experimented with is image captioning. The idea is that some of the image dynamics may have been missed, even though image features are fed into the GPT-2 model. Adding generated captions proves to be effective as shown in Table 6.3. Meshed-Memory transformer model [107] with beam search decoding is used for image captioning. The process of adding these captions is illustrated in Fig. 6.2c.

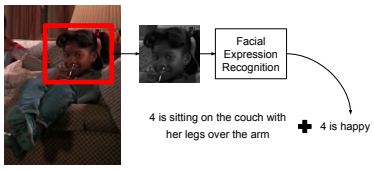
A mixture of different contextual information is shown to be more effective than individual ones. A combination of concept words, image captions, and facial expressions of relevant individuals in the image achieve the best result compared to other experiments. As seen in Table 6.4, this combination can achieve comparable results to full-data finetuning by only using 35%-40% of

Method	Inference Data	Data Size	BLEU-2	METEOR	CIDEr
GPT-2 [84]	N/A	111,796 (100%)	18.05	**13.21**	22.72
CW + C + FE	C + CW + FE	39,000 (35%)	18.38	12.97	22.65
CW + C + FE	C + CW + FE	45,000 (40%)	**18.58**	13.01	**22.97**

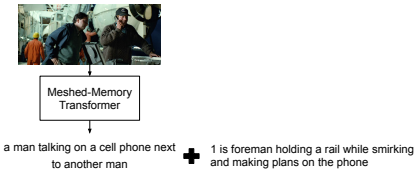
Table 6.4: Analyzing the effect of combining multiple contextual data. All models are finetuned for five epochs. Contexts are added based on the order shown. CW: Concept Words. C: Captions. FE: Facial Expressions.



(a) Process of adding relevant concepts.



(b) Process of adding facial expressions.



(c) Process of adding image captions.

Figure 6.2: The process of extracting and adding prompts shown through examples.

the annotated data. This results in less human time spent doing annotations and can potentially reduce costs and completion times of projects. Results of other experiments are included in the Appendix.

To reduce the effects of other variables in these experiments, we have limited ourselves to only train the final models for five epochs. The decoding method and hyperparameters are also kept constant throughout the experiments. We use nucleus sampling [80] with $p = 0.9$ to generate five sentences for each scenario of intent, before and after. The finetuning was run on two NVIDIA RTX GPUs with 24 GB memory each. For the case with all concept words, captions, and facial expression contexts, the fine-tuning time is around 1.5 hours per epoch while using mixed precision.

6.5.1 Prompt Selection Details

Experimentation results using different types of training and inference prompts are included in this section. The model used in the experiments is GPT-2 as described in [84].

The best types of prompts are chosen to be combined. The experiments show that the order in which prompts are added can affect the final results.

The vision-based inference prompts seem to better affect the final metric results when compared to the text-based inference prompts. This could be due to the lack of enough visual attention paid during the decoding process. Future work could involve developing a multimodal model that makes better use of visual contexts not only during the training phase, but also the inference time.

The quality of the annotated data can have an impact on the training model. We do not hand-select the annotated data based on quality and this may result in variability in final results when training with different data sizes. It can be a good practice to assess the quality of the annotated data and prompts based on the final goal of the model.

6.6 Conclusion

In this work, we analyzed the effects of automatically-generated contexts in multimodal transformer models used in a commonsensical task. These prompts can help us reduce the human annotation needed in the task by as much as 60%-65% and still, achieve comparable results to when the whole human-annotated dataset is used. These findings result in time and cost savings for future multimodal data annotation projects.

As future work, it is interesting to find a lower bound for data annotation reduction without affecting the final result of a model. It is also useful to find a method to automatically find and apply the best contextual data for different tasks and models.

Training Prompt	Inference Prompt	Training Data Size	BLEU-2	METEOR	CIDEr
None	None	111,796 (100%)	17.94	13.14	22.71
None	None	25,000 (22%)	13.81	10.85	15.37
CS (AtLocation)	None	25,000 (22%)	12.26	10.42	15.09
CS (AtLocation) + Place	None	25,000 (22%)	14.3	11.08	14.66
CS (CapableOf)	None	25,000 (22%)	12.65	10.6	15.9
CS (CapableOf) + Place	None	25,000 (22%)	14.78	11.16	15.1
CS (HasA)	None	25,000 (22%)	12.73	10.65	15.83
CS (HasA) + Place	None	25,000 (22%)	14.58	11.15	15
CS (HasProperty)	None	25,000 (22%)	12.25	10.5	15.52
CS (HasProperty) + Place	None	25,000 (22%)	15.25	11.38	16.3
CS (IsA)	None	25,000 (22%)	12.73	10.41	15.73
CS (IsA) + Place	None	25,000 (22%)	14.04	11.03	14.32
CS (PartOf)	None	25,000 (22%)	12.65	10.51	15.71
CS (PartOf) + Place	None	25,000 (22%)	14.26	11.19	14.64
FE	None	25,000 (22%)	14.45	11.27	15.7
FE	FE	25,000 (22%)	15.11	11.23	16.03
CW (PartOf + HasProperty)	None	25,000 (22%)	17.25	12.17	19.79
CW (PartOf + HasProperty)	CW (PartOf + HasProperty)	25,000 (22%)	17.17	12.28	19.34
C	None	25,000 (22%)	14.08	10.78	15.63
C	C	25,000 (22%)	16.49	11.85	18.8
C + FE	C + FE	25,000 (22%)	16.75	12.19	19.16
CW + C + FE	None	25,000 (22%)	12.6	10.18	14.57
CW + C + FE	CW + C + FE	25,000 (22%)	17.4	11.97	20.03
CW + C + FE	CW + FE	39,000 (35%)	16.57	12.32	19.68
CW + C + FE	C + FE	39,000 (35%)	16.71	12.27	19.01
CW + C + FE	CW + C + FE	39,000 (35%)	16.75	12.4	19.86
CW + C + FE	CW + C + FE + Syns	39,000 (35%)	16.7	12.43	19.94
CW + C + FE + PCW	CW + C + FE + PCW	39,000 (35%)	17.74	12.73	20.52
CW + C + FE	C + CW + FE	39,000 (35%)	17.34	12.45	20.11
CW + C + FE	CW + C + FE	45,000 (40%)	17.46	12.84	21.56
CW + C + FE	C + CW + FE	45,000 (40%)	17.66	12.95	21.41

Table 6.5: Experimentation results of using different training and inference prompts. The model used is GPT-2 [84]. Results are shown at epoch four and evaluated on validation data of size 100. Prompts are added based on the order shown. CW: Concept Words. C: Captions. FE: Facial Expressions. CS: Concept Sentences. Syns: Synonyms. PCW: Place Concept Words.

