CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF INFORMATION TECHNOLOGY

Human Alignment of Natural Language Processing Models

MASTER'S THESIS

2024 Anastasiia Solomiia HRYTSYNA

FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

Assignment of master's thesis

Title:	Human Alignment of Natural Language Processing Models
Student:	Bc. Anastasiia Solomiia Hrytsyna
Supervisor:	Rodrigo Augusto da Silva Alves, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2024/2025

Instructions

A key goal in the cognitive sciences is the development of mathematical models for mental representations of object concepts. In this context, a large triplet-based dataset (named 'Things') on object concepts was created, composed of triplets of pictures presented to human subjects who judged which one is the most different. For instance, a triplet can be considered with the following three figures: (A) a dog; (B) a cat; and (C) a bus. This triplet is presented to a human subject who should answer, according to their opinion, which of these three pictures is the odd one when considering the group (e.g, (C) Bus). With a large dataset of these experiments, mathematical models were developed to better understand how humans form concepts. This dataset was also previously utilized to study the human alignments of pretrained computer vision models. However, less attention has been given to natural language processing (NLP) models. In this thesis, the main objective is to analyze how pretrained NLP models align with human's object concepts based on the 'Things' dataset. The following tasks are expected:

1) Captioning the images of things based on at least three captioning-based NLP models.
2) Conducting a literature review to outline at least fifteen general use pre-trained NLP models (e.g., Bert) that form a representative set of the state-of-the-art.
3) Extracting feature vectors from the last layers of the NLP models, by using the captions of (1) as inputs and (2) as pre-trained models.
4) Comparing the feature vectors (using at least two metrics) to classify tripplets, considering the 'Things' triplets dataset in a zero-shot learning fashion.
5) Comprehensively discuss the results of the thesis involves verifying possible

---

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 16 January 2024 in Prague.

interpretable aspects and how different classes of NLP models align with 'Things' concepts.

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 16 January 2024 in Prague.

Master's thesis

HUMAN ALIGNMENT OF NATURAL LANGUAGE PROCESSING MODELS

Bc. Anastasiia Solomiia Hrytsyna

Faculty of Information Technology
Department of Applied Mathematics
Supervisor: Rodrigo Augusto da Silva Alves Ph.D
June 27, 2024

Contents

List of Figures

List of Tables

I would like to express my sincere gratitude to my master thesis supervisor, Rodrigo Augusto da Silva Alves, Ph.D. His professional guidance, support and expertise have been invaluable throughout this journey. Dr. Alves' insightful feedback, weekly consultations, encouragement, constant positive approach and unwavering commitment to excellence have greatly enriched my research experience and contributed to the successful completion of both thesis and article. I am deeply grateful for his PATIENCE and dedication to my academic and personal growth. I can not even imagine a better mentorship from the one I got.

I am also grateful to my partner for his constant support throughout this journey, for his encouragement and for believing in me. I hope that his thesis process will be as smooth as mine, and that in six months we can sit together and laugh at all the absurd struggles and nights without sleeping we went through. I also hope that one day he will forget this trauma and continue on his way to a PhD... And finally thanks me for writing this acknowledgment for him, because he asked me to.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Czech Technical University in Prague has the right to conclude a licence agreement on the utilization of this thesis as a school work pursuant of Section 60 (1) of the Act.

In Prague on June 27, 2024

Abstract

This study proposes a strategy for evaluating the performance of Large Language Models (LLMs) in comparison to natural human behavior. A comprehensive dataset of basic everyday concepts paired with relevant images was employed. Each image was annotated using three captioning methods: BLIP, ViT, and ChatGPT-4V. In subsequent stages, the generated captions, along with the original image names, were processed through 24 state-of-the-art natural language processing models from eight distinct families. Vector representations from the last five layers of these models were extracted and compared using two scoring metrics. These embeddings were then utilized to solve an odd-one-out task on a complex triplet dataset. The evaluation results indicate that accurately mimicking human perception remains challenging for most models. However, models from the GPT-2 and BERT families achieved an approximate accuracy of $40 - 50\%$ in analyzing complex triplets. Furthermore, the findings suggest that longer and more detailed input texts facilitate more human-like responses from the models. Despite these advancements, there remains significant scope for improving LLMs to ensure their applicability across diverse environments and better understanding of nuanced concepts without the need for extensive adjustments.

Keywords Large Language Models, Human Concept Representations, Odd-one-out Problem, Zero-shot Learning, Image Captioning, Explainable Artificial Intelligence

Abstrakt

Tato studie navrhuje strategii pro hodnocení výkonu velkých jazykových modelů ve srovnání s přirozeným lidským chováním. Byl použit rozsáhlý dataset základních každodenních konceptů spárovaných s relevantními obrázky. Každý obrázek byl anotován pomocí tří metod popisu: BLIP, ViT a ChatGPT-4V. V následujících krocích byly vygenerované popisky spolu s původními názvy obrázků zpracovány prostřednictvím 24 state-of-the-art modelů na zpracování přirozeného jazyka z osmi různých rodin. Byly extrahovány vektorové reprezentace z posledních pěti vrstev těchto modelů a porovnány pomocí dvou hodnotících metrik. Tyto vektory byly následně využity k řešení úlohy "najdi vetřelce" na obsáhlém datasetu trojic obrázků. Výsledky hodnocení ukazují, že přesné napodobení lidského vnímání zůstává pro většinu modelů obtížné. Nicméně modely z rodin GPT-2 a BERT dosáhly přibližné přesnosti $40 - 50\%$ při analýze trojic. Zjištění rovněž naznačují, že delší a podrobnější vstupní texty usnadňují modelům vytvářet odpovědi podobnější těm lidským. Navzdory těmto pokrokům zůstává značný prostor pro zlepšení LLM, aby byla zajištěna jejich použitelnost v různých prostředích a lepší pochopení základních konceptů bez nutnosti rozsáhlých úprav.

Klíčová slova Velké Jazykové Modely, Reprezentace Lidských Konceptů, Problém Vyčnívajícího Prvku, Učení bez Vzorku, Titulkování Obrázku, Pochopitelná Umělá Iteligence

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
BART	Bidirectional Autoregressive Transformer
BERT	Bidirectional Encoder Representations from Transformers
BLIP	Bootstrapping Language-Image Pre-training Model
DL	Deep Learning
FFN	Feedforward Neural Network
GPT	Generative Pre-trained Transformer
LLM	Large Language Model
ML	Machine Learning
NLP	Natural Language Processing
NN	Neural Networks
RNN	Recurrent Neural Network
ViT	Vision Transformer
ZSL	Zero-shot learning

Chapter 1
INTRODUCTION

1.1 Motivation

Understanding and processing human language remains a hard task that interested people for more than a century. The early history of this pursuit can be traced back to the mid-20th century, marked by the initial attempts to decode information and establish the foundational theories of language perception [1]. By the 1960s and 1970s, a rule-based approach dominated language understanding endeavors. Several systems that relies on sets of linguistic rules were created (for instance ELIZA - the first program using natural language processing (NLP) [2]), but, unfortunately, they were limited by the complexity of natural language variations and struggled to encompass all linguistic nuances effectively.

Subsequently, during the following years up to 1980, only minor advances were achieved, leading to a period often referred to as "the first AI winter". After some time the concept of statistical language models emerged as a prominent paradigm. These type of models employed various statistical methodologies to analyze extensive amount of not-standardized input textual data, calculate how often certain words appear within this text and make some predictions about the following potential word [3]. While representing an advancement over rule-based systems, statistical models still had multiple limitations in accurately capturing context and semantics.

The next important breakthrough in language processing occurred with the rise of neural network-based methodologies, particularly with the evolution of deep learning techniques in the 2010s. This era faced the transforming impact of recurrent neural networks (RNNs) on NLP [4]. RNNs completely revolutionized the field by providing simplified processes for interpreting contextual information and identifying semantic relationships within the text. Their ability to dynamically capture dependencies over sequences of data fundamentally changed the landscape of NLP, opening up new horizons for advanced language understanding and generation.

Since then, the computational capabilities of computers raised, empowering researchers to perform experiments with larger and more complex neural network architectures trained on massive datasets. Among these advancements, the introduction of the transformer architecture [5, 6] stands out prominently. Soon after its inception, transformers swiftly replaced recurrent neural networks for solving variety of NLP tasks due to their

ability to capture long-range dependencies more efficiently.

In contemporary times, Large Language Models (LLMs) have emerged as ubiquitous instruments in the realm of NLP-based artificial intelligence, being an important tool to researchers, specialists and practitioners alike [7]. Due to their capacity to comprehend (and often produce) text analogously to human language, LLMs have attracted considerable interest and are utilized across a broad spectrum of tasks and applications. It is important to notice how they have transformed the field of AI and language processing, by being widely used not only for natural language understanding purposes, but also for machine translation generation, text summarization, question answering and beyond [8]. Their flexibility and ability to adjust gives them invaluable possibilities for coping with linguistic obstacles, overcoming traditional boundaries and fostering innovation across various fields. Therefore, recent advancements in LLMs have raised the field of NLP to new heights [9, 7].

Most successful LLMs are based on transformers. For instance, the BERT model (Bidirectional Encoder Representations from Transformers) [10, 11, 12] has the ability to adapt and solve a variety of problems. Additionally, in recent years models like GPT (Generative Pre-trained Transformer) by OpenAI [13, 14, 15] and BART (Bidirectional and Auto-Regressive Transformers) by Google [16] appealed. They democratized access to advanced language processing capabilities and gave an opportunity to ordinary people to interact and actively use these technologies with Chat GPT [17] or Gemini [18]. LLMs changed our everyday life from using ordinary search engines and virtual assistants like Siri or Alexa to automated content creation tools.

Still, even in the present day, Large Language Models often remain 'black boxes' [19]. Complete understanding of their inner processes and architecture is challenging, which makes it difficult to predict or control their precise outputs [20, 21, 22]. Consequently, it becomes crucial to extensively monitor and examine their performance to ensure their reliability, effectiveness and mitigate potential risks associated with their deployment. Additionally, assessing how closely their outputs resemble human expressions and generation is essential for evaluating their utility and potential usage across various applications [23, 24].

Therefore, the primary focus of this research is to understand how representations of objects by black-box-based LLMs align with human concept representations. Previous related studies have employed diverse methodologies to the task of understanding LLM's black boxes mechanisms, such as perturbing input examples [25, 26], employing probing strategies [27, 28], and utilizing surrogate models with simplified representations [29]. Our alignment strategy significantly deviates from these approaches. This thesis builds on previous research in the realm of aligning human and neural representations [30]. However, instead of focusing on computer vision, this study explores the natural language processing domain. Our primary aim in our experiment setup is to compare the outcomes of solving triplet-based odd-one-out tasks between human participants and artificial intelligence approaches. This problem involves presenting three different items and asking participants to identify the one they consider most different, according to their own judgment, or equivalently, the two most similar. This task is based on pattern recognition and logical reasoning and can be used to assess cognitive skills such as categorization and problem-solving and it's a important and well-studied research problem in the field of cognitive sciences.

In our experiment, for a comprehensive comparison, we utilized the THINGS dataset [31], which includes a diverse range of triplets based on a variety of object images. Our experimental pipeline can be briefly described as follows: the images were processed using state-of-the-art image captioning models, and the resulting captions were then fed into language models. Subsequently, we extracted the embeddings from these large language models (LLMs) to perform the odd-one-out task using only these embeddings in a zero-shot learning fashion. Ultimately, the image deemed most dissimilar was compared with the human response to identify potential areas of disparity or conceptual challenges. The overarching objective of this study was to highlight differences in how LLMs represent concepts, with the aim of facilitating future improvements and enhancing user awareness.

1.2 Assignment and Contributions

This master's thesis fulfills all the objectives of its proposed assignment.

The main contributions of this thesis and their relationship to the objectives are as follows:

1. In this thesis, we propose a framework to analyze the alignment of human object concepts with the representations in the latent space of LLMs. Our pipeline is described in Chapter 4.2 and encompasses captioning, extracting feature vectors of LLMs (items 1, 3, and 4 of the original assignment);
2. We analyze our pipeline with four captioning methods and extract embeddings from twenty-four different LLMs (items 1, 2, and 3 of the assignment). Results can be seen in Chapter 4.3 and 4.4;
3. We compare the embeddings using Cosine and Euclidean distances in a zero-shot learning approach, using the THINGS dataset (assignment item 4).
4. We discuss our results by comparing captioning models and LLMs regarding the alignment in Chapter 5;
5. A theoretical background and state-of-the-art approaches are available in Chapter 2;

Part of the results of this thesis were submitted to the ACM Transactions on Intelligent Systems and Technology (SCOPUS indexed). It is currently under review.

Anastasiia Hrytsyna and Rodrigo Alves. From Representation to Comprehension: Aligning Large Language Models with Human Object Understanding. **Under review in:** *ACM Transactions on Intelligent Systems and Technology*, 2024.

1.3 Thesis Organisation

The thesis organization is divided into few central chapters that effectively guide reader through the various aspects of the research, from foundational concepts to experimental findings and conclusions. Below is a compilation of these sections:

1. **INTRODUCTION**

   This section provides an overview of the thesis topic, starting with an introduction to Large Language Models (LLMs) and offering a brief historical background to contextualize their development and significance.

2. **THEORETICAL FOUNDATION**

   This section delves into the core concepts relevant to the research, including Odd-one-out Problem, Zero-shot Learning, and Image Captioning. It explores different types of image captioning, their applications and associated challenges. Another significant subsection focuses on LLMs (Large Language Models), covering various aspects such as different architectures (e.g., transformers), tokenization, word embedding, and attention mechanisms. This provides readers with a comprehensive understanding of the theoretical foundations underpinning whole research.

3. **RELATED WORK**

   The following chapter, **related works**, presents a comparative analysis of a few existing approaches for image captioning, text processing using different NLP models and their evaluation in comparison to the human beings. By reviewing related work, it is easier to understand how this research contributes to the existing body of knowledge in the field.

4. **METHODOLOGY**

   This section outlines the experimental methodology employed in the thesis. It includes descriptions of the input Dataset, the Image Captioning Models used for image describing and Language Processing for language processing, and the process of extracting the last hidden state. Additionally, it explains how zero-shot learning was scored using various metrics. And, finally, it provides some insights into the Experiment Pipeline.

5. **EXPERIMENTS**

   Here, some findings of the experiment are presented. This involves comparing the outcomes of the models to human labels (Zero-shot Prediction), addressing any complications encountered during the process (e.g., imprecise image captioning - The Impact of Image Captioning), and discussing the impact of factors such as LLM type, size and architecture (The Impact of Language Model Selection) and image type (The Impact of Image Origin) on the results. You also analyze the effectiveness of zero-shot prediction in general based on the experiments.

6. **CONCLUSION**

   This section offers a Summary of your research findings and discusses potential avenues for Further Enhancements.

Chapter 2

THEORETICAL FOUNDATION

2.1 Odd-one-out Problem

Odd-one-out task is a fundamental problem often found in modern Intelligence Quotient (IQ) tests that helps to understand different relationships among the objects. The main idea of the task itself is to identify the item that does not belong among other sets of items or concepts.

Let S be a collection containing 3 elements, denoted as $S = \{x_1, x_2, x_3\}$. The goal is to identify the odd-one-out element x_o (in this study - the object concept), within the collection S based on a specified criterion or property. Mathematically, the odd-one-out problem can be represented as:

$$x_o = \arg \min_{i \in (1,2,3)} f(x_i, S)$$

where:

x_o is the odd-one-out item in the collection S.