Chapter 7

Negated Complementary Commonsense using Large Language Models

7.1 Introduction

The larger the language models (LLMs) become, the better they demonstrate new, outstanding capabilities. For example, one is conducting a conversation about commonsense scenarios. However, our interaction with LLMs has led us to observe that the models tend to emphasize the normal flow of events and seem to struggle with questions involving a negated form of verbs, such as *not* or *cannot*. An example of that is in Figure 7.1. Therefore, in this paper, we focus on demonstrating the issue and then suggest an approach to remedy the problem.

To better clarify the problem statement, we start with an example and then formalize it using elements of the set theory. Let us look at the scenario in Figure 7.1; the standard question is "Who PersonX can be?". The answer to this question is *Santa Claus*. The answer to the *negated complementary* question – "Who PersonX *cannot* be?" – should be all valid answers which are not the answer to the standard (can be) question. A valid answer fits the scenario described. In this case, we ask about a person, so a non-person cannot be a valid answer. To better illustrate the concept of a *negated complementary* question, we refer to the basic notion of the complement of a set, Figure 7.2. Furthermore, we define a set of correct answers to a *negated complementary*,

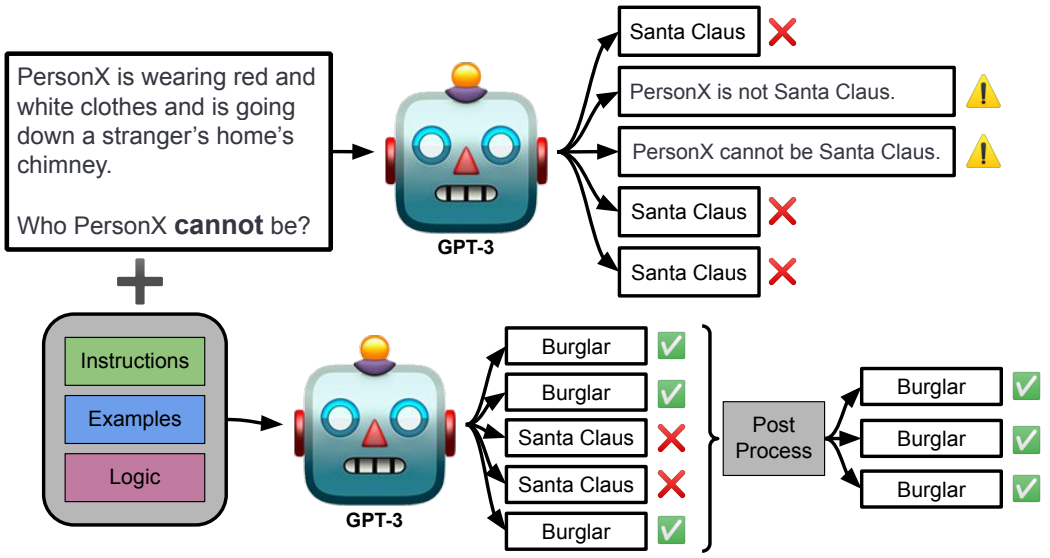


Figure 7.1: An example of a large language model (GPT-3) generating negated commonsense. Five responses per query are demonstrated. The applied pre-processing and post-processing can improve the performance of the models in negated commonsense cases. Non-specific answers, such as *not Santa*, are considered incorrect.

Equation 7.1.

$$NC = V \cap A' = \{x \mid x \in V \wedge x \notin A\} \tag{7.1}$$

where NC represents answers to the *negated complementary* question, V is the set of all valid answers, A is the set of correct answers to the standard question, and A' is the complement of A under the universal set of all answers (U).

We focus our efforts on commonsensical questions as the uncertainty of results depends on the context and experiences of people answering the questions. As defined in [62], commonsense is a collection of world models representing what is likely, plausible, or impossible. In light of that, our goal is to assess the ability of LLMs to answer plausible questions that could be refuted or accepted in a given context.

Given their pre-training nature, we hypothesize that LLMs have an inherent bias towards likely scenarios, which are the most repeated in the common text. Most of the text available on the web contains information supporting answers to 'positive' questions, like, how to do things or where to go, not to

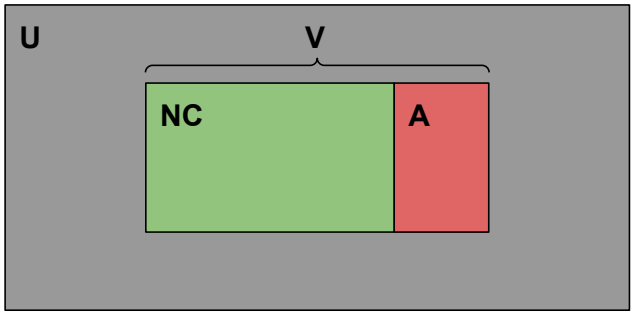


Figure 7.2: Venn diagram of answer sets: U is the universal set of answers; V is the set of all valid answers that includes two sets – correct answers to a standard question A, and correct answers to its negated complementary version NC.

questions such as how things could not be done or where not to go. It results in an imbalance of the training datasets due to the sparsity of plausible or impossible scenarios. In this paper, we demonstrate that LLMs have difficulty answering *negated complementary* questions, which results in responses representing plausible, but not impossible, answers. Although LLMs are shown to have this shortcoming, we claim that enough instructions and examples, especially showing reasoning processes, can guide the LLMs into the right path to answer *negated complementary* questions with commonsense context.

Our contributions are as follows. (1) We present an analysis exposing the shortcomings of LLMs when it comes to *negated complementary* questions in commonsensical scenarios. (2) We propose a novel methodology to improve the performance of the GPT-3 model when *negated complementary* questions are asked; compare the results with the results obtained using conventional methods. Our code, human-evaluation process, and data will be publicly available.

7.2 Related Work

Language models with transformer architectures have revolutionized the natural language processing landscape in recent years [12], [108]. It is shown that improved performance and new capabilities emerge when scaling up the size of language models [11], [85], although more is needed in challenging tasks, such as commonsense [109].

A body of research focuses on analyzing and extracting commonsense from language models [3], [48], [110], [111]. Authors of [112] focus on implications of negated statements and contradictions, where in a commonsense triple relationship (head-relation-tail), the head is either contradicted or logically negated. Comparably this paper focuses on negating relations instead of the head, as explained in Section 7.4.

7.3 Commonsense Data

The commonsense dataset used in this paper is the ATOMIC-2020 dataset [48]. It includes general purpose commonsense knowledge, divided into three main categories – physical, event-centered, and social commonsense. The ATOMIC 2020 dataset is licensed under CC-BY and we use it according to the license.

In our experiments, ten relation types are selected from the twenty-three relations from the ATOMIC-2020 dataset. These ten relation types showed worse performance in our initial evaluation of *negated complementary* questions. The relations are: *xWant*, *xReact*, *oWant*, *CapableOf*, *Desires*, *HinderedBy*, *isBefore*, *isAfter*, *AtLocation*, *HasSubEvent*.