$f(x_i, S)$ is a function that evaluates the similarity of element $x_i \in S$ to the remaining triplet elements $\{y \in S | y \neq x_i\}$. In this study, the odd-one-out is determined by minimizing this function.

The choice of function $f(x_i)$ depends on the specific context and characteristics of the elements in the set, as well as the criteria used to define the odd-one-out.

Figure 2.1 Example of triplets that can be used in the odd-one-out task. All of the images are copyright-free images from *THINGS+* dataset [32]

In our study, the challenge of the problem lies in identifying the most distinct image concept by comparing generated image captionings and analyzing them with LLMs. It is crucial to examine the difference between human visual perception and language models text understanding in terms of defining differences and similarities within the exact context. While some solutions may be straightforward, others demand considerable effort or creative thinking. For example, in Figure 2.1 is shown "Triplet 1" and it is relatively easy to identify that "Goat" is a probable distinct item because it's an animal, while the others are fruits. Yet, many cases offer less direct understanding, by presenting a higher level of subjectivity. For instance, in "Triplet 2" the odd-one-out may be "Fox" due to its fur color or "Kangaroo" for not being a local animal for Czech Republic territory. Moreover, in certain cases (as seen in "Triplet 3"), discerning dependencies can prove challenging even for human beings. This is why directly comparing model results to human labels is not a straightforward task, as the ground truth is not clearly defined. Nonetheless, observe that, different from the case of image captioning as is known, the odd-one problem from triplets problem does not have ground truth. The decision of which item is the most different depends on a series of aspects, such as culture, taste, and knowledge of the participant choosing. However, we note that the collective behavior can be studied.

In the realm of computer science, the odd-one-out problem serves as a valuable tool for testing robots and artificial intelligence [33, 34]. Solving this task often requires model training to recognize some patterns in the given input, analyze the characteristics or attributes of each item, define exact associations among a set of items and then determine the outlier that does not conform to the established pattern [35]. This can be achieved through various problem-solving approaches and methods [36, 34, 37], including natural language understanding and processing, image recognition, pattern analysis or even anomaly detection. Furthermore, a wide range of different techniques can be employed, ranging from providing a selected model with labeled examples (supervised learning), allowing the model to discover patterns on its own (unsupervised learning) or guiding it through trial and error (reinforcement learning).

2.2 Zero-shot Learning

Zero-shot learning (or zero-data learning) [38] has surged in popularity over recent decades within the machine learning sphere [39, 40]. The main idea is to train some models to recognize classes and categorize input objects that have never been encountered during the training process. This approach holds particular relevance in scenarios where acquiring labeled dataset for all potential classes is either extremely challenging, time-consuming or prohibitively expensive.

In comparison to traditional supervised learning methods that require labeled dataset for its training, zero-shot learning aims to recognize new (unseen) classes without explicit examples by leveraging semantic similarities and extracting crucial features associated with the classes. These important features could include textual descriptions, visual resemblances, semantic embeddings or other forms of structured data. Combining multiple feature extraction techniques often proves highly effective and more accurate in ZSL. Through this process, input items are mapped to the most corresponding class in a semantic space later on [40]. An illustration of zero-data learning could involve instruct-

ing a model to identify a particular animal, such as a koala, without directly exposing it to any koala images. Instead, the model receives descriptions indicating that the animal is four-legged creature, resembling a small bear and characterized by gray fluffy fur and big nose, among other features.

It is crucial to understand that ZSL still relies on labeled training data, however it does not necessitate labels for every individual class. Sufficient labeled data is required to establish patterns or contextual understanding, which can subsequently be generalized to unseen classes.

Nowadays, ZSL finds widespread application across various machine learning domains, including chat bots, search engines, multi-label classification [41], news sentiment classification [42], visual emotions recognition [43], clinical language processing [44] and numerous other tasks, capable of handling novel situations [39, 45, 46]. In our study, we chose to employ zero-data learning to address natural language processing tasks and explore the disparity between human perception of odd-one-out items among triplets and the embedding representations generated by Large Language Models.

2.3 Image Captioning

Image captioning is a complex computer vision and NLP task of generating textual descriptions for input images [47, 48]. This sophisticated computational process aims to produce coherent and informative sentences that accurately portray the visual content depicted within the image. For doing this most of the systems use encoder-decoder framework, where input image (sequence of pixels) is transformed to the set of meaningful features that are hidden in it and then this information is decoded to the descriptive text (sequence of words) (see Figure 2.2). That is why often image captioning is considered a Sequence to Sequence problem.

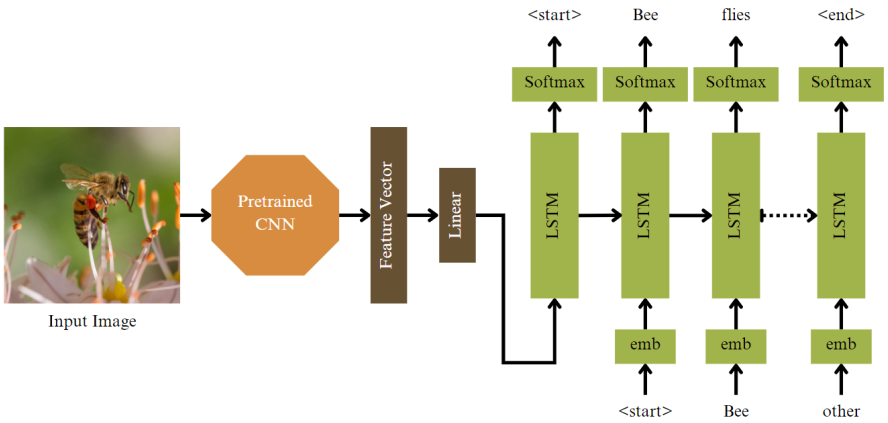


Figure 2.2 Encoder-Decoder Framework for Image Captioning Process. Input image is copyright-free image from *THINGS+* dataset [32]

2.3.1 General Concept

The whole process of creating textual descriptions for input images is crucial in terms of effective passing visual information to the next processing stages. The more exact captioning is, the more details have been noticed and recorded as a text. It may be divided into few important stages [48]:

1. **Image Understanding**

   Regrettably, none of the recurrent neural networks can directly process an RGB image tensor as input. Therefore, an input image must undergo processing by a Convolutional Neural Network (CNN) to extract essential high-level features that may be visually observed in the image's content [49]. The central component of a CNN is the Conv2d layer, which reveals concealed patterns by extracting crucial features. These features serve as foundational elements that enable the network to comprehend the content depicted in the image and encapsulate details regarding objects, colors, types, shapes, textures, and various spatial relationships present in the image. Popular CNN architectures used for this purpose include VGG-16, ResNet and Inception, each renowned for its ability to effectively capture intricate visual features.

2. **Feature Encoding**

   In the subsequent stage, all previously extracted visual features are passed to a recurrent neural network (RNN), typically employing long short-term memory (LSTM) or gated recurrent unit (GRU) cells [50]. LSTMs are well-suited for sequential data processing tasks due to their ability to retain information over long sequences and mitigate the vanishing gradient problem often encountered in traditional RNNs. Another method (GRU) is quite similar to LSTM, but with a simplified architecture, making it computationally more efficient for solving the same tasks. In general, the RNN then systematically processes these features one by one, generating a sequence of hidden states (embeddings) that encode the concealed information within the image input. These hidden states, represented as vectors, facilitate their seamless integration into subsequent phases of the process.

3. **Language Generation**

   Following the feature encoding, the caption generation takes place. It is produced by another branch of RNN that uses an initial start token and progressively crafting the complete caption word by word. At each time step, the RNN predicts the next word by using the current hidden state and previously generated words (tokens). This iterative process continues until either the end token is generated or a maximum caption length is reached.

4. **Training and Evaluation Phases**

   In the last phase all previously mentioned steps have to be combined and one final model has to be trained. During the training model tries to generate different captions and then they are compared to the original one provided for training (in this step reinforcement learning [51] may be used). The more similar captions are, the better it is.

After the training phase the model performance has to be evaluated by comparing its generated outputs to human-written annotations. For doing this, metrics for text difference analysis like BLEU (Bilingual Evaluation Understudy), METEOR (Metric for Evaluation of Translation with Explicit Ordering) or CIDEr (Consensus-based Image Description Evaluation) may be very useful [52].

2.3.2 Types of Image Captioning

There are several approaches to image captioning, each characterized by its own strengths, weaknesses or limitations [53]. Understanding these diverse methodologies is essential for navigating the landscape of image captioning research effectively. Here are some of the primary types:

- **Template-based Captioning.** This method relies on pre-established sentence templates or structures to maintain consistency in generating captions. Foundational to the field of image captioning, it defines a structured framework for ensuring coherence and uniformity in textual descriptions.
- **Deterministic Captioning.** With deterministic models, the caption generated for a particular image remains constant upon repeated analysis. This approach stands as a conventional method within the realm of image captioning models.
- **Stochastic Captioning.** In contrast to the previous method, stochastic models have the capacity to provide a variety of different captions for the same input image, operating on the basis of probabilities. This aspect represents a progressive dimension of AI image captioning, fostering dynamic interpretations and enriching the realm of artificial intelligence.
- **Attention-Based Captioning.** This method utilizes attention mechanisms to direct the model's focus to the exact regions of the image during caption generation. This dynamic adjustment enables the model to produce captions that are more contextually relevant by emphasizing relevant visual features.
- **Conceptual Captioning.** The model prioritizes the generation of captions that depict the concepts and correlations within the image, rather than merely offering a straightforward description of its visual elements.
- **Free-form Captioning.** True to its name, this approach grants a wider scope for expression with the use of modern advancements in AI, facilitating a diverse range of textual interpretations [54].

2.3.3 Application and Challenges

In today's modern world, image captioning serves as a versatile tool [55] with a multitude of applications spanning across diverse fields and industries. For instance, on digital platforms like web, social media, galleries or e-commerce platforms image description may enhance user experiences and human-computer interaction by adding more necessary insights, information or citation about the content [47, 56]. In terms of online visibility,

the incorporation of an image captioning can significantly improve a website's search engine by adding more content to the items. This augmentation makes the items more understandable and discoverable through conventional text-based searches [57].

Another possible usage includes clarifying images to assist visually impaired individuals [58]. As an illustration, NVIDIA Corporation supports numerous initiatives, where image captioning technologies are employed to assist individuals with limited or no eyesight [1].

Peering into the future, the integration of these captioning systems may be widely used within augmented (AR) and virtual reality (VR). This innovation will empower users to receive the information and descriptive captions of unknown objects just by wearing the glasses in real-time surrounding.

Furthermore, image captioning holds significant potential in the educational sector. For instance, it can facilitate content creation by automatically generating descriptions for images or educational videos (it goes by the name of closed captions [59, 60]), thereby enriching learning experiences for students. Similarly, in the medical field, captioning assists healthcare practitioners in interpreting and documenting diagnostic images, aiding in medical diagnosis and treatment planning [61].

Nonetheless, all applications of AI and image captioning struggle with certain drawbacks or hurdles. An inaccurate image description could result in diverse interpretations or even misconceptions. Additionally, training data may sometimes bring biases and impact final outcomes. And last but not least, real-time image captioning remains a challenging task due to its computational complexity.

In the present study, some modern technologies have been assessed, specifically examining the disparities between image captioning with AI language understanding and human perception. A few image captioning models (see Image Captioning Models Section) have been used to generate different types of captions that lately were processed by language models.

2.4 Large Language Models

Processing a sentence in a machine learning model (directly from a string, such as "Hello world") in one go is generally not feasible. The main reason it that standard neural network uses and works with matrices in real number space. There is no dictionary with pairs sentence-numerical representation of every possible sentence, so to obtain the numerical representation of the sentence, it is necessary to calculate it. First, to process the whole sentence, it is essential to divide it into smaller units for which some function that maps these segments into their numerical representation may be obtained. Thus, the initial step involves Tokenization, wherein the sentence is dissected into smaller semantic units such as words or subwords. Subsequently, these tokens undergo Word Embedding, a process whereby they are transformed into dense vectors, facilitating numerical processing.

Within the realm of neural network architectures, the transformer model is particularly distinguished for its capability to effectively manage long-range dependencies in

[1] Link to the project:
`https://developer.nvidia.com/embedded/community/jetson-projects/a-eye_for_the_blind`

textual data. This is primarily attributed to its utilization of the self-attention mechanism, which allows for a dynamic weighting of the influence of all other tokens in the sequence, regardless of their positional distance. This mechanism enables the model to establish connections between words regardless of their physical proximity. For example, in German sentences that always have the negation of the verb at the end of the sentence or more complex English sentences when we mention the same person multiple time using pronouns [62]. In this case, it is important that words do not depend on physical proximity.

The Transformer architecture comprises two primary components: the Encoder and the Decoder, as described in the fundamental paper "Attention is All You Need" [63]. The Encoder's role entails processing the input sentence to distill a comprehensive representation of its underlying concept. Subsequently, the Decoder utilizes this conceptual representation, in conjunction with access to the encoded input, to generate the desired output. In tasks like text generation, the Decoder employs a next-word-prediction approach to iteratively construct the output sequence.

The overall architecture does not always have to consist of both encoder and decoder parts. For tasks like classification, where generating human-readable output is unnecessary, an encoder that can summarize the context and idea of the input text is sufficient. On the other hand, if we imagine that the decoder forms the answer by composing the answer word by word based on the outputs generated so far, it can be adapted to generate output sequences from predefined input text, obviating the need for an Encoder. This Decoder-only architecture finds application in prominent models such as GPT (e.g., GPT-2, GPT-3, GPT-4) [64] and ChatGPT. A well-known representative of the Encoder-Only architecture is BERT [10]. BERT has spawned numerous derivative models like RoBERTa [65] and ALBERT [66].

Typically, to represent a sentence, the output of the Encoder's final layer is utilized, often by considering the classification token (e.g., *[CLS]* in BERT [10]) or by averaging the last hidden states across all tokens. Similarly, by employing the classification token in the beginning of the input we can perform classification tasks using decoder-only transformers.

2.4.1 Tokenization

Tokenization, within the domain of NLP and machine learning, involves transforming a sequence of text into smaller components called tokens. In the context of LLMs, tokenization is crucial for converting raw text data into a format that the model can process efficiently without losing text meaning. Typically, tokenization involves splitting the text into individual words or subwords, and sometimes even characters, syllables, depending on the specific tokenization scheme used. The main significance of this procedure lies in its ability to assist machines in comprehending, analyzing and interpreting human language [67].

While segmenting sentences by words appears logical [68], as each word encapsulates the entirety of its meaning, it presents a challenge in maintaining an exhaustive dictionary encompassing all encountered words. When we later come across a new, previously unseen word that is not in the dictionary, we cannot classify it. An illustrative instance occurs when we encounter a word with same or similar meanings, albeit in slightly var-

ied forms, such as ”offer”, ”offers”, ”offering”, and ”offered”, necessitating understanding their meanings repeatedly (during embedding learning). If we wanted to reduce the dictionary to the smallest possible size, we could split the sentence into individual letters. This poses comprehension challenges as the model must decipher word meanings incrementally letter by letter, leading to subpar results due to the aforementioned issues.

A harmonious compromise between these approaches involves segmenting sentences and words into subwords. Subwords are word fragments that offer the advantage of encapsulating broader semantic units akin to words, yet retain the ability to infer word meaning even with novel inputs, equivalent to character-level tokenization. Among tokenization methods, subword tokenization emerges as the most prevalent and widely adopted technique.