The dataset is formatted in a triple style. Each atomic piece of data contains $\langle head - relation - tail \rangle$. For example, $\langle a\ curved\ yellow\ fruit\ (head) - CanBe\ (relation) - banana\ (tail) \rangle$.

7.4 Methodology

We propose a pipeline system to improve the performance on *negated complementary commonsense* questions. The pipeline consists of an input prompting technique and a post-processing module. The input prompt adds relevant context and logic in the form of chain-of-thought prompting [74] to improve the LLM performance. The post-processing module selects the outputs with a higher chance of correctness and filters out the rest.

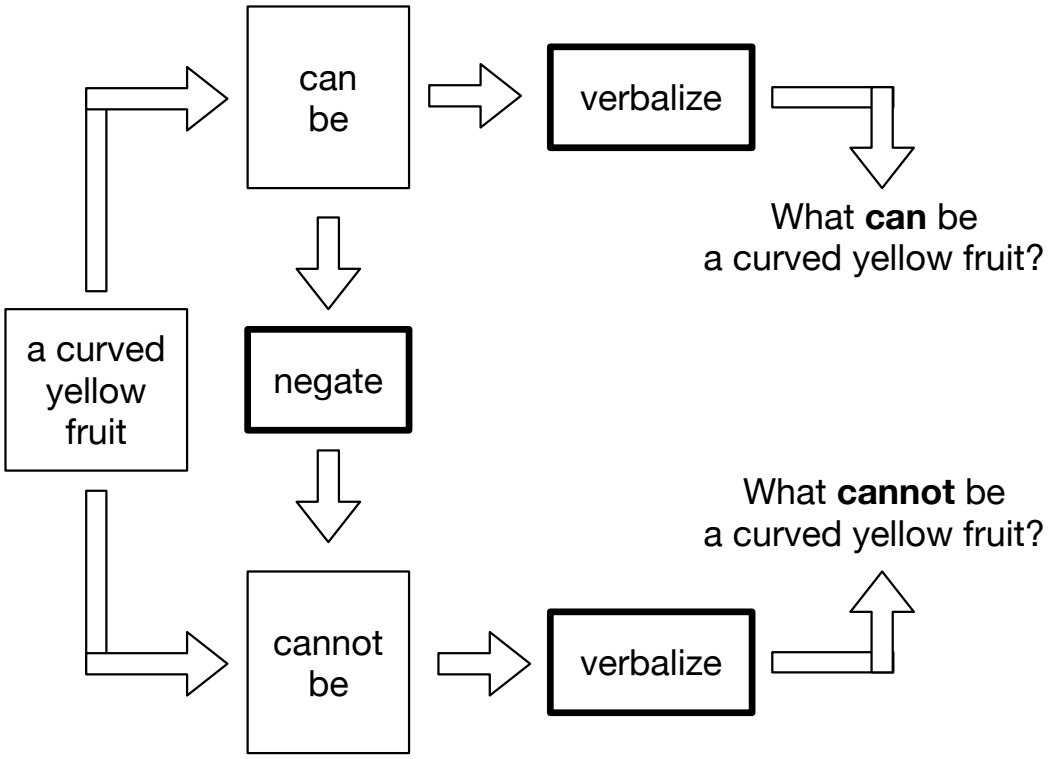


Figure 7.3: The process to automatically generate negated complementary questions from dataset triples. The head and relation nodes are used to form a question.

7.4.1 Generating Negated Complementary Questions

As described in Section 7.3, the used dataset is in the format of triples. To form a standard question, we use the head and the relation nodes and leave out the tail to be answered. By standard, we mean utilizing the head, relation, and tail, without any modifications. Assuming a triple, *a curved yellow fruit* (head), *CanBe* (relation), *banana* (tail), the standard question is *What can be a curved yellow fruit?*. The *negated complementary* question is formed by negating the relation and verbalizing the resulting triple in question format: *What cannot be a curved yellow fruit?* A valid answer to the standard question is *banana*, and a reasonable response to the *negated complementary* question is *apple*. The process is visualized in Figure 7.3. For the complete list of triple verbalizations, please see Appendix 7.5.4.

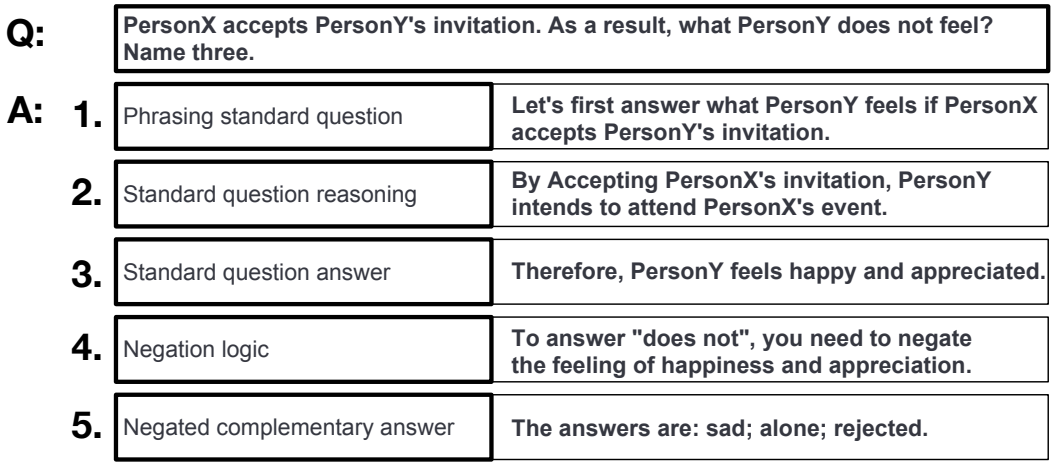


Figure 7.4: Chain-of-thought steps for each answer. The process is to answer the standard question first and then lead the model to answer the negated complementary version.

7.4.2 Prompting Technique

The proposed methodology to improve the performance of LLMs relies on building an adequate prompt. It starts with a general introduction of what negations are and emphasizes a need to pay special attention to the word *Not*. The chain-of-thought prompt in each answer has five sections in sequence: 1) phrasing standard question; 2) standard question reasoning, 3) standard question answer; 4) negation logic, and 5) *negated complementary* question answer. The steps are visualized in Figure 7.4. For a fair comparison, we used the same number of five question/answer examples in the prompts. We also used the same questions for all prompts.

7.4.3 Post Processing

Inspired by [113], we feed the question and answer pair back to the GPT-3 model and ask if it considers a question/answer pair correct. The prompt has instructions for assessing an answer and includes five sample questions/answer pairs. Interestingly, this extra step can improve the results by almost one percent. To better understand the effect of this step, please refer to Table 7.2.