2.4.2 Word Embedding

Word embeddings are dense, multi-dimensional vector representations of words that facilitate the transformation of text data into numerical formats. Within Large Language Models (LLMs), word embeddings serve to represent each token as a fixed-size vector within a continuous vector space. In this vector space, words that are semantically similar used to have similar vector representation, so they are located close to each other (are embedded nearby each other - see Figure 2.3) [69, 70]. We distinguish between static and contextualized embeddings [71, 72].

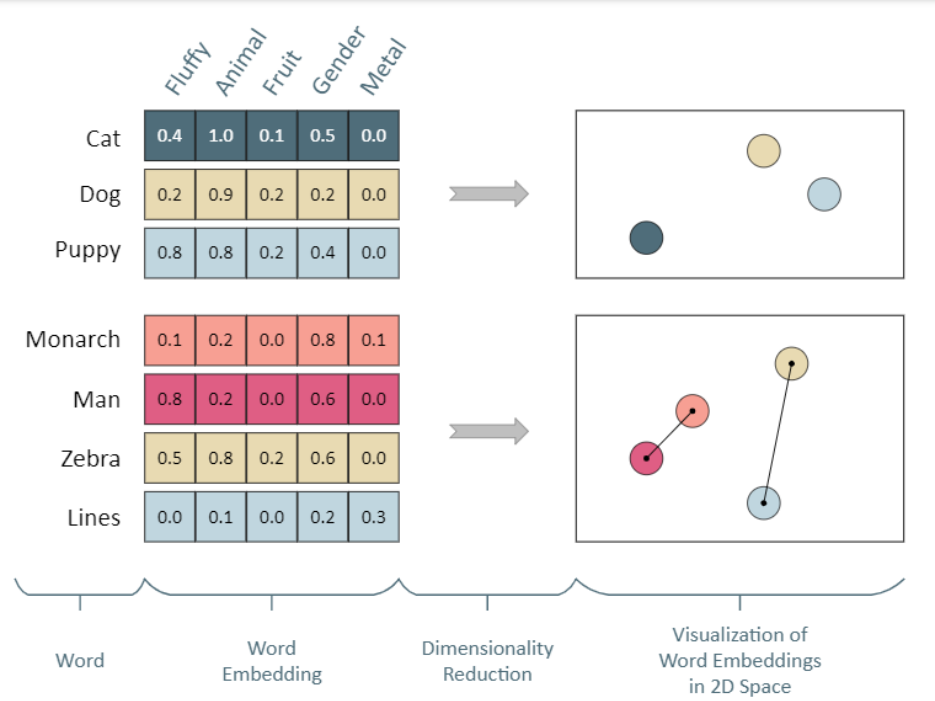


Figure 2.3 Word Embedding. Basic Pipeline of Word Transformation to a Vector. On the right side of the graph these vectors are represented as a dots in the 2D space.

Static embeddings (or content-independent embeddings), exemplified by techniques like Word2Vec [72], GloVe [73] and FastText [74], provide fixed representations for words irrespective of their context within a sentence or document. These embeddings encode semantic and syntactic information based on co-occurrence statistics in large text corpora. However, this type of embeddings lack information about the word's position in the sentence and its surrounding context. Consequently, any word with different meaning in distinct sentences acquire the same embedding, posing challenges for the model to achieve more deeper, nuanced understanding of the complete sentence. For example, consider the word "mouse", which can denote either a small mammal or a computer peripheral device.

In contrast to static embeddings, **contextualized embeddings** (content-dependent embeddings), exemplified by RNN based (CoVe, Flair or ELMo) and transformer based models, such as BERT [10] and GPT-2 [64], dynamically tailor word representations according to their contextual surroundings. By harnessing deep neural networks trained on extensive text corpora, contextualized embeddings capture real properties of each word in a sentence. Each word's embedding is not only influenced by its own identity but also by the neighboring words in the given sentence, enabling the model to grasp nuanced meanings and linguistic nuances more effectively.

2.4.3 Transformer Architecture

2.4.3.1 Encoder

Encoder is part of the transformer architecture that is responsible for processing inputs and their representation within the transformer, it does not create any new text in the output. It typically consists of several encoder blocks connected in series, where higher blocks serve to process more complex relationships and dependencies.

2.4.3.1.1 Input embedding

At the outset, when tokenizing the input text for model ingestion, tokens necessitate encoding into dense vector representations. The initial entry into the first block will be created by static embedding, which, however, does not contain any information about the position.

Since the transformer encoder processes the entire input at once, it is not possible to ensure that the position information is processed. Consequently, preceding the initial encoder block entry, positional embedding supplements the static token embedding, thereby integrating token position information into the model. In this way, the position of each token is introduced into the model. Given that the self-attention mechanism governs inter-token links, the encoder block's initial output yields contextualized token embeddings. While both input and output dimensions remain consistent across encoder blocks, positional information is solely appended to the initial entry, persisting through subsequent blocks via transmission.

2.4.3.1.2 Self-Attention Mechanism

At the input of token embedding into the encoder block, three vector representations are created, which are used for processing and the influence of other parts of the text on each token. The following description describes the processing of one token from input to output of the encoder block. This processing happens simultaneously for all inputs (tokens).

Using the weight matrices W_q, W_k, W_v, three vectors Q (query), K (key) and V (value) are created. Weight matrices are trainable parameters that acquire their value during network training. For any input:

$$Q = XW^Q$$

$$K = XW^K$$

$$V = XW^V$$

For trainable weight matrices $W^Q, W^K \in \mathbb{R}^{d \times d_k}$ and $W^V \in \mathbb{R}^{d \times d_v}$

The Self-Attention mechanism calculates the impact of each token from the input on the resulting output vector of our token. Pairwise computation across input tokens involves Q (query) and K (key) vector multiplication, culminating in weight determination signifying contextual relevance. The result is a list of weights, how important and how much influence each vector should have on our input. These values are reduced and normalized using the $softmax$ function and thus their sum in the result is 1. The resulting value is the weighted average V of the vectors of all elements of the input [75].

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

The output is a new vector that has also acquired properties from its neighbors based on common contextual binding.

2.4.3.1.3 Multi-Head Attention

While the self-attention mechanism singularly targets contextual features, by multiplying the number of attention heads, multi-head attention is created, which can encompass multiple contextual relations concurrently [76].

Each head, sharing identical embeddings or preceding encoder block outputs, generates unique vectors (Q, K, V) via distinct weight matrices $(W_q i, W_k i, W_v i)$. Following analogous self-attention procedures, resultant vectors from multiple heads conflate into a consolidated vector. The resulting vectors from several heads are concatenated together to form one large vector. The dimensionality of the vector is reduced using the weighting matrix W_o, typically to the original input size.

2.4.3.1.4 Feedforward Neural Network

The resulting vector from self-attention passes through the FFN layer, which introduces nonlinearity. An FFN layer can be, for example, a fully connected ReLU layer with four times as many hidden units as inputs, followed by another fully connected layer without activation with original size [77]. Initially, Layer Normalization was applied after the residual connection; however, in the newer improved configuration, layer normalization is exclusively applied in the FFN layer and precedes the Fully connected layers.

2.4.3.1.5 Normalization and residual connections

The mechanism of encoder block is a bit more complex as there are residual connections involved as well as normalizations of the output of the multi-head attention and FFN layer.

2.4.3.1.6 Output of the Encoder Block

Each encoder block processes its input and creates a representation with a deeper context at the output, similar to how CNN convolutional neural networks create more complex image features within each layer.

Since the block processed the vector representation of the token, its output was also a vector representation for each input separately.

2.4.3.2 Decoder

The decoder block within the Encoder-Decoder transformer architecture shares similarities with the Encoder in its structure and functionality. One of the main differences it, that decoder block creates an output in form of generated text. In a way, how the decoder works, the text is formed through word-by-word generation.

The embedding is received at the input of the decoder in a manner analogous to that of the encoder. Notably, multiheaded attention now incorporates masking, restricting the decoder's view to positions of tokens that have been generated in the sentence so far. A conventional method to mask other positions involves setting their "weight" before the softmax to $-\infty$.

Subsequently, the token's processing proceeds to the subsequent MultiHeaded attention, specifically the Encoder-Decoder cross attention. Here, the query vector Q in the decoder is generated from the token, while vectors K and V originate from the encoder. Following this stage, FFN and normalization are applied.

The principal distinctions between the encoder and the decoder lie in the decoder's limited scope during self-attention, focusing solely on the output generated thus far, and its incorporation of cross-attention in each decoder block, leveraging the processing of input text from the encoder.

2.4.3.2.1 Decoder-Only Architecture

Models like the ones belonging to the GPT family adhere to the Decoder-Only architecture paradigm.

Unlike the original architecture described in "Attention Is All You Need" [63], the encoder is omitted. Consequently, modifications are requisite in the structure of the Decoder block, necessitating the removal of the cross-attention head, thereby retaining only masked multihead attention capable of observing the inputs generated up to that point.

Within a decoder-only architecture, the input is inserted at the beginning of the decoder output, allowing the decoder to proceed with output generation. Notably, the input processing of decoder-only architectures closely resembles that of encoder-only architectures, particularly when comparing the output from (encoder/decoder) blocks, resulting in vectors representing the contextual representation of the input

2.4.3.3 Sentence Representation

2.4.3.3.1 Encoder

When obtaining a general representation of the entire input, one possibility is to use a special token, which is typically added to the beginning of the input, whose role is to represent the content of the entire sentence. However, for such a mechanism to be effective, the model must be trained to recognize and accurately interpret the designated classification token. For instance, BERT [11] directly incorporates such tokens and employs them for Next-Sentence-Prediction during training. Nonetheless, alternative models, such as RoBERTa [65], abstain from utilizing Next-Sentence-Prediction during training.

Alternatively, the second option involves deriving vectors from the entire output and passing them through a pooling layer—commonly through methods such as averaging. Average pooling emerges as the most prevalent form of pooling, although alternatives like max pooling are also feasible.

2.4.3.3.2 Decoder

A prevalent approach to obtain a contextual vector representation of the entire input entails averaging the outputs originating from the final decoder block. Alternatively, in certain scenarios, the utilization of classification tokens is feasible, particularly if the model has been trained for such a purpose.

Chapter 3

RELATED WORK

3.1 Image Captioning

Initial efforts to generate automated image captions were made before the advent of deep learning techniques. These methods primarily involved template-based approaches that relied on creating basic template sentences, which were then populated with specific results from an object detection model [78, 79, 80]. An alternative method is the retrieval-based approach, which utilizes a dataset of images with their related captions, rank them by measuring the similarities to find the set of images most similar to the one in question and then reuses their existing captions to create a new description for the queried image. This strategy is significantly limited when handling images that are not in the dataset and thus remain unclassified, i.e., unseen [81, 82].

Shortly after the advent of advanced computer vision techniques, many sophisticated and efficient image captioning methods were developed [83]. A lot of them use classic approach of processing visual objects, trying to understand the main concepts and then generate some phrases out of these concept descriptions with the use of linguistic models. These process is called **bottom-up approach**.

Another approach is the **top-down** method, which differs primarily by framing the image description task as a machine language translation problem. Instead of translating from one language to another, the model translates visual information into natural language.

This approach primarily leverages neural networks and convolutional neural networks to encode visual information and generate output text using recurrent neural networks (for more details, see Image Captioning Section) [84, 85]. Various implementations of image captioning differ mainly based on the type of RNN utilized, but many face the issue of uniformly distributing semantic concepts throughout the description. Therefore, a significant improvement was achieved by employing graph convolutional neural networks, which better understand the semantic and spatial relationships between objects [86, 87]. This enhancement allows additional information, such as location and velocity, to enrich the generated captions [88].

On the contrary, there is also ongoing research aimed at improving the quality of the caption text itself. For example, in [89], the authors proposed ATT-BM-SOM, a method tailored to enhance readability, sentence syntax, and overall caption structure. This framework leverages the attention balance mechanism and syntax optimization module to effectively integrate image information into the captions.

Current most popular state-of-the-art in image captioning is mixed approach that select and combine the results from both bottom-up and top-down approaches or transformer-based models [90].

Likewise, recent approaches often leverage transformer models, which offer the advantage of using attention mechanisms without relying on RNNs for image captioning [63], or create convolution-free networks. Many studies explore modifications to the self-attention operator [91, 92, 93]. Additionally, as previously mentioned, transformer architectures can be applied directly to image analysis, reducing the necessity for CNN usage [94, 95]. Specifically, in this context, a pre-trained Vision Transformer network (ViT) (that was also used in this experiment) serves as an encoder, followed by a standard transformer decoder for caption generation.

3.2 Language Processing

LLMs are sophisticated AI systems trained on extensive text datasets that may be used for text understanding, generating, translation and interacting with human language in a contextually aware manner [96, 9].

Before the era of transformers, LLMs used architectures like long short-term memory (LSTM) to create some connected word embeddings. For example, ELMo [97] represented a significant advancement in contextual understanding within language processing tasks, while ULMFiT [98] utilized LSTM to pre-train a language model on large datasets and fine-tune it for diverse tasks, pioneering effective transfer learning in natural language processing.

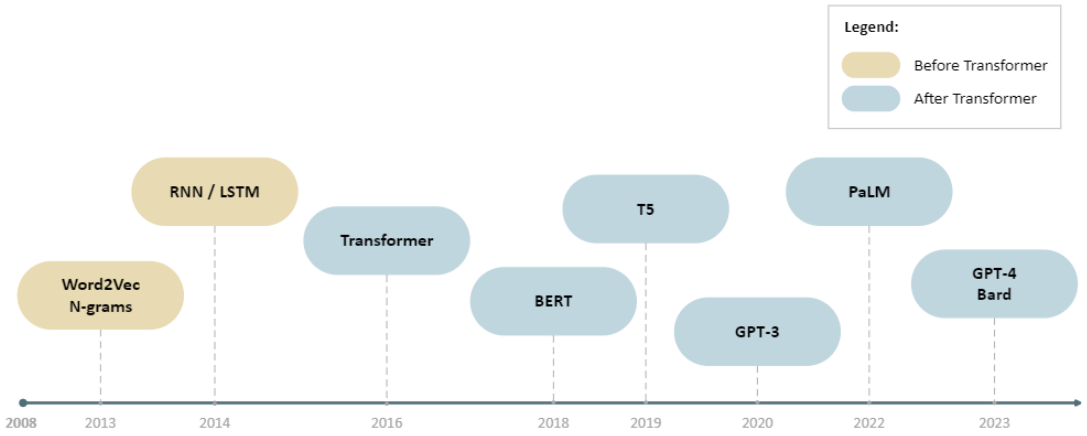


Figure 3.1 Overview of key milestones in the history of language models

With the advent of attention mechanisms [63], transformers have become the state of the art in language models, fundamentally altering the landscape of natural language processing (NLP) (see Figure 3.1). BERT [10] revolutionized the field by introducing deep bidirectional training, which allows for a more detailed and comprehensive understanding of context.

Building on BERT's success, several modifications have been developed to further enhance performance and address specific needs. RoBERTa (Robustly optimized BERT approach) [65] optimizes the pre-training process by training on a larger dataset and longer sequences, resulting in its improved accuracy and robustness. DistilBERT [99] was created to make the model architecture faster and lighter, reducing the model size by 40% while retaining 97% of BERT's performance. ALBERT (A Lite BERT) [66] introduces parameter-sharing strategies across layers to reduce model size and memory consumption significantly, increasing training speed and overall scalability. These modifications not only improve performance but also make BERT more adaptable to various applications, from large-scale industrial tasks to smaller, edge-based deployments.

Since then, the largest and most effective Large Language Models (LLMs) have been built using a decoder-only transformer-based architecture. This design enables efficient processing and generation of large-scale text data. Prominent examples include OpenAI's GPT series (e.g., GPT-3.5 and GPT-4, used in ChatGPT and Microsoft Copilot) [9], Google's Gemini [100], Meta's LLaMA family of models [101], Anthropic's Claude models [102] and Mistral AI's models ([1]).

The major advantage of these models is their ability to generate human-like text and solve a wide range of problems, from answering questions to generating creative content. These models excel in interacting through chatbots, providing insightful and context-aware responses to the primary queries. They can assist in various applications, such as drafting emails, creating content, providing customer support, translating languages and even engaging in complex problem-solving and decision-making tasks. This versatility and efficiency have made transformer-based LLMs the cornerstone of modern natural language processing and AI-driven communication tools.

To achieve multitasking capabilities and enhance the performance of LLMs, various alignment techniques and examinations have been conducted [103]. One notable method is on-policy reinforcement learning (RL), which trains and rewards the system based on selected data. This approach is exemplified by proximal policy optimization (PPO) [104], which fine-tunes the model using a reward system to optimize performance. A similar method, direct policy optimization (DPO), focuses on direct learning from human preferences without a reward system [105].

Another effective technique is iterative preference fine-tuning, where the model's performance is enhanced by repeatedly optimizing based on newly generated preference outcomes in each iteration. Modern approaches also achieve satisfactory results through prompt engineering and querying, which involves crafting specific input prompts to guide the model's responses. Additionally, a novel method called self-improving has been proposed, where the model iteratively plays against instances of itself, thus refining its capabilities without the need for additional human expert opponents [106]. These

[1]Link to the source: `https://docs.mistral.ai/getting-started/models/`

diverse strategies contribute to the robust and versatile performance of contemporary LLMs, enabling them to excel in various tasks and applications.

Nowadays, multimodal models (the one that take more than one type of input) become very popular not even for the researchers, but also for ordinary people in their everyday life. Vision language models, only one type of of multimodal models, are used on daily bases for performing image captioning (e.g. CLIP [107]), visual question answering (e.g. GPT-4-Vision [2]) and image or video from text generation (e.g. DALL-E [108], SORA [109]).

Another advantage of current models is their ability to learn not only from a vast amount of training data but also from ongoing interactions. This means that the more they are used, the better they become at generating relevant and accurate responses. This continuous learning process allows the models to adapt and improve over time, enhancing their performance and making them more effective in various applications.

3.3 LLMs Performance Evaluation

Despite their numerous advantages, LLMs raise significant ethical concerns, particularly in the areas of data privacy, accuracy and correctness. Furthermore, they can inadvertently perpetuate biases present in their training data, resulting in skewed or unfair outcomes. That is why ensuring the correctness of LLM-generated content is critical, as inaccuracies can have profound consequences due to the widespread use and trust in these technologies by the general public.

Addressing these challenges is essential for the responsible deployment of LLM technology. There are numerous frameworks and platforms designed to evaluate LLMs. These evaluation systems utilize various metrics to assess performance, such as accuracy, fairness and responsiveness. Some platforms offer even real-time evaluations of AI outputs to check for retrieval performance and responsibility.

It is crucial to clearly define evaluation criteria at the first place to ensure comprehensive and meaningful assessments.

Evaluating LLMs in real-world scenarios is a complex and time-consuming task. Unlike traditional models that aim to produce outputs strictly matching a predefined ground truth, LLMs are often expected to generate human-like responses. This expectation adds a layer of complexity to the evaluation process. It is not enough for an LLM to simply perform as designed; it must also produce outputs that meet the nuanced and subjective criteria of human communication. This dual requirement necessitates sophisticated evaluation methods that often go beyond standard metrics, taking into account factors such as coherence, relevance and the naturalness of the generated responses.

As an example, see Table 3.1 [3]. This table outlines five different techniques used to measure LLM performance. Some methods are designed to assess whether the model can perform a basic set of tasks, while others evaluate sentence completion or multitasking ability. Additionally, some techniques focus on the quality of the response, considering factors such as perplexity, relevance and truthfulness. In reality, even more advanced

[2] Link to the source: `https://cdn.openai.com/papers/GPTV_System_Card.pdf`

[3] Link to the source page: `https://medium.com/data-science-at-microsoft/evaluating-llm-systems-metrics-challenges-and-best-practices-664ac25be7e5`

Table 3.1 An example of different LLM evaluation benchmarks

Benchmarks	Description	Reference URL
GLUE (General Language Understanding Evaluation)	A benchmark offering a standardized collection of diverse NLP tasks to assess the effectiveness of various language models	`https://gluebenchmark.com/`
SuperGLUE	A benchmark comparing more challenging and diverse tasks to GLUE, incorporating comprehensive human baselines	`https://super.gluebenchmark.com/`
HellaSwag	A benchmark assessing the proficiency of an LLM in completing sentences	`https://rowanzellers.com/hellaswag/`
TruthfulQA	A benchmark evaluating the accuracy and truthfulness of the model's responses	`https://github.com/sylinrl/TruthfulQA`
MMLU (Massive Multitask Language Understanding)	A benchmark assessing the multitasking capability of the LLM	`https://github.com/hendrycks/test`

methods exist to assess models' regulatory compliance, hallucination levels and ability to handle harmful content [96]. All of these varied approaches ensure a comprehensive evaluation of LLMs, addressing both their technical capabilities and correct human-like quality of their outputs.

With all of these needs, a new position has emerged in the current era: LLMOps (a specialization of MLOps designed specifically for Large Language Models). The primary task of LLMOps is to integrate a CI/CE/CD (Continuous Integration/Continuous Evaluation/Continuous Deployment) approach to effectively oversee the lifecycle of applications powered by LLMs. This role ensures that LLMs are consistently updated, rigorously evaluated and reliably deployed, maintaining high standards of performance, safety and ethical compliance throughout their use.

The challenge of evaluating LLMs arises from the repetitive cycle of running LLM applications on a set of prompts, manually inspecting outputs, while attempting to check the input quality as well. This process often requires human interaction and evaluation, making it time-consuming, subjective and costly, particularly for large-scale assessments.

Such dependence on human input for evaluating LLM outputs highlights the inefficiency and potential bias in current evaluation methods. For this reason some modern investigations tries to perform more automatic evaluations and use some advantages of other AI technologies. For example in article [110] authors explore the effectiveness of Large Language Models (LLMs) as automatic evaluators using simple prompting and in-context learning. By assembling 15 LLMs of varying sizes, the study evaluates their output responses through preference rankings conducted by other LLMs. The Cognitive Bias Benchmark for LLMs as Evaluators (CoBBLEr) is introduced to measure six cognitive biases. Findings reveal approximately 40% of significant biases among differ-

ent models, questioning LLMs' reliability as evaluators. Additionally, the study finds a 49.6% Rank-Biased Overlap (RBO) score between human and machine preferences, indicating its misalignment. Consequently, the research suggests that LLMs may not yet be suitable for automatic annotation that aligns with human preferences.

This study compares the LLMs' understanding of basic concepts from our everyday life to human perception. The following chapters detail the entire process and present key findings.

Chapter 4
METHODOLOGY

4.1 Dataset

The *THINGS* dataset [31] comprises a collection of naturalistic images depicting basic real-life concepts along with their corresponding names and fuzzy categories. Fuzzy categories here mean that one concept can belong to more than one category. To build this database, the authors engaged with 1,395 workers from the online crowdsourcing platform Amazon Mechanical Turk. These workers were involved in various tasks related to dataset creation, including object image collection, preprocessing, naming and categorization. Dataset primary purpose is to facilitate cognitive science research focusing on overall object concepts across various domains such as visual perception and semantic awareness. The dataset holds relevance across a diverse array of disciplines, including psychology, neuroscience and computer science.

The dataset consist of 1,854 meticulously chosen object concepts from American English, accompanied by corresponding images and labels. These selections underwent a harsh process that combined word-sense disambiguation and crowdsourcing, ensuring precise representation of common concepts. Subsequently, all concepts were categorized into 27 overarching categories using both human judgment and WordNet's taxonomic classification, resulting in an extensive collection of over 26,000 high-quality images. The final dataset was reduced to include one image per concept and a public version of all images was released in the *THINGS+* dataset [32]. Furthermore, this dataset includes an additional 53 higher-level fuzzy categories, allowing objects to be assigned to multiple categories simultaneously. By exploring object categorization and semantic relationships, this dataset serves as a valuable resource for understanding how ordinary people perceive, recognize and interact with objects in their everyday environment. An examples of images from *THINGS+* dataset [32] along with their respective categories may be observed in Figure 4.1.

Employing the *THINGS* dataset [31], the researchers conducted a triplet odd-one-out experiment (see Section 2.1) to evaluate perceived image similarity within the context of a third (most dissimilar) image. Human participants were presented with sets of three object figures and tasked with identifying the most distinct one. Through this process, the study aimed to check both context-independent and context-dependent perceived similarities. This methodology yielded a total of 4.70M human similarity judgments.

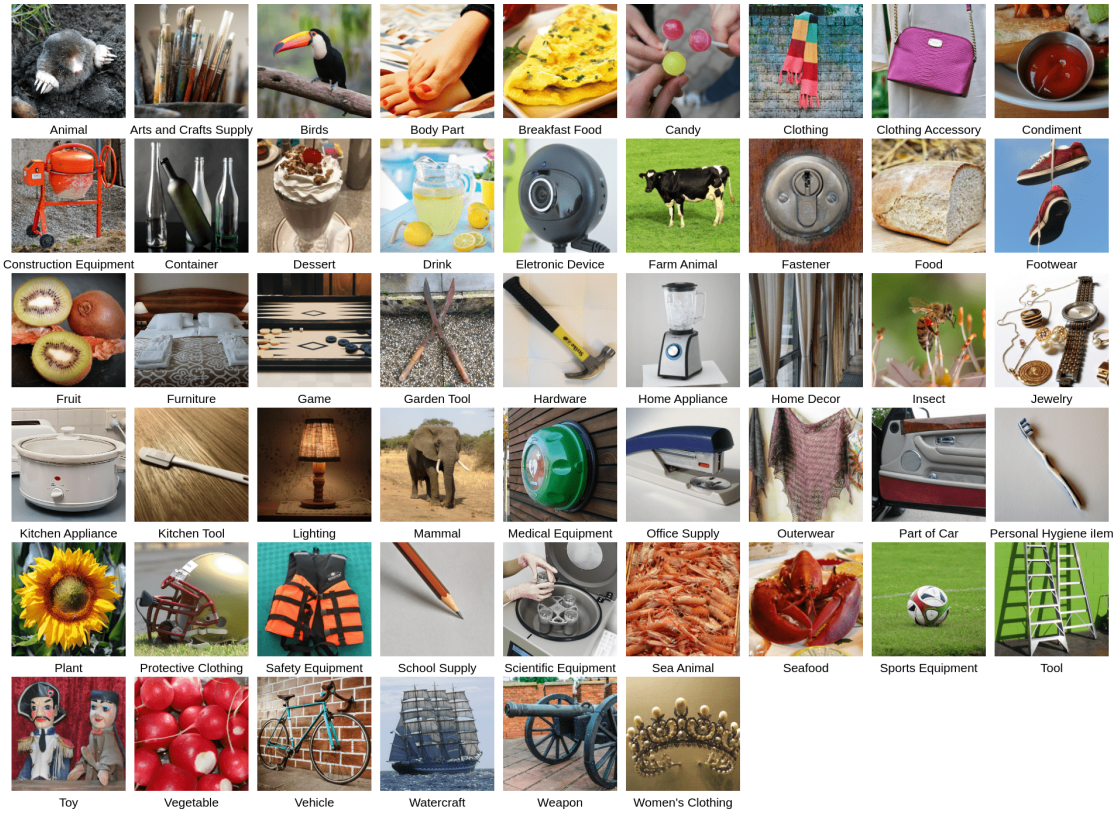


Figure 4.1 Fifty-three broader image categories that have been examined in the study. All of the images are copyright-free images from *THINGS+* dataset [32].

For the purpose of this study, the initial focus was on verifying the quality of the image dataset and the collected triplets. Efforts were made to ensure uniformity in image format, size, and correspondence between image names and content, facilitating their usability in subsequent processing stages. Additionally, duplicates were removed from the original triplet dataset to enhance the credibility of the final evaluation. Consequently, the input triplet dataset with 4.7M records was divided into two subsets: one containing duplicates and the other devoid of them. Throughout this text, the subset consisting solely of repeated images within the triplets we denoted as the **"Repeated Triplet Dataset"**.

It's crucial to understand that in this case, identifying the odd-one-out lacks a definitive ground truth. For instance, presenting a triplet like $\{peacock, toucan, owl\}$ to various individuals would likely result in a wide range of answers. However, by gathering responses from a significant number of participants, it becomes possible to calculate the collective probability for each object concept that humans, in general, would perceive as more dissimilar within the triplet. To explore this, the researchers intentionally gathered batches of trials (from different participants) for a randomly selected set of triplets. Therefore, the inclusion of duplicate triplets aimed to assess within-subject variability and gain insights into individual differences in perceived similarity judgments.

To illustrate further, once again consider a triplet consisting of $\{peacock, toucan, owl\}$.

After conducting 100 trials, let's say we obtained the following proportions of answers: $\{0.5, 0.2, 0.3\}$. In this scenario, the ideal classifier would predict *peacock*, achieving a maximum accuracy of 0.5. Conversely, the worst classifier would select *toucan*, resulting in an accuracy of 0.2. Comparatively, a random classifier would have an expected accuracy of 1/3, as it would randomly select one of the three options, aligning with the chance of selecting the correct answer out of the available choices.

4.2 Experiment Pipeline

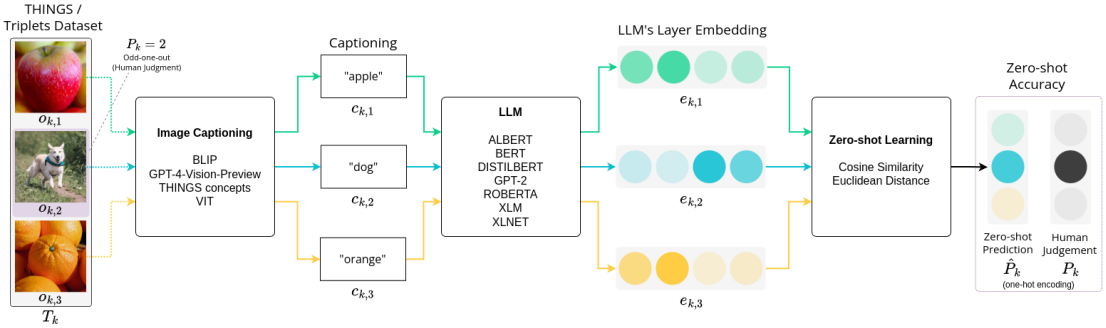


Figure 4.2 Illustration of overall methodology utilized in this study. From left to right: input THINGS and triplets dataset preprocessing, image captioning, description processing by LLM, extraction of hidden states and their distance analysis, selection of odd-one-out item, final results analysis. All of the input images are copyright-free images from *THINGS+* dataset [32].

To verify the alignment of LLM representations with human concept understanding, we used object triplets extracted from the THINGS dataset. Our full pipeline is illustrated in Figure 4.2. It consisted of the following stages:

1. A triplet (collection of three object concepts) is **the input** of our method. In the example (in Figure 4.2), a triplet k is composed of the objects $o_{k,1}, o_{k,2}, o_{k,3}$ that respectively represent an *apple*, *a dog*, and *an orange*.