7.5 Experiments

Experiments are conducted on each type of relation mentioned in Section 7.3. A hundred data points (triples) are sampled randomly from the dataset. The head and relation from each triple are verbalized and fed into the GPT-3 model (*text-davinci-002*). The goal is to predict the tail for two forms of questions: (1) standard question; (2) *negated complementary* question. For each question, three responses are requested from the model. They are then parsed, and the answers (tails) are automatically extracted. Therefore, three possible tails are obtained for each head and relation, which results in 600 total answers per method.

In social commonsense scenarios, PersonX and PersonY are used in place of gender-specific pronouns to make the questions and answers gender-neutral.

The experiments are done using the GPT-3 model [85] with version *text-davinci-002*, which has 175 billion parameters. The temperature is set to 0.7, and in case of no answer, it is increased to 1.0. The maximum length of the output is set between 100 and 150 tokens, depending on the method. The presence and frequency penalties are set to 0. GPT-3 is commercially available, and we have used it within its intended usage and terms of service.

7.5.1 Human Evaluations

We use Amazon mTurk evaluations via AWS SageMaker to evaluate the results. Each answer is written in a sentence format and given to nine different annotators for assessment. Instructions and examples are provided with each question to assist the annotators better. The options to choose from are: (1) Makes sense; (2) Sometimes makes sense; (3) Does not make sense or incorrect; (4) The first part and the second part are not related; or not enough information to judge; (5) Unfamiliar to me to judge. The first two options are considered correct, the second two are considered incorrect, and the last is considered unfamiliar. To measure inter-rater reliability, we use Krippendorff's alpha and make sure the value is above acceptable amounts (minimum 0.667) [57]. The evaluators were paid based on AWS guidelines.

Method	Standard	Negated Complementary
Few-shot	**88.7%**	78.7%
Ours	88.1%	**89.8%**

Table 7.1: Our method compared with the few-shot method when applied to ATOMIC-2020 dataset.

7.5.2 Results

As seen in Table 7.1, our method outperforms the few-shot method by more than eleven percentage points when answering *negated complementary* questions. The few-shot method includes five different questions in the prompt with their answers without chain-of-thought prompting. The performance of our method can mainly be attributed to the specific chain-of-thought prompting with negation logic description, Figure 7.4. More information about the main contributing factors is in Section 7.5.3. Although chain-of-thought prompting seems to help the *negated complementary* questions, it adversely affects answers to the standard questions. Please note that the chain-of-thought prompt for the standard questions does not include negation logic, and a post-processing technique similar to negated complementary questions is performed.

7.5.3 Ablation Studies

To gain insight into the importance of elements of our method, we perform an ablation study, Table 7.2. As we can see, adding standard question reasoning (step 2 of Figure 7.4) results in more than 7% improvement in the results. Adding the thought process explaining the negation logic (steps 1, 3, and 4 of Figure 7.4) adds another 3% performance improvement. Finally, the post-processing (Section 7.4.3) is responsible for about 1% improvement in the results.

7.5.4 Verbalizations

The questions are verbalized from triples using pre-defined formats. Table 7.3 summarizes the verbalizations organized by relation types. The question template formats are inspired by the sentence format used in [48].

Method	Neg. Comp.
Ours	89.8%
Ours-wo-pp	89.0%
Ours-wo-nl-pp	86.0%
Few-shot	78.7%

Table 7.2: Ablation study of the method: *Ours-wo-pp* is ours without post-processing; *Ours-wo-nl-pp* is ours without negation logic and post-processing.

7.5.5 Human Evaluation Instructions

The following instructions are given to each human evaluator to better understand and respond to the task:

Based on your own commonsense, choose one of the five options. Examples are provided in the description. IMPORTANT: Please note the CANNOT, DO Not, and other negated cases.

Instruction notes: Based on your own commonsense, choose one of the five options. Examples are provided in the description.

IMPORTANT: Please note the CANNOT, DO Not, and other negated cases.

1. Instead of names, PersonX and PersonY are used to be gender-neutral.
2. Please ignore grammatical errors and focus on commonsense.
3. If a response is vague, such as *not fireman*, or if a random word does not fit the scenario, please choose 4 (not enough information).

Added to the instructions, we also provided some examples to clarify the task better:

Unfamiliar to me to judge: PersonX discovers a new planet. The planet is in the Alpha Centauri system.

First part and second part are not related! Or not enough information to judge: PersonX rides a bike. Elephants are not birds. (Although the second part is correct, it is not related to the first part)

Makes sense: It is NOT likely to see an elephant on the table.

Does not make sense: It is likely to see an elephant on the table.

Relation	**Question**
xWant	[head]. What does PersonX want to do?
	[head]. What does PersonX not want to do?
xReact	[head]. What does PersonX feel about it?
	[head]. What does PersonX not feel about it?
oWant	[head]. What does PersonY want to do?
	[head]. What does PersonY not want to do?
CapableOf	What is [head] capable of?
	What is [head] not capable of?
Desires	What does [head] desire to do?
	What does [head] not desire to do?
HinderedBy	[head]. What can hinder/obstruct it?
	[head]. What cannot hinder/obstruct it?
isBefore	[head]. What happens before it?
	[head]. What does not happen before it?
isAfter	[head]. What happens after it?
	[head]. What does not happen after it?
AtLocation	Where is the [head] located?
	Where is the [head] not located?
HasSubEvent	[head]. What will you do while: [head]?
	[head]. What you will not do while: [head]

Table 7.3: Question templates for each relation type. The first row for each relation shows the standard question format, and the second row shows the negated complementary format. [head] refers to the head in a triple.

The answers are written in a text format to make it easier for human evaluators. Here is an example:. Here is an example:

PersonX sends PersonY to the showers. PersonX will not be amused.

7.5.6 ChatGPT

The experiments of this paper were done using *text-davinci-002* version of GPT-3. The new version of GPT-3 with reinforcement learning from human feedback (RLHF) is introduced and is called ChatGPT. Out of curiosity, we did limited experiments with ChatGPT to see if the problem of *negated complementary* questions persists. Here are a few answers from ChatGPT to the question shown in Figure 7.1:

Q: PersonX is wearing red and white clothes and is going down a stranger's home's chimney. Who PersonX cannot be?

A: It is likely that PersonX is not Santa Claus, as he is commonly depicted wearing red and white clothes and going down chimneys.

As we can see, the model answers with *not Santa Claus* and then describes Santa Clause. Asking the same question again, we get the following answer:

A: PersonX cannot be a real person as it is impossible for someone to physically fit into a chimney. The scenario you described is likely referencing a fictional character, such as Santa Claus, who is often depicted as wearing red and white clothes and entering homes through chimneys on Christmas Eve to deliver presents to children.

In this answer, there is a good start of context; however, somewhere within the text, the model forgets about the *cannot* word and again describes Santa Claus. For the third time, we ask the same question and we get the following answer:

A: It is likely that PersonX is Santa Claus, a fictional figure who is often depicted wearing red and white clothes and delivering presents to children through the chimneys of houses.