2. Each object is then **captioned** using the same image captioning method (e.g., BLIP). For instance, the first picture could receive the caption "apple". Captioning methods are discussed in Section 4.3.

3. The generated captions are **tokenized and processed by an LLM** (e.g., BERT) to extract feature vectors (hidden states). See Section 4.4.

4. The next stage includes **zero-shot learning methodology** with distance metrics and the most distinct object identification (see Section 4.4.2). As label, we use the odd-one-out object that was labeled by the human subject $P_k()$. In our toy example, $P_k = 2$ represents that dog was the selected one.

In conclusion, the general LLM performance was analyzed (captioning×language models×layers×distances $= 4 \times 21 \times 5 \times 2 = 840$ different results have been obtained) and discussed in the following Section 5. To avoid the burden of computation, we first preprocess the embeddings for each image and then use the preprocessed version in our analysis.

4.3 Image Captioning Models

In the initial phase of this experiment, captions were generated for all images within the THINGS dataset [31]. To accomplish this, several deep learning models for vision interpreting were employed, as aggregating multiple responses often leads to superior outcomes. Additionally, the original image labels were slightly adjusted and incorporated into subsequent processing stages. Below is a list of all the methods utilized for the image captioning task.

(a) Image Label: mouse1
Basic Caption: mouse animal
ViT Caption: a small brown and white bird sitting on a piece of paper
BLIP Caption: a mouse on a mouse board
GPT Caption: The image shows a close-up view of a rodent, likely a mouse or rat, standing on a surface that has printed text. The rodent has a grayish-brown coat of fur, large round black eyes, prominent rounded ears, and a pinkish snout. Its forelimbs are visible with pinkish skin, and it appears to be looking off to the side. The background contains a blurred section of text, which is not readable

(b) Image Label: mouse2
Basic Caption: computer mouse
ViT Caption: a black mouse sitting on top of a white surface
BLIP Caption: a computer mouse on a white surface
GPT Caption: The image shows a wired computer mouse on a white background. The mouse is predominantly black with a gray trim around the base. It features a left and a right click button, a scroll wheel, which appears to be orange, and possibly a third button on the left side. The cord extends from the front of the mouse and appears to be black as well. There is a textured pattern on the mouse pad or surface on which the mouse is resting.

Figure 4.3 Example of captioning results for 2 input images. Both images are copyright-free images from *THINGS+* dataset [32]. For image captioning ViT [111], BLIP [112] and GPT-4V [90] models have been used.

4.3.1 Vision Transformer (ViT) with GPT-2

The first model employed for captioning input images combines the Vision Transformer (ViT) [111] with GPT-2 [64].

The encoding component utilizes the Vision Transformer [111], specifically trained on the ImageNet dataset. It operates by dividing image inputs into patches and incorporates an additional token to encapsulate global image features such as size and color palette. Subsequently, this sequence of token embeddings is fed into the standard Transformer, which employs attention mechanisms to discern relationships within the input and process it in a manner akin to conventional NLP tasks. This streamlined approach has demonstrated impressive results while minimizing computational costs, rendering it a popular choice among contemporary methodologies.

The decoding stage, on the other hand, involves a language model, GPT-2 [64], which is tasked with generating textual descriptions based on the encoded image features. One of the disadvantages is that this model may struggle with generating captions for images that contain multiple objects or conveying certain abstract concepts that are difficult to represent visually (for example emotions, metaphors, etc.).

4.3.2 Bootstrapping Language-Image Pre-training (BLIP)

Introducing BLIP [112], a pretrained model that may be used for image captioning tasks, designed to facilitate great portability across a spectrum of vision-language tasks through its multimodal combination of encoder and decoder parts. This advanced model not only comprehends the intricate relationships between objects within images but also leverages spatial arrangements to craft descriptive captions. Its' architecture enables flexible transferability between vision-language understanding and generation tasks.

BLIP offers three key advantages: Firstly, it supports multiple languages. Secondly, its versatility extends beyond image captioning, encompassing tasks such as visual question answering, visual dialog, zero-shot text-video retrieval, and more. Lastly, in the realm of captioning, BLIP distinguishes itself by prioritizing the creation of human-like captions, eschewing generic descriptions for more nuanced and contextually relevant interpretations.

The BLIP model, pioneered by Salesforce Research, underwent training on the COCO dataset, a comprehensive collection comprising more than 120,000 image-caption pairs. At its core, BLIP integrates the versatile Vision Transformer (ViT) as its encoder mechanism. Subsequently, the decoding process focuses on crafting textual descriptions based on these extracted features. In this phase, BLIP employs a Language Modeling Loss, that motivates this decoder to generate precise and contextually relevant descriptions.

4.3.3 Generative Pre-trained Transformer 4 (GPT-4V)

In the subsequent phase, captions were generated using the GPT-4V model ('V' for 'Vision') [90] - current state-of-the-art. Similar to its predecessors, this model utilizes the transformer architecture, renowned for its adeptness with sequential data processing. The versatility of GPT models extends to various applications, with the option to fine-tune them on specific datasets for even better outcomes [113]. Nonetheless, GPT-4 is

not without its known constraints, including social biases, potential hallucinations and susceptibility to adversarial prompts.

In comparison to the previous methods, these descriptions are much more realistic, longer, detailed and more human-like. Employing the GPT-4-Vision API [1] with precise prompts to describe images facilitated this advancement. In this study, generated answer is limited to a maximum of 300 tokens, which is approximately 200 words. However, there were limitations encountered during this phase, notably certain image types prohibited from processing due to API policies. For instance, images featuring lingerie remained uncaptioned as they were ineligible for processing (in the result it was necessary to remove at about 7.5K triplets with this object).

4.3.4 Original Naming

Lastly, a captioning relying on the original image names is employed. Certain input image labels undergo minor preprocessing, entailing the addition of essential information to enhance clarity and distinguish the content depicted in the images. For instance, the term *pipe* may be associated with both *plumbing pipe* and *smoking pipe*. - for this reason we added an adjective for disambiguation.

Figure 4.3 displays selected outcomes of image captioning. It also shows the difficulties in the correct concept understanding (see some issues in ViT descriptions concerning a bird, mouse as an animal and mouse as a computer device misinterpretations).

It is important to emphasize that 3 captioning methods (Original, ViT and BLIP) produces deterministic captions, while GPT-4V model - stochastic ones.

It is crucial to note that the caption generation and evaluation were conducted using the *THINGS* dataset [31]. However, for the purpose of this work, publicly available images from the *THINGS+* dataset [32] were shown, potentially resulting in slight variations in color palette.

Neither of the mentioned captioning techniques covers the object's identity or its context within the image. Instead, they solely focus on conveying the visual content inherent in the image, discernible to the observer.

Assessing the quality of captions in this study without direct human involvement proves exceedingly challenging. The dataset comprises solely brief image names, typically one or two words, rather than comprehensive descriptions, making the establishment of expected ground truth captions unfeasible. Alternatively, evaluating the correlation between generated captions is plausible; however, the inherent variability in length and linguistic style across the captions also undermines the precision and reliability of such an assessment.

To evaluate the captions, the percentage of captions containing the precise input image concept derived from the *THINGS* dataset was considered. The output result was 13% for ViT model, 40% for BLIP and 80% for GPT-4-Vision-Preview.

[1] Link to the official page: `https://openai.com/index/gpt-4`

4.4 Language Processing

Following the completion of captioning for all images in the input dataset, the text processing phase takes place. This phase involves analyzing various potential NLP models, putting necessary captions and then extracting the final hidden state from each model (see Utilized Language Models). Subsequently, these hidden vectors are combined into triplets and passed into subsequent scoring stages to assess distances and extract the farthest image vector representation (see Scoring Zero-shot Learning).

4.4.1 Utilized Language Models

Down below there is a list of 8 open source machine learning framework types for NLP (with the exact models used in this study).

1. **BERT**

   The BERT (Bidirectional Encoder Representations from Transformers) architecture [10] consists of bidirectional transformers, allowing the model to capture the information from left and right contexts in a given input text sequence. During the training phase, BERT learns how to generate contextualized embeddings for each token in the input by considering surrounding words. The model was trained with the use of unsupervised learning on a wide scope of input text data. In addition, it uses masked language modeling and next sentence prediction to provide a better final result. Combination of this approaches makes it easier for the model to understand the relationships between words in the entire sentence.

   BERT models come in several variations distinct mainly in different language contexts and computational requirements. The ***bert-base-cased*** variant is a base model trained on cased text data. The ***bert-base-multilingual-cased*** model extends base model capabilities to multiple languages, making it suitable for multilingual applications. For more computationally complex tasks, there are larger variants such as ***bert-large-cased***, which offers increased capacity and performance. Additionally, corresponding to its name, variants like ***bert-large-cased-whole-word-masking*** incorporate techniques like whole-word masking during pre-training to further improve model performance and robustness.

2. **ALBERT**

   ALBERT (A Lite BERT) [66] (with its relevant variations: ***albert-large-v1***, ***albert-large-v2***, ***albert-xxlarge-v1*** and ***albert-xxlarge-v2***) is a modification of the BERT architecture that aims to improve its efficiency and suitability while maintaining or even surpassing the performance of traditional BERT models. ALBERT achieves this by significantly reducing the number of model parameters (with the use of matrix factorization and cross-layer parameter sharing) without sacrificing overall effectiveness. Additionally, model utilizes self-supervised learning during the training, including masked language modeling and sentence order prediction (checking whether both sentences are in the correct or inverse order), to learn rich representations of text data.

3. **DistilBERT**

   DistilBERT is a smaller and faster variant of traditional BERT, designed to be more efficient with fewer amount of computational resources (at about 60% of BERT original size) [99]. This reduction in size is achieved through different techniques, mainly knowledge distillation.

   The ***distilbert-base-multilingual-cased*** and ***distilbert-base-uncased*** variants are modification specifically used for multilingual applications. Consequently, The "cased" variant preserves the case information in the input text, while the "uncased" variant treats all text as lowercase.

4. **RoBERTa**

   Another variant of BERT model is RoBERTa [65], which does not use next sentence prediction, has modified key hyperparameters and distinct training process in comparison to the traditional method. Unlike the other models, which are trained on diverse datasets with various pre-training objectives, RoBERTa is using a single large corpus of text data with a consistent objective and employs dynamic masking. This approach enables to learn more robust and generalizable representations of language.

   RoBERTa is available in several variants, including ***roberta-base*** and ***roberta-large***, which differ in terms of model size and capacity. Additionally, RoBERTa includes variants like ***roberta-large-openai-detector***, which are specifically trained for specific use cases (for example it may predict if text was generated by a GPT model). It makes this variant powerful tools for a wide range of different NLP applications.

5. **GPT2**

   GPT-2 (Generative Pre-trained Transformer 2) is a state-of-the-art language generation model developed by OpenAI [9]. It is based on the transformer architecture and trained on a dataset of 8M web pages using unsupervised learning. The main idea is to predict the next word in a sequence based on the previous ones. Because of attention mechanism GPT-2 model may focus on some exact part of input text to generate to be the most relevant outcome. Still, this approach has some disadvantages like generating repetitive or nonsensical sentences while producing long text.

   GPT-2 model is also available in several sizes, each with varying capacities and capabilities (for example, ***gpt2***, ***gpt2-medium***, ***gpt2-large*** and ***gpt2-xl***).

6. **GPT**

   Another separate model family is OpenAI's text embeddings represented on their */embeddings* endpoint [2] in the OpenAI API [3]. Several models of different size and architecture have been used: ***text-embedding-ada-002***, ***text-embedding-3-small*** and ***text-embedding-3-large***. Their precise architecture and training process remains unknown.

[2]Link to the endpoint: `https://platform.openai.com/docs/api-reference/embeddings`
[3]Link to the OpenAI API: `https://platform.openai.com/docs/overview`

7. **XLM**

   XLM (Cross-lingual Language Model) is another family of pre-trained multilingual language models developed by Facebook AI Research [114]. One of their key features is their ability to encode and process text in different languages using a shared vocabulary and model parameters, helping cross-lingual transfer learning.

   The XLM models are available in various configurations. For example, ***xlm-mlm-xnli15-1024*** variant is fine-tuned on the XNLI dataset and uses a multilingual masked language modeling. On the other hand, the ***xlm-mlm-en-2048*** variant has larger model size and is optimized for English-language related tasks only.

8. **XLNET**

   The last model is XLNET [115], which is another bidirectional autoregressive model that use permutation language modeling, which allowing it to capture dependencies among word positions in a sentence. For achieving this, the probability of a word token among all permutations of word tokens in a sentence is considered during the training phase.

   There are a few different modifications of XLNET, for example ***xlnet-base-cased*** and ***xlnet-large-cased***.

In the context of language models and transformers, **"hidden state"** term takes place. Basically, these models consist of several similar layers that are stacked one by one. Each layer has its certain output that is passed to the next layer as an input. This layer output represent the whole important processed information from the input as a vector and is called "hidden state" (on Figure 2.2 states are represented as horizontal lines between the model layers).

For the purposes of this study, the hidden states from the last 5 model layers have been extracted and used for the future evaluation. Each model layer output consisted of a few token vector representation. These vectors have been averaged to the one final vector that proceeded to the next evaluation steps. This process was applied to each LLM model.

4.4.2 Scoring Zero-shot Learning

Let a triplet T_k be sampled from dataset $\mathcal{T}$ and composed of the illustration objects $\{o_{k,1}, o_{k,2}, o_{k,3}\}$, where $o_{k,1} \in \{1, 2, \cdots, 1854\}$. These objects may be a triplet of {peacock, toucan, giraffe} images. After a brief triplet analysis most of human beings will probably choose image of *giraffe* as the most distinct one. On the other hand, for artificial intelligence it is much harder task.

To tackle this task, the input image undergoes processing to be represented as a single vector. As previously discussed, this vector representation, referred to as the hidden state obtained by image description processing, plays a pivotal role in the analysis. Consequently, the triplet of objects $\{o_{k,1}, o_{k,2}, o_{k,3}\}$ is transformed into a triplet of embeddings $\{h_{k,1}, h_{k,2}, h_{k,3}\}$, which can also be explored within a vector space. The most distinct object (the odd-one-out) is represented by the most distinct vector. This vector has to be located farthest apart from the others in this space.

To identify the most dissimilar vector Euclidean or Cosine distance metrics may be utilized.

The **Euclidean distance** is a simple measure of the shortest straight-line distance between two vectors in Euclidean space. The smaller this distance is, the more similar input vectors are. Mathematically, the Euclidean distance between two object vector representations $h_{k,1}$ and $h_{k,2}$ with corresponding quasi-coordinates $(h_{1,n}, h_{2,n}, \cdots, h_{k,n})$ in k-dimensional space for each $n \in \{1, 2, \cdots, 1854\}$ is given by the formula:

$$euclidean\_distance(h_{k,1}, h_{k,2}) = \sqrt{(h_{1,1} - h_{1,2})^2 + (h_{2,1} - h_{2,2})^2 + \cdots + (h_{k,1} - h_{k,2})^2}$$

where:

$h_{k,1}$ and $h_{k,2}$ are vector representations (hidden states) for $\mathbf{i} = 1, 2, \cdots, \mathbf{k}$ of $o_{k,1}$ and $o_{k,2}$ respectively

$euclidean\_distance(h_{k,1}, h_{k,2})$ marks for a Euclidean distance between input vectors ($h_{k,1}$ and $h_{k,2}$)

Another scoring metric is **Cosine distance** that measures how different two vectors in multi-dimensional space are. For doing this cosine value of the angle between the these vectors is considered, as it represents the relationship among vectors regardless of their magnitudes. In general, the cosine distance between $h_{k,1}$ and $h_{k,2}$ is given by:

$$cosine\_distance(h_{k,1}, h_{k,2}) = 1 - \frac{h_{k,1} \cdot h_{k,2}}{\|h_{k,1}\| \cdot \|h_{k,2}\|} = \frac{\sum_{i=1}^{k} h_{k,1} h_{k,2}}{\sqrt{\sum_{i=1}^{k} {h_{k,1}}^2} \sqrt{\sum_{i=1}^{k} {h_{k,2}}^2}}$$

where:

$h_{k,1} \cdot h_{k,2}$ is scalar product of hidden states $h_{k,1}$ and $h_{k,2}$

$\|h_{k,1}\| \cdot \|h_{k,2}\|$ is the product of their lengths

$cosine\_distance(h_{k,1}, h_{k,2})$ marks for a Cosine distance between input vectors ($h_{k,1}$ and $h_{k,2}$)

Cosine distance ranges from 0 to 2, with 0 indicating that the vectors are identical, 1 indicating they is no correlation among them and values close to 2 indicating they are orthogonal or unrelated.