The sentence is technically correct, but it is not what we asked for, which is who *cannot* be. Therefore, we can consider the last answer incorrect as well.

Given the limited interaction with ChatGPT, it seems that the *negated*

complementary task issue persists and needs further investigation in the new version as well.

7.6 Conclusion

In this paper, we demonstrate how simple changes in question formats, which can be trivial for humans, can be challenging for large language models (LLMs). We specifically focus on *negated complementary* questions in a commonsense context, which is constructed by negating a relation in a commonsense triple. Given the vast amount of knowledge embedded in LLMs, we show that by appropriate guidance, the models could perform well on *negated complementary* tasks. Our method results in more than eleven percent improvement compared to the vanilla few-shot method. Given the widespread usage of LLMs and their growth rate, we believe focusing on and solving the model's weaknesses is imperative. As future work, *negated complementary* task can be further analyzed in different formats, such as sentence instead of a question, and also different contexts, e.g., new datasets.

Chapter 8

Conclusion

8.1 Contributions

In this thesis, we explored different aspects of commonsense understanding, generation, and reasoning by utilizing deep learning models.

We developed methodologies to autogenerate commonsense knowledge graphs using vision models inspired by how toddlers gain world understanding before lingual development (Chapter 3). As context can affect the meaning of a concept, we further expanded the vision-based commonsense knowledge graph generation methodologies to recognize the contextual importance and meaning of concepts. We also demonstrated how this contextual understanding could be used for commonsense reasoning (Chapter 4).

Similar to how toddlers build on top of their visual understanding of the world with lingual data, we used language models to expand vision-based commonsense knowledge graphs. Using language models and their inherent concept mappings in an n-dimensional space, we can expand beyond the known concepts in visual models to new concepts and novel relations between them (Chapter 5).

Commonsense knowledge is inherently uncertain and context-dependent given the group of people exposed and the occurrence's frequency, time, and location. We kept this understanding in mind throughout all of our research. Wherever possible, the generated commonsense relationships are associated with predicted uncertainty weights and context.

During our research, we understood the necessity of human involvement in

commonsense research in data annotation and evaluation steps. However, this essential human involvement comes at a relatively high monetary and time cost. The methods introduced in Chapters 3, 4 and 5 alleviate the time and cost complexities by introducing autogeneration methodologies, which require minimal new human annotations. In cases where we need extensive human annotations, we introduced a prompting technique to reduce the data annotation required in visual commonsense tasks (Chapter 6).

Recent literature shows novel capabilities emerge with scale in language models [114]. Some of these capabilities, such as commonsensical understanding, are non-existent or weak in smaller language models. Given this insight, we utilized large language models to expand vision-based commonsense knowledge graphs. These experiments showed that large language models have weaknesses in commonsense understanding, despite the excellent improvements. Specifically, we focused on commonsensical questions with negated forms of verbs and coined the term *negated complementary* questions. Although these commonsense questions can be trivial for humans, they can throw the models off guard. We introduced a methodology to improve large language models in these scenarios with minimal computation overhead (Chapter 7).

8.2 Future Work

Given our research and understanding, we envision larger models, such as large language models, to play an essential role in our collective future. Therefore, it is vital to understand better and mitigate the shortcomings. Commonsense is not limited to knowledge and thus should be evaluated in reasoning and action levels. In edge scenarios, such as in-field limited-connection automated robots, usage of on-device large models have hardware limitations. Therefore, it is also important to think of methodologies to expand the commonsensical abilities of smaller models. On another point, although new capabilities emerge with scaling up language models, they are inherently limited to the text they are trained on. Therefore, a human-level commonsense understanding of the world requires conjunction with more data modalities, such as vision.

References

[1] N. Rezaei, M. Z. Reformat, and R. R. Yager, "Image-Based World-perceiving Knowledge Graph (WpKG) with Imprecision," in *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Springer International Publishing, 2020, pp. 415–428, ISBN: 978-3-030-50146-4.

[2] ——, "Generating contextual weighted commonsense knowledge graphs," in *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Cham: Springer International Publishing, 2022, pp. 593–606, ISBN: 978-3-031-08971-8.

[3] N. Rezaei and M. Z. Reformat, "Utilizing language models to expand vision-based commonsense knowledge graphs," *Symmetry*, vol. 14, no. 8, 2022, ISSN: 2073-8994. DOI: 10.3390/sym14081715. [Online]. Available: https://www.mdpi.com/2073-8994/14/8/1715.

[4] ——, "Super-prompting: Utilizing model-independent contextual data to reduce data annotation required in visual commonsense tasks," *arXiv preprint arXiv:2204.11922*, 2022.

[5] ——, "Negated complementary commonsense using large language models," *ACL [submitted - under anonymity period]*, 2023.

[6] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[7] Y. LeCun, "On a vision to make AI systems learn and reason like animals and humans," *Meta AI Inside the Lab*, 2022. [Online]. Available: https://ai.facebook.com/blog/yann-lecun-advances-in-ai-research/.

[8] J. McCarthy and V. Lifschitz, *Formalizing Common Sense: Papers*, ser. Ablex series in artificial intelligence. New York City, New York: Ablex Publishing Corporation, 1990, ISBN: 9780893915353. [Online]. Available: https://books.google.com/books?id=V2M%5C_0jH-1lcC.

[9] R. Speer, J. Chin, and C. Havasi, "ConceptNet 5.5: An Open Multilingual Graph of General Knowledge," in *AAAI Conference on Artificial Intelligence*, vol. 31, San Francisco, California: AAAI Press, Feb. 2017, pp. 4444–4451. DOI: 10.1609/aaai.v31i1.11164. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11164.

[10] M. Sap, R. Le Bras, E. Allaway, C. Bhagavatula, N. Lourie, H. Rashkin, B. Roof, N. A. Smith, and Y. Choi, "ATOMIC: An atlas of machine commonsense for if-then reasoning," in *Proc. of the Conf. on AI (AAAI-19).*, vol. 33, 2019, pp. 3027–3035.

[11] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, "Palm: Scaling language modeling with pathways," *CoRR*, vol. abs/2204.02311, 2022. [Online]. Available: `https://doi.org/10.48550/arXiv.2204.02311`.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc., 2017.

[13] M. Sap, R. L. Bras, E. Allaway, C. Bhagavatula, N. Lourie, H. Rashkin, B. Roof, N. A. Smith, and Y. Choi, "ATOMIC: An Atlas of Machine Commonsense for If-Then Reasoning," en, *AAAI*, pp. 3027–3035, 2019, ISSN: 2374-3468. (visited on 02/15/2020).