To sum up, the **similarity-based** zero-shot procedure is as follows. For every ordered objects triplet $T_k = \{o_{k,1}, o_{k,2}, o_{k,3}\}$ the corresponding vector $\{h_{k,1}, h_{k,2}, h_{k,3}\}$ is created and then passed to any of mentioned scoring methodology (Euclidean or Cosine). The distance between each pair of vectors is calculated. It is represented by $D_k = \{d_{k,1}, d_{k,2}, d_{k,3}\}$, where $d_{k,1}$ value represents distance between the remaining other pair of vectors $h_{k,2}$ and $h_{k,3}$. The estimated position $\hat{P}_k$ of the odd-one-out concept in triplet T_k is defined as

$$\hat{P}_k = \text{argmin} D_k$$

as the furthest vector (the most distinct one) in a triplet is not the part of the shortest distance

Chapter 5

EXPERIMENTS

5.1 Zero-shot Prediction

In this section some results obtained from the whole experiment (see Section 4) will be discussed. As it was mentioned before, in the final evaluation was considered mainly Accuracy metric (see its formula down below) for each combination of captioned image from *THINGS* dataset [31], NLP model processing output, models' layers and distance calculation procedure.

$$Method\ Accuracy = \frac{number\ of\ correctly\ separated\ triplet\ objects}{number\ of\ all\ triplets} \times 100\%$$

Note that, three captioning models (ViT [111], BLIP [112] and GPT-4V [90]) have been used along with processed image name. All of them are described in Section 4.3. Moreover, 24 different models from multiple LLM families have been considered: ALBERT [66], BERT [10], DistilBERT [99], GPT and GPT-2 [9], RoBERTa [65], XLM [114] and XLNet [115]. In Section 4.4 their main properties have been mentioned. In addition, more exact details about their architecture and training data are described in Table 5.1. Also, the last 5 model layers' output have been extracted an compared with the use of 2 distance evaluation metrics (see Section 4.4.2).

It is important to note that most of the analysis in this section are separate for original *Triplet Dataset* (without any repeated object triplet combination) and *Repeated Triplet Dataset* (the one that contains multiple human judgements about odd-one-out object within the same triplet. For more details see Section 4.1.

Table 5.1 shows some insights about maximum experiment performance within its group (model type, pre-trained model, its architecture details, dataset and distance evaluation metric).Values of the best performers are highlighted.

It should be highlighted the correlation between *Triplet* and *Repeated Triplet Datasets*. As expected, most of the time better model performance on the first mentioned dataset is also notable on the second one. Still, the maximum accuracy for *Repeated Triplet Datasets* is much lower - it is just slightly higher from Random Classifier, so it will be discussed in the separate Section 5.2.

Table 5.1 Zero-shot Learning percentual accuracy and model information. The metric represents the best accuracy (in %) within model layers and all captioning methods. **Legend:** L is total number of layers, H is the size of model last hidden state and P is the number of parameters (weights) in the model.

Model Type (Dataset)	Model	L	H	P	Time (min)	Triplet Dataset		Repeated Triplet Dataset	
						Cosine	Euclidean	Cosine	Euclidean
BERT (Wikipedia, BookCorpus)	bert-base-cased	12	768	110M	1.54	47.51	47.55	34.31	34.35
	bert-base-multilingual-cased	12	768	110M	1.62	44.3	44.79	34.06	34.19
	bert-large-cased	24	1024	340M	3.02	47.03	46.84	34.52	34.48
	bert-large-cased-whole-word-masking	24	1024	340M	3.06	48.4	48.36	34.39	34.42
ALBERT (Wikipedia, BookCorpus)	albert-large-v1	24	1024	17M	3.02	43.99	43.93	34.26	34.27
	albert-large-v2	24	1024	17M	3.42	46.49	46.4	34.47	34.47
	albert-xxlarge-v1	12	4096	223M	7.53	44.56	44.64	34.34	34.37
	albert-xxlarge-v2	12	4096	223M	7.67	43.6	44.07	34.31	34.34
DISTILBERT (Wikipedia, BookCorpus)	distilbert-base-multilingual-cased	6	768	134M	0.98	42.28	42.22	34.12	34.09
	distilbert-base-uncased	6	768	66M	0.97	49.09	48.99	34.65	34.51
ROBERTA (BookCorpus, English Wikipedia, CC-News, OpenWebText, Stories)	roberta-base	12	768	125M	1.65	43.5	43.46	34.0	33.98
	roberta-large	24	1024	355M	3.18	49.0	49.03	34.57	34.55
	roberta-large-openai-detector	24	1024	355M	3.18	38.41	38.46	33.68	33.65
GPT2 (WebText)	gpt2	12	768	117M	1.81	48.34	47.72	34.3	34.38
	gpt2-large	36	1280	774M	5.57	49.97	50.01	34.54	34.55
	gpt2-medium	24	1024	345M	3.52	49.74	49.44	34.55	34.6
	gpt2-xl	48	1600	1558M	8.52	49.53	49.15	34.6	34.55
GPT	text-embedding-ada-002		1536		50.0	33.36	33.36	33.43	33.43
	text-embedding-3-small		1536	125M	50.0	33.33	33.33	33.51	33.51
	text-embedding-3-large		3072	760M	50.0	33.34	33.34	33.52	33.52
XLM (Wikipedia, Toronto Book Corpus)	xlm-mlm-en-2048	12	2048	550M	2.87	36.89	36.89	33.69	33.72
	xlm-mlm-xnli15-1024	12	1024	550M	2.41	34.36	34.35	33.59	33.61
XLNET (Wikipedia, BookCorpus, Giga5, etc.)	xlnet-base-cased	12	768	110M	2.75	48.85	48.34	34.54	34.56
	xlnet-large-cased	24	1024	340	5.9	49.96	49.93	34.49	34.52

Another table parameter is *Time* column, which indicated how long it takes (in minutes) to process whole input dataset of image descriptions and extract the last hidden state out of the model [1]. It is important to note that for greater exactness, the experiment should be carried out multiple times and then the average value should be considered. Still, even in case of one run (like in this study) *Time* value may show some approximate trends among the models. It is may be observed from a Table 5.1 that larger architectures mostly takes twice longer time to process the outcome in comparison to their corresponding smaller models. Additionally, embeddings obtained from OpenAI's API need much more time for generation due to the connection to API, while does not seem to give more efficient results.

The next trend, notable from the Table 5.1 is that some *Model Types* have better performance in comparison to the others. For example, in *Triplet Dataset* GPT2 models may reach accuracy nearly 50%, while XLMs' maximum possible accuracy is just 36.9%. The number of models' parameters did not seem to have a notable effect on the results. Probably, this fact may be explained by training dataset of the models. Some of them may be composed of more commonly used ideas that are also present in the input datasets, so even the most basic models may successfully handle a greater number of tokens.

Overall, members of the BERT family and their variations typically exhibit strong performance, often surpassing that of the Random Classifier, with many achieving a maximum accuracy exceeding 40%. It is also crucial to bear in mind that the task at hand lacks a ground truth, meaning the observed results reflect how closely NLP models perform relative to human beings.

[1] In the experiment NVIDIA TESLA P100 GPU accelerator (16 GB) was used. For its more detailed parameters visit `https://www.kaggle.com/`

Finally, another correlation worth noting is observed in the distance calculation method (Cosine and Euclidean). As may be observed from a Table 5.1 maximum accuracy remains consistent within one model and both of these metrics, indicating that they do not significantly influence the final results.

5.2 Repeated Triplets Dataset

In this section some results within *Repeated Triplet Dataset* will be discussed.

Repeated Triplet Dataset is much smaller in comparison to its predecessor (it consist only of 140K odd-one-out human judgement entries in comparison to 4.7M in *Triplet Dataset*). Also, there is nearly 10K of unique triplets and slightly more than 8K of them repeat just 2 or 3 times. The rest of the dataset consist of triplets that have a lot of occurrences (mostly 20 or 100) with different human labels. Therefore, the confidence of choosing the same object in a triplet a few times is slightly higher than 40% - see left graph on Figure 5.1.

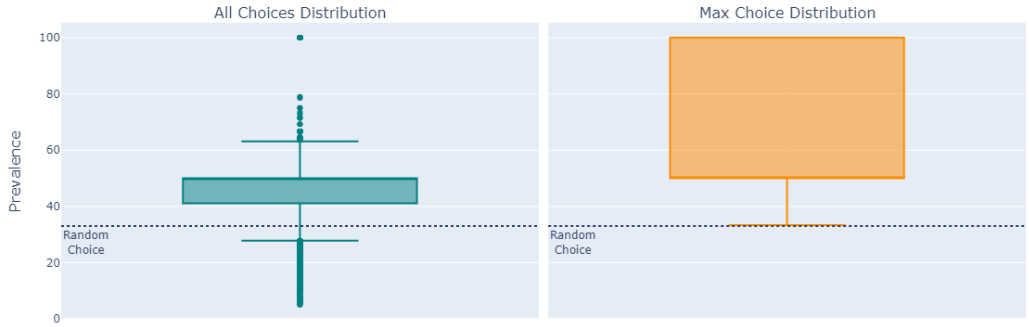


Figure 5.1 The boxplots for repeated human labels confidence prevalence. On the left-hand side of the picture the distribution of choice confidence among all triplet object is shown, while on the other side the distribution of maximum object probabilities is considered.

On the right side of Figure 5.1 the distribution for the concepts with the highest probability is shown. It may be observed that median is still around 50%, so most of the time the most distinct object within the triplet has it confidence rate of 50%.

This fact make the analysis of *Repeated Triplet Dataset* really complicated. Most of the time triplet analysis is complex even for human being, so there is no situation where most of the people choose one concept and the other answers may be considered as some type of ”outlier”. There is no situation, when most of human participant agree to define the same object within the triplet with probability of 90% (in our case it is just 50%, as it was mentioned before), which makes it harder to align with ML approach.

For this reason most of the models have low performance (similar to Random Classifier) within *Repeated Triplet Dataset* (see Figure 5.2) and it is almost impossible to select a preferred one. The worst performance have within all model types have XLM.

Figure 5.2 Comparative percentual accuracy of image captioning models on odd-one-out problem: *Triplet Dataset* (TD) vs. *Repeated Triplet Dataset* (RTD) with the use of cosine (CD) and euclidean (ED) distances as scoring metrics.

Also, in the contrast to the original dataset, any of the captioning methods shows its benefit in general maximum zero-shot accuracy. Neither of them, within the difference between long description or just exact one or two word label, proves to somehow better overall performance.

5.3 The Impact of Image Captioning

In this study all of the image descriptions, generated with the use of 4 different approaches, have been compared in terms of its general performance.

On Figure 5.2 some major correlations in captioning methods in each selected model may be observed. The graph is considered for each model, dataset and distance metric separately, so the final results are more accurate.

Due to the task complexity and the absence of a definitive ground truth, the performance on *Repeated Triplet Dataset* is low regardless any captioning method discussed in Section 4.3. On the other hand, a significant accuracy difference is presented within the original *Triplet Dataset* depending on its captioning. It might be marked that image descriptions generated with the use of GPT-4 model have better performance in comparison to the other methods.

It is important to notice that this captioning approach differs from others in its capacity to generate longer and more detailed image descriptions. While BLIP and ViT models typically produce brief word combinations or small sentences, the GPT-4 model excels at providing extensive details about all the subjects depicted in the image: their textures, colors and the relationships among them. Consequently, with a richer descriptive context, it becomes easier for a NLP model to understand the underlying concept. It is also noteworthy that among the various captioning methods represented in Figure 5.2, none has achieved higher maximum accuracy than the GPT-4 model. Understanding this point is crucial in terms of this study, indicating that while correctly labeling concepts is important, comprehensive image descriptions, even without explicit concept names, prove to be more effective overall.

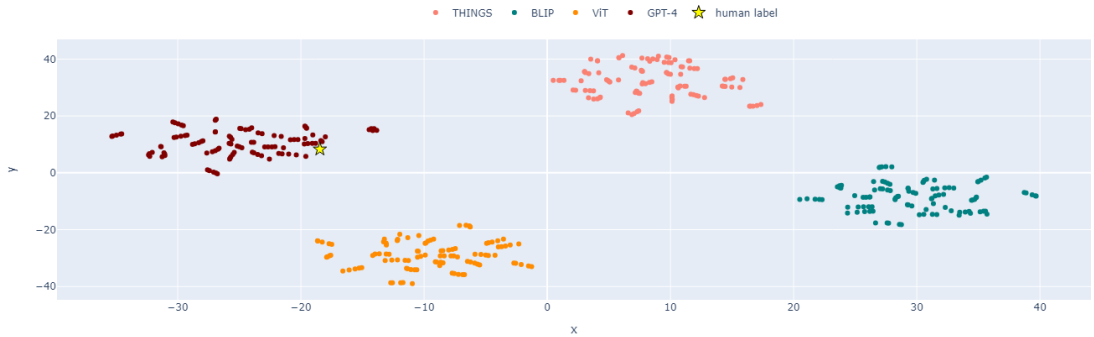


Figure 5.3 Different captioning methods representation in 2D vector space. For this visual representation predictions for random 100K triplets from THINGS dataset [31] have been considered. Yellow star in this graph presents human odd-one-out judgement.

To gain a deeper understanding of the previous statement, a random sample of 100K triplets was selected. Each model odd-one-out prediction for mentioned set of triplets was utilized and combined to a common vector that represents model predicted results. Additionally, human labels for each triplet have been combined into one vector as well and incorporated for comparison purposes. Consequently, a list of vectors (that represent human and AI predictions of odd-one-out in 100K triplets sample) was compiled. To make the next steps more straightforward and not dependent on scoring metric, among all models consider the one, where cosine distance for predictions was used.

In the subsequent step, a dimensionality reduction technique was applied. T-distributed Stochastic Neighbor Embedding (t-SNE) [116] was employed to reduce the input high-dimensional vectors, allowing them to be visualized in a standard 2D vector space. Some of the results of this process are depicted in Figure 5.3. Each point on the graph represents a concept from the THINGS dataset [31], with human label denoted by a "yellow star". Notably, human judgment appears to be closely located to the "GPT" cluster, indicating that human responses exhibit similar behavior to the results obtained using GPT-4 captioning.

5.4 The Impact of Language Model Selection

During this examination 24 language models, described in Section 4.4, have been considered. All of the models have distinct architectures, parameters or training process and part of their performance overview is presented in Table 5.1.

Throughout this examination, while the direct influence of the model's parameters on the final results has not been precisely noticed, it is apparent that the scope and nature of the input training data do exert a significant impact. Models trained on a more extensive and diverse scope of input data (such as GPT2 or XLNET) inherently possess a richer and more nuanced understanding of concepts in general. This broader knowledge enables them to interpret incoming information more correctly, generalize it when it is necessary or discern certain distinctions, thereby enhancing their capacity to accurately identify the most distinct object within a given triplet.

In terms of model size itself, an unconventional observation emerges: larger model architectures do not necessarily guarantee more accurate results compared to human performance. Consider models within GPT2 architecture: gpt2, gpt2-medium, gpt2-large, gpt2-xl. In Table 5.1, their maximum accuracies are outlined, revealing that their average values are quite similar, hovering around 48%, 49.5%, 50%, and 49.3% respectively. This finding suggests that even simpler models can achieve commendable performance, and in some cases, they may generalize even better than their larger counterparts. In the subsequent section, detailed considerations regarding the influence of model layers outcome on overall performance are examined.

Also, it is important to point out that novel approaches produced by OpenAI (GPT Model Type) do not seem to produce better outcome within the examined dataset in comparison to other models. Therefore, in this case this method may not be considered as the most advanced one as it is much more complex in comparison to the others and does not produce results that proves deeper understanding of some basic concepts.

5.4.1 Model Layers Comparison

In Section 4.4, it was noted that, for a deeper understanding of how NLP models interpret input, the last five layers have been specifically examined (except from GPT Model Type due to the OpenAI API limitation: the embedding may be generated only on the final stage - last layer). These layers are denoted as follows: L represents the last layer, $L-1$ is the preceding layer and $L-2$, $L-3$, $L-4$ are labeled accordingly.

In this study, the output of each model layer consists of hidden states for every generated token. To combine these outputs into a main characteristic vector, Average Pooling was employed, averaging across all hidden states. This characteristic vector was then utilized to identify the odd-one-out object within the triplet and then the final accuracy was calculated for each of these layers. This approach empower an investigation for a better understanding the relationship between the layer and the overall performance by putting some light on the dependencies between the two.

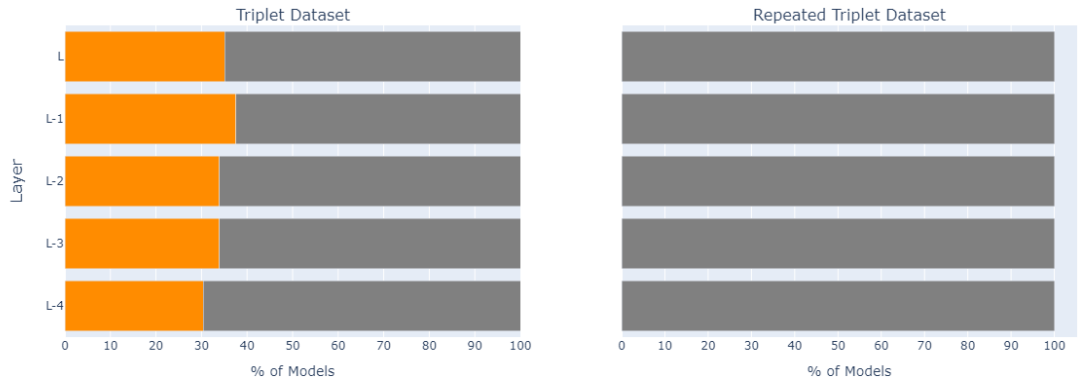


Figure 5.4 Percentage of models with final accuracy higher that 40% depending on the utilized dataset. These models are highlighted in the graph with orange color, when the rest of the models with poorer performance are presented in gray.

Figure 5.4 illustrates some performance results depending on the test dataset and model layer. All of the models performances have been considered and divided into 2 main group: the one that exceeded the accuracy of 40% and the rest models with a lower accuracies. The first group, the main point of interest, is highlighted in the graph with an orange color. On the right side of the graph none of the models are colored in orange, indicating that all of them achieved lower accuracy within *Repeated Triplet Dataset*.

On the other hand, the output from the fifth last layer in 30% of models gives a satisfactory performance (accuracy is higher that 40%). It is crucial to understand that even on such early stages some of the models may emphasize important details for a future possible correct estimation. Even DistilBERT model type, which has relatively smaller architecture and contains just a few layers in general, managed to identify some of important patterns and produce accuracy higher than 40% in 6 out of 16 models (37.5%) (consider the results obtained from DistilBERT models in combination with $L-4$ layer).

Conversely, a positive correlation between general accuracy and layer order can be observed. Accuracy tends to increase slightly within the last few layers. This fact indicates that, most of the time, the model better comprehends the input information and selects necessary features in the later processing stages.

Consider Figure 5.5 that represent a dependency of model accuracy on considered layer and scoring metric. Note that median accuracy for most of the layers is quite similar (differs just in a few percents), while upper quartile increases with layer order. Also, among deeper layers ($L-4$, $L-3$) more precise accuracy (higher that 45%) is more likely to be an outlier (represented as a dot in the graph).

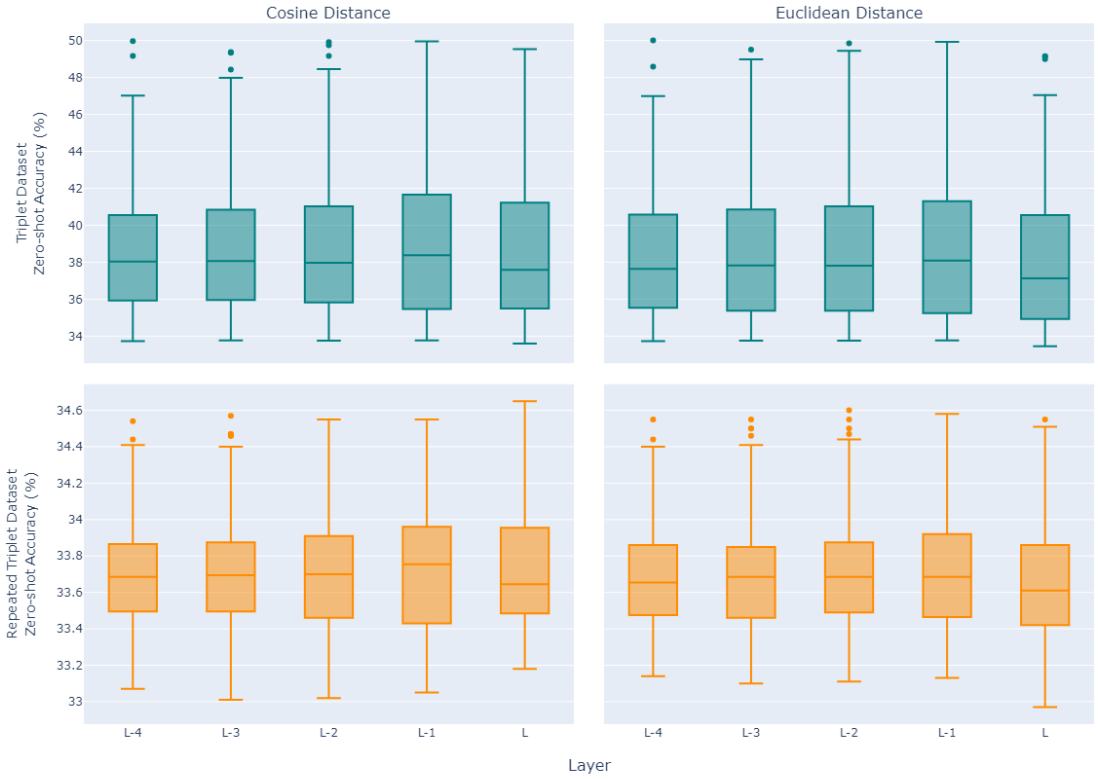


Figure 5.5 The boxplots show the distribution of accuracy of the different captioning methods and LLMs combinations organized by dataset, model layer and zero-shot scoring procedure (Cosine or Euclidean Distances).Legend: symbols on x axis L, $L-1$, $L-2$, $L-3$, $L-4$ represent model layer depth, where L is the output from the last layer

It is crucial to note that pooling technique may also slightly influence the result, so it is important to consider it during the examination.

5.4.2 Vector Scoring Method Comparison

In this study 2 different scoring metric have been considered (Euclidean and Cosine). The first metric examines the line distance among the input vectors, while the other one considers an angle between the objects to understand their similarity (more details have been discussed in Section 4.4.2).

Primary representation of different approaches utilizing both metrics are presented in Table 5.1. The table shows maximum model prediction accuracy depending on the dataset and metric used in the study. Due to the fact that both accuracies within the same dataset and model are quite similar (the difference is around 0.5% in most of the cases), it is possible to make an assumption that the importance of choosing an exact scoring metric does not have a major influence on the final prediction results (additionally, a similar trend can be observed in Figure 5.5). Both scoring metrics perceive multi-dimensional space similarly and can define the most distinct object prediction.

5.5 The Impact of Image Origin

In this stage, 27 overarching image categories were considered. It is important to note that only 1,295 images were classified into one of these categories, while the remaining images belonged to multiple fuzzy categories. Consequently, for a fair evaluation, in the subsequent stages only 1.5 million triplets (comprising images with overarching categories) were taken into account. To further elucidate this process, results obtained using the Cosine distance metric only were considered.

In Figure 5.6, the top 20 most popular triplets are displayed. It is observed that many of these triplets include 'animal' or 'food' items combined with other concepts. Additionally, most triplets consist of items from different categories (gray bars on the graphs) or from the same category (orange color). This complexity complicates the process of selecting the most dissimilar item and results in less clear-cut outputs.

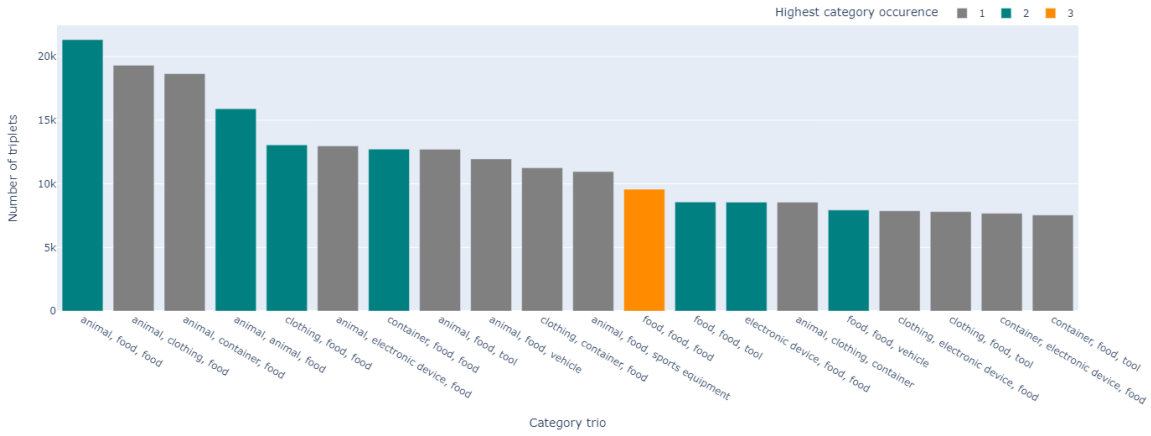


Figure 5.6 The distribution of top 20 most occured triplets and the category distribution within it. All of the input data have been collected from the *THINGS* dataset [31]

During the examination, it was crucial to compare the model's understanding across all categories. To facilitate this comparison, all images were replaced with their respective category names. Subsequently, a new accuracy metric was calculated based solely on whether the model's prediction matched the human-labeled category. The results of this experiment are presented in Table 5.2.

For each model type, a category that appears to be the easiest for the model to understand in a manner similar to humans is highlighted. For example, GPT-2 models perform well with the category "kitchen appliance", while BERT models most accurately recognize "drinks". It is also noteworthy that less common categories, such as "tool" or "container", are challenging for most models to understand. Another interesting observation is the lack of a consistent dependency between the best category performance and the training dataset. Models trained on similar datasets yield distinct outcomes, indicating that the dataset alone does not determine the model's proficiency in understanding specific categories.

Table 5.2 The percentual accuracy for correctly understanding the overarching category is dependency on the model family. This metric represents the average accuracy of comparing whether the odd-one-out choice matches the category of the human label

	ALBERT	BERT	DISTILBERT	GPT	GPT2	ROBERTA	XLM	XLNET
animal	36.40	41.26	41.09	33.22	39.43	37.78	34.26	36.81
body part	41.55	45.30	39.05	33.28	41.75	40.76	35.44	41.48
clothing	41.18	44.74	41.78	33.30	43.91	43.95	36.02	40.98
clothing accessory	42.36	44.59	40.08	33.27	38.91	42.65	36.23	41.56
container	36.40	38.51	37.06	33.35	37.64	36.76	33.78	36.44
dessert	44.02	46.17	43.21	33.38	44.62	44.92	34.69	41.13
drink	38.30	47.06	39.93	33.11	40.84	42.78	33.48	38.96
electronic device	43.52	42.47	44.42	33.40	43.29	41.40	37.29	39.65
food	42.73	45.20	43.96	33.24	43.01	41.21	34.73	41.47
furniture	39.00	41.17	38.74	33.40	41.09	40.40	35.43	38.42
home decor	37.80	40.99	41.15	33.37	39.19	41.61	35.29	38.52
kitchen appliance	42.61	42.18	39.53	33.30	49.28	43.56	38.36	34.78
kitchen tool	38.16	37.84	37.58	33.40	37.52	38.45	32.62	37.11
medical equipment	42.71	43.82	41.73	33.67	41.69	44.00	36.00	38.96
musical instrument	39.69	43.43	41.23	33.29	38.99	40.43	35.93	38.45
office supply	35.39	36.70	34.92	33.30	34.18	33.32	31.42	36.62
part of car	42.13	47.02	45.55	33.29	46.89	44.38	37.52	40.43
plant	43.60	45.86	44.49	33.23	41.99	38.74	33.27	39.86
sports equipment	41.76	43.41	42.40	33.41	38.84	41.06	38.75	37.64
tool	36.71	37.96	35.55	33.40	35.41	36.75	32.94	36.78
toy	37.52	38.79	40.20	33.40	37.28	38.86	34.21	38.27
vehicle	42.95	44.83	45.84	33.46	46.59	42.03	37.93	39.49
weapon	38.19	39.80	38.32	33.52	40.47	39.05	32.40	35.24

In the subsequent evaluation stage, the dependency of model errors on specific categories was examined. Figure 5.7 illustrates this analysis, with human labels represented in each row (the desirable outcome from the language model) and all categories shown in separate columns. The values indicate the probability that the model's selection of the odd-one-out will align with human expectations. In cases where the model's selection does not match the expected category, the figure also shows with which categories the model confused them.