[14] A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Celikyilmaz, and Y. Choi, "COMET: Commonsense transformers for automatic knowledge graph construction," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 4762–4779. DOI: `10.18653/v1/P19-1470`. [Online]. Available: `https://www.aclweb.org/anthology/P19-1470`.

[15] Y. Hao, Y. Zhang, K. Liu, S. He, Z. Liu, H. Wu, and J. Zhao, "An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge," in *55th ACL Meeting*, 2017, pp. 221–231.

[16] P. Kapanipathi, V. Thost, S. S. Patel, S. Whitehead, I. Abdelaziz, A. Balakrishnan, M. Chang, K. Fadnis, C. Gunasekara, B. Makni, N. Mattei, K. Talamadupula, and A. Fokoue, "Infusing knowledge into the textual entailment task using graph convolutional networks," in *AAAI*, 2020.

[17] J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, and D. L. Lee, "Billion-scale Commodity Embedding for E-commerce Recommend. in Alibaba," in *KDD '18*, 2018.

[18] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *The Semantic Web*, 2007, pp. 722–735, ISBN: 978-3-540-76298-0.

[19] D. Vrandečić and M. Krötzsch, "Wikidata: A free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014, ISSN: 0001-0782. (visited on 02/14/2020).

[20] T. Rebele, F. Suchanek, J. Hoffart, J. Biega, E. Kuzey, and G. Weikum, "YAGO: A Multilingual Knowledge Base from Wikipedia, Wordnet, and Geonames," en, in *Int. Semantic Web Conf.*, P. Groth, E. Simperl, A. Gray, M. Sabou, M. Krötzsch, F. Lecue, F. Flöck, and Y. Gil, Eds., 2016, pp. 177–185, ISBN: 978-3-319-46547-0.

[21] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *ACM SIGMOD*, 2008, pp. 1247–1250, ISBN: 978-1-60558-102-6. (visited on 02/14/2020).

[22] G. A. Miller, "WordNet: A lexical database for English," *Communication of the ACM*, vol. 38, pp. 39–41, 1995, ISSN: 0001-0782. (visited on 01/21/2020).

[23] "The Importance of Vision," English, *Ophthalmology*, vol. 94, pp. 9–13, 1987, ISSN: 0161-6420, 1549-4713. (visited on 02/15/2020).

[24] H. Jacobson, "The Informational Capacity of the Human Eye," *Science*, vol. 113, no. 2933, pp. 292–293, 1951, ISSN: 0036-8075.

[25] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *International Journal of Computer Vision*, vol. 123, pp. 32–73, 2016.

[26] S. Benferhat, D. Dubois, L. Garcia, and H. Prade, "Possibilistic logic bases and possibilistic graphs," in *15th Conf. on Uncertainty in AI*, 1999, pp. 57–64.

[27] J. Gebhardt and R. R. Kruse, "Automated construction of possibilistic networks from data," *J. of Applied Mathematics and Computer Science*, vol. 6, no. 3, pp. 101–136, 1996.

[28] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling, "Never-ending learning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015.

[29] A. Fader, S. Soderland, and O. Etzioni, "Identifying Relations for Open Information Extraction," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, UK.: Association for Computational Linguistics, 2011, pp. 1535–1545. (visited on 02/15/2020).

[30] J. Romero, S. Razniewski, K. Pal, J. Z. Pan, A. Sakhadeo, and G. Weikum, "Commonsense properties from query logs and question answering forums," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, Beijing, China: Association for Computing Machinery, 2019, pp. 1411–1420, ISBN: 9781450369763.

[31] N. Tandon, G. d. Melo, F. M. Suchanek, and G. Weikum, "WebChild: Harvesting and organizing commonsense knowledge from the web," in *WSDM '14*, 2014.

[32] N. Tandon, G. de Melo, and G. Weikum, "WebChild 2.0 : Fine-grained commonsense knowledge distillation," in *Proceedings of ACL 2017, System Demonstrations*, Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 115–120.

[33] S. K. Divvala, A. Farhadi, and C. Guestrin, "Learning Everything about Anything: Webly-Supervised Visual Concept Learning," in *2014 IEEE Conf. on Computer Vision and Pattern Recognition*, 2014, pp. 3270–3277.

[34] X. Chen, A. Shrivastava, and A. Gupta, "NEIL: Extracting Visual Knowledge from Web Data," in *2013 IEEE International Conference on Computer Vision*, 2013, pp. 1409–1416. DOI: 10.1109/ICCV.2013.178.

[35] S. Benferhat and C. Sossai, "Merging uncertain knowledge bases in a possibilistic logic framework," in *14th Conf. on Uncertainty in AI*, 1998, pp. 8–15.

[36] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, 2015.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[39] R. Girshick, "Fast R-CNN," in *IEEE Conf. on Computer Vision*, 2015, pp. 1440–1448.

[40] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[41] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene Graph Generation by Iterative Message Passing," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2017, pp. 3097–3106.

[42] R. Zellers, M. Yatskar, S. Thomson, and Y. Choi, "Neural motifs: Scene graph parsing with global context," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2018, pp. 5831–5840. DOI: 10.1109/CVPR.2018.00611. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00611.

[43] J. Zhang, K. J. Shih, A. Elgammal, A. Tao, and B. Catanzaro, "Graphical Contrastive Losses for Scene Graph Parsing," 2019, pp. 11 535–11 543. (visited on 01/16/2020).

[44] Y. Li, W. Ouyang, B. Zhou, J. Shi, C. Zhang, and X. Wang, "Factorizable Net: An Efficient Subgraph-based Framework for Scene Graph Gen.," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 335–351. (visited on 01/16/2020).

[45] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. Bernstein, and L. Fei-Fei, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *Int. J. of Computer Vision*, vol. 123, pp. 32–73, 2016.

[46] J. Gu, H. Zhao, Z. Lin, S. Li, J. Cai, and M. Ling, "Scene Graph Generation With External Knowledge and Image Reconstruction," in *IEEE/CVF Conf on Computer Vision and Pattern Recognition*, 2019, pp. 1969–1978.

[47] Neo4j, Inc., *Neo4j*, version 3.5.6, May 24, 2019. [Online]. Available: https://neo4j.com.

[48] J. D. Hwang, C. Bhagavatula, R. Le Bras, J. Da, K. Sakaguchi, A. Bosselut, and Y. Choi, "(COMET-)ATOMIC 2020: On symbolic and neural commonsense knowledge graphs," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 7, pp. 6384–6392, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16792.

[49] P. West, C. Bhagavatula, J. Hessel, J. D. Hwang, L. Jiang, R. L. Bras, X. Lu, S. Welleck, and Y. Choi, "Symbolic knowledge distillation: From general language models to commonsense models," *arXiv preprint arXiv:2110.07178*, 2021.

[50] H. Zhang, D. Khashabi, Y. Song, and D. Roth, "TransOMCS: From Linguistic Graphs to Commonsense Knowledge," in *Proc. of Inter. Joint Conf. on AI (IJCAI) 2020*, 2020.

[51] J. Chamorro-Martínez, N. Marín, M. Mengíbar-Rodríguez, G. Rivas-Gervilla, and D. Sánchez, "Referring expression generation from images via deep learning object extraction and fuzzy graphs," *2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–6, 2021.

[52] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, 2014.

[53] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 5987–5995, 2017.

[54] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[55] K. Tang, Y. Niu, J. Huang, J. Shi, and H. Zhang, "Unbiased scene graph generation from biased training," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3713–3722, Jun. 2020. DOI: 10.1109/CVPR42600.2020.00377. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR42600.2020.00377.

[56] R. Zellers, Y. Bisk, A. Farhadi, and Y. Choi, "From recognition to cognition: Visual commonsense reasoning," in *The IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[57] A. F. Hayes and K. Krippendorff, "Answering the call for a standard reliability measure for coding data," *Communication Methods and Measures*, vol. 1, pp. 77–89, 2007.

[58] D. Dubois, H. Prade, and P. Smets, "New semantics for quantitative possibility theory," in *2nd Inter. Symposium on Imprecise Probabilities and Their Applications*, 2001.

[59] H. Prade, "Data bases with fuzzy information and approximate reasoning in expert systems," in *IFAC Artificial Intelligence*, 1983, pp. 113–119.

[60] D. Dubois and H. Prade, *Possibility Theory*. Plenum Press, 1988.

[61] P. West, C. Bhagavatula, J. Hessel, J. D. Hwang, L. Jiang, R. L. Bras, X. Lu, S. Welleck, and Y. Choi, "Symbolic knowledge distillation: From general language models to commonsense models," *arXiv preprint arXiv:2110.07178*, 2021.

[62] Y. LeCun, "A path towards autonomous machine intelligence version 0.9.2," 2022. [Online]. Available: https://openreview.net/forum?id=BZ5a1r-kVsf.

[63] Y. Choi, "The Curious Case of Commonsense Intelligence," *Daedalus*, vol. 151, no. 2, pp. 139–155, May 2022, ISSN: 0011-5266. DOI: 10.1162/daed_a_01906. [Online]. Available: https://doi.org/10.1162/daed%5C_a%5C_01906.

[64] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[65] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. DOI: 10.18653/v1/D16-1264. [Online]. Available: https://aclanthology.org/D16-1264.

[66] N. Rezaei, M. Z. Reformat, and R. R. Yager, "Image-based world-perceiving knowledge graph (wpkg) with imprecision," *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, vol. 1237, pp. 415–428, 2020.

[67] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating Embeddings for Modeling Multi-relational Data," in *Advances in Neural Information Processing Systems*, C. J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26, Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf.

[68] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6575.

[69] C. Wang, X. Liu, and D. X. Song, "Language models are open knowledge graphs," *ArXiv*, vol. abs/2010.11967, 2020.

[70] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. Miller, "Language models as knowledge bases?" In *2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 2463–2473. DOI: 10.18653/v1/D19-1250. [Online]. Available: https://aclanthology.org/D19-1250.

[71] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. [Online]. Available: https://aclanthology.org/N19-1423.

[72] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[73] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, *et al.*, "Opt: Open pre-trained transformer language models," *arXiv preprint arXiv:2205.01068*, 2022.

[74] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou, "Chain of thought prompting elicits reasoning in large language models," *arXiv preprint arXiv:2201.11903*, 2022. [Online]. Available: https://arxiv.org/abs/2201.11903.

[75] T. Khot, A. Sabharwal, and P. Clark, "What's missing: A knowledge gap guided approach for multi-hop question answering," in *2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 2814–2828. DOI: 10.18653/v1/D19-1281. [Online]. Available: https://aclanthology.org/D19-1281.

[76] A. Fabbri, P. Ng, Z. Wang, R. Nallapati, and B. Xiang, "Template-based question generation from retrieved sentences for improved unsupervised question answering," in *58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computa-

tional Linguistics, Jul. 2020, pp. 4508–4513. DOI: 10.18653/v1/2020.acl-main.413. [Online]. Available: https://aclanthology.org/2020.acl-main.413.

[77] S. Jastrzębski, D. Bahdanau, S. Hosseini, M. Noukhovitch, Y. Bengio, and J. Cheung, "Commonsense mining as knowledge base completion? a study on the impact of novelty," in *Workshop on Generalization in the Age of Deep Learning*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 8–16. DOI: 10.18653/v1/W18-1002. [Online]. Available: https://aclanthology.org/W18-1002.

[78] J. Davison, J. Feldman, and A. Rush, "Commonsense knowledge mining from pretrained models," in *2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 1173–1178. DOI: 10.18653/v1/D19-1109. [Online]. Available: https://aclanthology.org/D19-1109.

[79] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, Hawaii: IEEE, 2017, pp. 5987–5995. DOI: 10.1109/CVPR.2017.634.

[80] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *International Conference on Learning Representations*, Virtual Conference: ICLR, 2020.

[81] X. Li, A. Taheri, L. Tu, and K. Gimpel, "Commonsense knowledge base completion," in *54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1445–1455. DOI: 10.18653/v1/P16-1137. [Online]. Available: https://aclanthology.org/P16-1137.

[82] H. Zhang, D. Khashabi, Y. Song, and D. Roth, "TransOMCS: From Linguistic Graphs to Commonsense Knowledge," in *Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, International Joint Conferences on Artificial Intelligence Organization, Jul. 2020, pp. 4004–4010. DOI: 10.24963/ijcai.2020/554. [Online]. Available: https://doi.org/10.24963/ijcai.2020/554.

[83] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," *arXiv preprint arXiv:2203.11171*, 2022.

[84] J. S. Park, C. Bhagavatula, R. Mottaghi, A. Farhadi, and Y. Choi, "Visualcomet: Reasoning about the dynamic context of a still image," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., Cham: Springer International Publishing, 2020,

pp. 508–524, ISBN: 978-3-030-58558-7. DOI: https://doi.org/10.1007/978-3-030-58558-7_30.

[85] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[86] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "SuperGLUE: A stickier benchmark for general-purpose language understanding systems," in *Advances in neural information processing systems*, 2019, pp. 3266–3280.

[87] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html.

[88] P. He, X. Liu, J. Gao, and W. Chen, *DeBERTa: Decoding-enhanced BERT with disentangled attention*, 2020. arXiv: 2006.03654 [cs.CL].

[89] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *ACL*, 2018.

[90] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. A. Smith, "Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping," *ArXiv*, vol. abs/2002.06305, 2020.

[91] C. Lee, K. Cho, and W. Kang, "Mixout: Effective regularization to finetune large-scale pretrained language models," *ArXiv*, vol. abs/1909.11299, 2020.

[92] T. Zhang, F. Wu, A. Katiyar, K. Q. Weinberger, and Y. Artzi, "Revisiting few-sample bert fine-tuning," *ArXiv*, vol. abs/2006.05987, 2020.

[93] T. Schick and H. Schütze, *Exploiting cloze questions for few shot text classification and natural language inference*, 2020. arXiv: 2001.07676 [cs.CL].

[94] ——, *It's not just size that matters: Small language models are also few-shot learners*, 2020. arXiv: 2009.07118 [cs.CL].

[95] T. Gao, A. Fisch, and D. Chen, *Making pre-trained language models better few-shot learners*, 2020. arXiv: 2012.15723 [cs.CL].

[96] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *ICML*, 2019.

[97] J. Pfeiffer, A. Rücklé, C. Poth, A. Kamath, I. Vulić, S. Ruder, K. Cho, and I. Gurevych, "Adapterhub: A framework for adapting transformers," in *EMNLP*, 2020.

[98] X. L. Li and P. Liang, *Prefix-tuning: Optimizing continuous prompts for generation*, 2021. arXiv: 2101.00190 [cs.CL].

[99] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.703.

[100] N. Mostafazadeh, A. Kalyanpur, L. Moon, D. Buchanan, L. Berkowitz, O. Biran, and J. Chu-Carroll, "GLUCOSE: GeneraLized and COntextualized story explanations," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 4569–4586. DOI: 10.18653/v1/2020.emnlp-main.370. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-main.370.

[101] R. Zellers, Y. Bisk, A. Farhadi, and Y. Choi, "From recognition to cognition: Visual commonsense reasoning," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6713–6724, 2019.

[102] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. [Online]. Available: https://www.aclweb.org/anthology/P02-1040.

[103] M. Denkowski and A. Lavie, "Meteor universal: Language specific translation evaluation for any target language," in *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, Maryland, USA: Association for Computational Linguistics, Jun. 2014, pp. 376–380. DOI: 10.3115/v1/W14-3348. [Online]. Available: https://www.aclweb.org/anthology/W14-3348.

[104] R. Vedantam, C. L. Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4566–4575. DOI: 10.1109/CVPR.2015.7299087.

[105] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016. DOI: 10.1109/cvpr.2016.90. [Online]. Available: http://dx.doi.org/10.1109/cvpr.2016.90.

[106] I. J. Goodfellow, D. Erhan, P. L. Carrier, A. C. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee, Y. Zhou, C. Ramaiah, F. Feng, R. Li, X. Wang, D. Athanasakis, J. Shawe-Taylor, M. Milakov, J. Park, R. T. Ionescu, M. Popescu, C. Grozea, J. Bergstra, J. Xie, L. Romaszko, B. Xu, C. Zhang, and Y. Bengio, "Challenges in representation learning: A report on three machine learning contests," *Neural networks : the official journal of the International Neural Network Society*, vol. 64, pp. 59–63, 2015.

[107] M. Cornia, M. Stefanini, L. Baraldi, and R. Cucchiara, "Meshed-Memory Transformer for Image Captioning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[108] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. [Online]. Available: https://aclanthology.org/N19-1423.

[109] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, H. F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, E. Rutherford, T. Hennigan, J. Menick, A. Cassirer, R. Powell, G. van den Driessche, L. A. Hendricks, M. Rauh, P.-S. Huang, A. Glaese, J. Welbl, S. Dathathri, S. Huang, J. Uesato, J. Mellor, I. Higgins, A. Creswell, N. McAleese, A. Wu, E. Elsen, S. M. Jayakumar, E. Buchatskaya, D. Budden, E. Sutherland, K. Simonyan, M. Paganini, L. Sifre, L. Martens, X. L. Li, A. Kuncoro, A. Nematzadeh, E. Gribovskaya, D. Donato, A. Lazaridou, A. Mensch, J.-B. Lespiau, M. Tsimpoukelli, N. Grigorev, D. Fritz, T. Sottiaux, M. Pajarskas, T. Pohlen, Z. Gong, D. Toyama, C. de Masson d'Autume, Y. Li, T. Terzi, V. Mikulik, I. Babuschkin, A. Clark, D. de Las Casas, A. Guy, C. Jones, J. Bradbury, M. Johnson, B. A. Hechtman, L. Weidinger, I. Gabriel, W. S. Isaac, E. Lockhart, S. Osindero, L. Rimell, C. Dyer, O. Vinyals, K. Ayoub, J. Stanway, L. Bennett, D. Hassabis, K. Kavukcuoglu, and G. Irving, "Scaling language models: Methods, analysis and insights from training gopher," *CoRR*,

vol. abs/2112.11446, 2021. [Online]. Available: https://arxiv.org/abs/2112.11446.

[110] P. West, C. Bhagavatula, J. Hessel, J. Hwang, L. Jiang, R. Le Bras, X. Lu, S. Welleck, and Y. Choi, "Symbolic knowledge distillation: From general language models to commonsense models," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 4602–4625. DOI: 10.18653/v1/2022.naacl-main.341. [Online]. Available: https://aclanthology.org/2022.naacl-main.341.

[111] J. Da, R. L. Bras, X. Lu, Y. Choi, and A. Bosselut, "Analyzing commonsense emergence in few-shot knowledge models," in *3rd Conference on Automated Knowledge Base Construction*, 2021. [Online]. Available: https://openreview.net/forum?id=StHCELh9PVE.

[112] L. Jiang, A. Bosselut, C. Bhagavatula, and Y. Choi, ""I'm not mad": Commonsense implications of negation and contradiction," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online: Association for Computational Linguistics, Jun. 2021, pp. 4380–4397. DOI: 10.18653/v1/2021.naacl-main.346. [Online]. Available: https://aclanthology.org/2021.naacl-main.346.

[113] S. Kadavath, T. Conerly, A. Askell, T. J. Henighan, D. Drain, E. Perez, N. Schiefer, Z. Dodds, N. DasSarma, E. Tran-Johnson, S. Johnston, S. El-Showk, A. Jones, N. Elhage, T. Hume, A. Chen, Y. Bai, S. Bowman, S. Fort, D. Ganguli, D. Hernandez, J. Jacobson, J. Kernion, S. Kravec, L. Lovitt, K. Ndousse, C. Olsson, S. Ringer, D. Amodei, T. B. Brown, J. Clark, N. Joseph, B. Mann, S. McCandlish, C. Olah, and J. Kaplan, "Language models (mostly) know what they know," *ArXiv*, vol. abs/2207.05221, 2022. DOI: 10.48550/ARXIV.2207.05221. [Online]. Available: https://arxiv.org/abs/2207.05221.

[114] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus, "Emergent abilities of large language models," *Transactions on Machine Learning Research*, 2022, Survey Certification. [Online]. Available: https://openreview.net/forum?id=yzkSU5zdwD.