	animal	body part	clothing	clothing accessory	container	dessert	drink	electronic device	food	furniture	home decor	kitchen appliance	kitchen tool	medical equipment	musical instrument	office supply	part of car	plant	sports equipment	tool	toy	vehicle	weapon
animal	40.0	1.1	5.7	1.0	5.8	2.1	1.1	4.7	11.4	2.1	2.1	0.3	1.6	1.3	2.1	1.4	2.0	0.9	3.5	3.8	1.3	3.4	1.4
body part	5.5	41.4	3.2	0.7	5.3	1.9	1.0	4.2	12.0	1.7	2.1	0.3	1.4	0.9	1.7	1.1	1.7	1.4	3.1	3.4	1.3	3.6	1.3
clothing	7.2	0.8	43.7	0.4	4.8	2.0	1.0	3.8	13.6	1.3	1.7	0.3	1.5	0.8	1.6	1.0	1.5	1.5	2.7	3.3	1.1	3.2	1.3
clothing accessory	8.4	1.2	3.0	41.4	4.5	2.1	1.0	3.2	13.8	1.5	1.5	0.2	1.3	0.8	1.3	0.7	1.2	1.6	2.7	2.9	1.1	3.4	1.2
container	8.9	1.9	5.6	0.7	40.4	2.0	0.9	3.1	13.7	1.3	1.7	0.2	1.1	1.0	1.6	0.8	1.5	1.6	3.2	3.0	1.3	3.2	1.3
dessert	7.0	1.7	6.2	0.9	4.4	43.5	0.2	4.1	3.9	1.8	1.9	0.1	0.9	1.3	2.0	1.2	2.1	1.4	3.8	3.8	1.5	4.6	1.7
drink	7.7	1.5	6.3	0.9	3.7	0.7	40.7	4.0	6.6	1.6	1.9	0.1	0.9	1.4	1.9	1.2	2.1	1.4	3.8	3.9	1.6	4.3	1.7
electronic device	8.7	1.9	5.3	0.7	3.7	2.0	0.9	43.4	13.7	1.2	1.7	0.1	1.0	0.9	1.2	0.7	1.0	1.8	2.7	2.6	1.2	2.5	1.1
food	6.4	1.7	6.7	0.9	4.7	0.7	0.3	4.3	46.0	1.9	1.9	0.2	1.0	1.4	2.1	1.2	2.1	1.0	3.8	3.8	1.5	4.6	1.7
furniture	8.8	1.7	4.6	0.6	3.8	2.0	1.1	2.9	14.4	40.0	1.5	0.2	1.3	0.9	1.4	0.8	1.4	1.6	2.9	2.8	1.1	2.9	1.3
home decor	7.9	1.7	5.0	0.6	4.2	1.9	1.0	3.4	12.9	1.2	40.4	0.2	1.3	1.1	1.4	0.9	1.5	1.2	3.1	3.0	1.1	3.4	1.4
kitchen appliance	8.7	1.8	5.7	0.8	3.5	2.0	0.8	1.7	11.8	1.1	1.7	42.3	0.6	1.0	1.4	1.0	1.1	1.5	3.2	2.7	1.5	3.2	1.1
kitchen tool	9.7	2.0	6.4	0.7	3.8	1.9	0.9	2.9	13.1	1.4	1.8	0.1	37.8	0.9	1.3	0.7	1.4	1.8	3.1	2.0	1.3	3.6	1.1
medical equipment	8.3	1.3	4.1	0.6	4.4	2.1	1.1	3.0	13.8	1.4	2.0	0.2	1.1	42.1	1.4	0.7	1.2	1.6	2.3	2.3	1.2	2.8	1.0
musical instrument	8.9	1.8	5.3	0.6	4.3	2.3	1.0	2.5	15.1	1.3	1.5	0.2	1.0	1.0	40.2	0.7	1.3	1.7	2.3	2.4	1.0	2.7	0.9
office supply	9.5	1.9	5.6	0.6	4.4	2.3	1.1	3.0	14.8	1.4	1.8	0.3	1.1	0.9	1.4	35.1	1.4	1.8	3.0	2.4	1.2	3.6	1.2
part of car	8.2	1.5	4.9	0.6	3.9	2.3	1.0	2.2	14.3	1.3	1.7	0.1	1.1	0.9	1.3	0.8	44.2	1.6	2.3	2.4	1.2	1.4	0.9
plant	4.8	1.5	5.9	0.8	5.1	1.8	0.9	4.5	9.1	1.9	1.5	0.3	1.4	1.3	1.9	1.1	1.9	41.7	3.3	3.3	1.4	3.2	1.3
sports equipment	8.2	1.6	4.6	0.6	4.6	2.2	1.1	3.2	14.5	1.5	1.9	0.2	1.2	0.9	1.2	0.8	1.2	1.5	42.0	2.7	0.9	2.4	1.0
tool	9.3	1.9	5.8	0.7	4.4	2.5	1.2	3.1	15.5	1.4	1.8	0.2	0.9	0.8	1.2	0.7	1.3	1.7	2.6	37.8	1.2	3.1	0.8
toy	7.9	1.7	4.8	0.7	4.9	2.0	1.1	3.6	13.8	1.5	1.7	0.3	1.3	1.0	1.4	0.9	1.6	1.6	2.4	3.0	38.5	3.1	1.3
vehicle	7.0	1.7	5.0	0.8	4.3	2.2	1.1	2.7	14.3	1.4	1.9	0.2	1.3	0.9	1.4	1.0	0.7	1.5	2.2	2.8	1.1	43.7	0.9
weapon	8.0	1.7	5.5	0.7	4.7	2.5	1.1	3.1	15.0	1.7	1.9	0.2	1.0	0.9	1.2	0.9	1.3	1.6	2.4	2.2	1.2	2.4	38.6

Figure 5.7 The matrix depicts the possibility (in %) of sucessfully chosen category (represented as columns in a matrix) for each category that was human choice from the triplet (rows). The visualization reveals patterns of different category associations and relative frequencies within the dataset

It is evident that many categories frequently struggle with incorrectly selecting “food“ as the odd-one-out. This may be due to outliers in the data chosen for this examination, either because the “food“ category is more prevalent within the triplets or it contains more images with known classifications. A similar trend can be observed in the “animal“ and “clothing“ categories. These categories are very popular and well-known to most of the models, which increases their likelihood of being selected as the odd-one-out. Despite

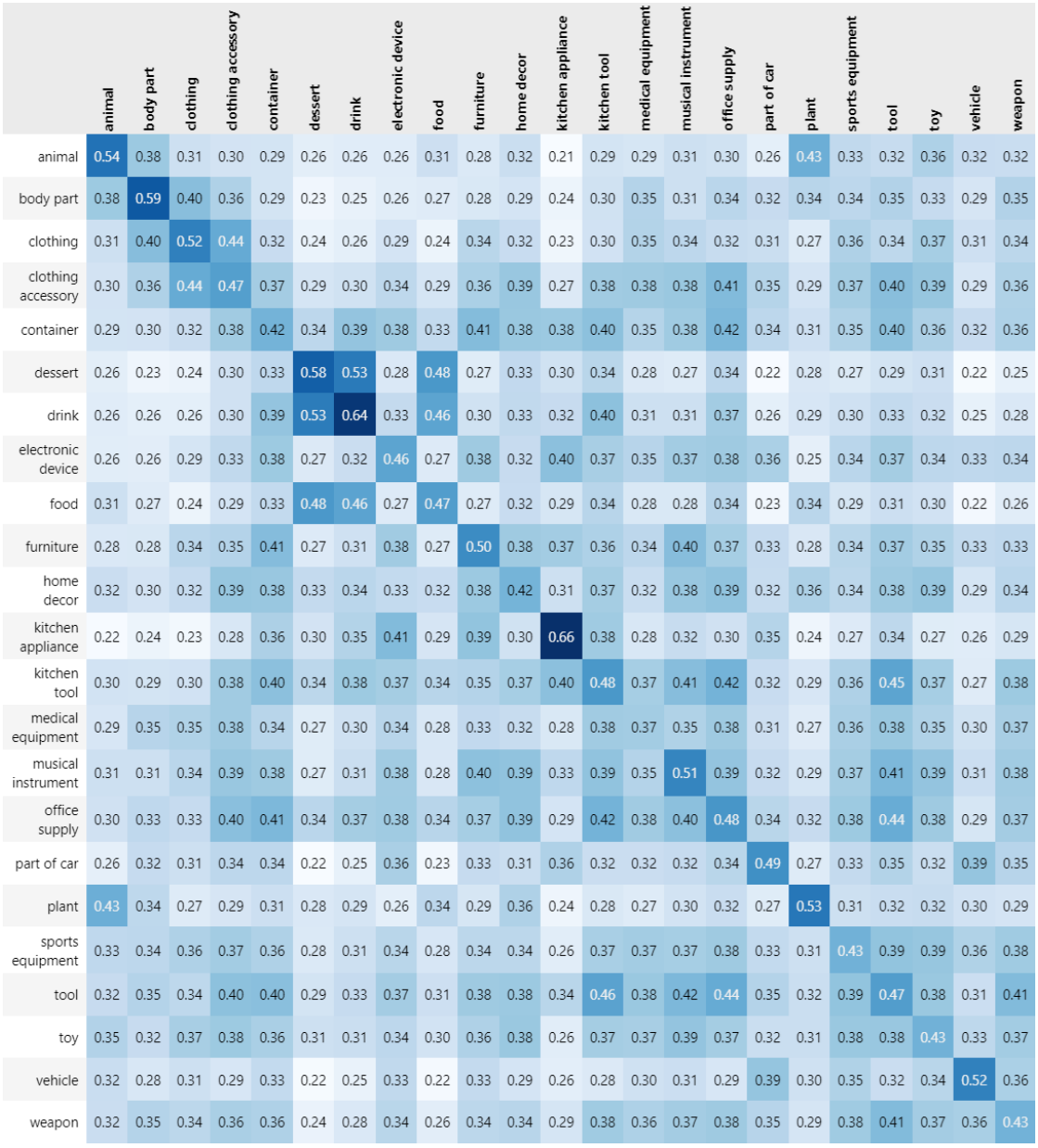
this issue, it is also noticeable that some categories, which are logically similar (such as "part of car" and "vehicle"), are not often confused by the models. This suggests that the models are able to distinguish between closely related categories more effectively than might be expected.

	animal	body part	clothing	clothing accessory	container	dessert	drink	electronic device	food	furniture	home decor	kitchen appliance	kitchen tool	medical equipment	musical instrument	office supply	part of car	plant	sports equipment	tool	toy	vehicle	weapon
animal	0.81	0.40	0.24	0.15	0.17	0.22	0.18	0.11	0.30	0.13	0.21	0.11	0.12	0.16	0.13	0.10	0.13	0.48	0.19	0.16	0.24	0.22	0.21
body part	0.40	0.83	0.56	0.39	0.22	0.24	0.27	0.20	0.23	0.26	0.22	0.11	0.19	0.43	0.25	0.21	0.24	0.24	0.30	0.25	0.32	0.21	0.27
clothing	0.24	0.57	0.76	0.61	0.34	0.20	0.24	0.31	0.16	0.42	0.38	0.24	0.24	0.46	0.36	0.35	0.35	0.23	0.43	0.33	0.42	0.34	0.32
clothing accessory	0.16	0.38	0.61	0.67	0.41	0.20	0.26	0.42	0.17	0.46	0.45	0.31	0.37	0.47	0.48	0.51	0.46	0.23	0.45	0.43	0.46	0.34	0.40
container	0.17	0.22	0.34	0.41	0.52	0.32	0.43	0.46	0.28	0.48	0.43	0.52	0.50	0.40	0.42	0.44	0.42	0.25	0.39	0.43	0.36	0.38	0.37
dessert	0.22	0.23	0.19	0.19	0.31	0.86	0.79	0.23	0.74	0.23	0.27	0.43	0.41	0.18	0.17	0.17	0.13	0.29	0.17	0.16	0.22	0.12	0.11
drink	0.19	0.24	0.24	0.27	0.43	0.79	0.92	0.28	0.68	0.26	0.31	0.51	0.50	0.19	0.25	0.21	0.18	0.28	0.20	0.18	0.23	0.17	0.14
electronic device	0.11	0.19	0.31	0.41	0.46	0.22	0.29	0.68	0.18	0.50	0.39	0.64	0.49	0.42	0.55	0.50	0.57	0.15	0.42	0.48	0.37	0.47	0.44
food	0.30	0.23	0.16	0.18	0.28	0.74	0.68	0.19	0.75	0.18	0.26	0.37	0.39	0.15	0.14	0.16	0.12	0.45	0.16	0.16	0.18	0.11	0.13
furniture	0.13	0.26	0.43	0.45	0.48	0.22	0.28	0.49	0.18	0.72	0.47	0.45	0.41	0.47	0.49	0.47	0.45	0.21	0.43	0.46	0.42	0.42	0.34
home decor	0.21	0.24	0.38	0.44	0.42	0.27	0.33	0.39	0.26	0.49	0.53	0.37	0.37	0.34	0.47	0.42	0.39	0.41	0.37	0.41	0.45	0.32	0.33
kitchen appliance	0.10	0.15	0.25	0.38	0.52	0.41	0.47	0.66	0.38	0.49	0.35	0.82	0.62	0.39	0.42	0.28	0.52	0.16	0.32	0.44	0.26	0.39	0.38
kitchen tool	0.12	0.20	0.24	0.36	0.48	0.42	0.44	0.48	0.38	0.39	0.38	0.61	0.73	0.45	0.51	0.48	0.43	0.21	0.39	0.59	0.37	0.32	0.48
medical equipment	0.16	0.43	0.47	0.49	0.40	0.17	0.18	0.45	0.15	0.45	0.35	0.39	0.44	0.63	0.45	0.48	0.46	0.21	0.50	0.52	0.41	0.42	0.49
musical instrument	0.13	0.25	0.37	0.49	0.43	0.17	0.25	0.55	0.14	0.48	0.48	0.38	0.49	0.45	0.86	0.50	0.50	0.19	0.54	0.54	0.51	0.44	0.53
office supply	0.10	0.21	0.35	0.49	0.45	0.17	0.23	0.50	0.15	0.48	0.42	0.41	0.48	0.50	0.50	0.74	0.45	0.20	0.43	0.57	0.46	0.33	0.42
part of car	0.13	0.22	0.34	0.49	0.42	0.13	0.17	0.57	0.12	0.45	0.36	0.53	0.44	0.47	0.47	0.44	0.76	0.18	0.49	0.51	0.40	0.68	0.49
plant	0.48	0.23	0.23	0.23	0.24	0.30	0.33	0.15	0.45	0.21	0.39	0.15	0.21	0.20	0.19	0.19	0.17	0.87	0.27	0.26	0.25	0.25	0.25
sports equipment	0.18	0.31	0.43	0.47	0.38	0.16	0.21	0.42	0.16	0.42	0.36	0.35	0.39	0.49	0.53	0.42	0.48	0.26	0.68	0.48	0.52	0.50	0.49
tool	0.16	0.25	0.33	0.43	0.43	0.15	0.19	0.47	0.16	0.45	0.41	0.43	0.59	0.52	0.54	0.56	0.51	0.25	0.48	0.68	0.42	0.42	0.58
toy	0.24	0.30	0.42	0.42	0.35	0.21	0.20	0.37	0.18	0.42	0.43	0.33	0.36	0.41	0.50	0.42	0.39	0.25	0.52	0.42	0.60	0.40	0.41
vehicle	0.22	0.21	0.34	0.36	0.39	0.12	0.16	0.47	0.11	0.41	0.33	0.39	0.31	0.42	0.45	0.33	0.69	0.24	0.49	0.41	0.40	0.82	0.51
weapon	0.21	0.26	0.33	0.40	0.36	0.12	0.13	0.44	0.14	0.35	0.33	0.42	0.46	0.47	0.52	0.43	0.53	0.24	0.50	0.58	0.39	0.50	0.72

Figure 5.8 A matrix with color-coded cells representing each category similarity based on human perception

In the following stage, an inverse comparison was undertaken to examine category similarity. Figure 5.8 illustrates the outcomes derived from the analysis of human labels. Each value within the matrix quantifies the perceived similarity between categories based on human judgments. For instance, the analysis of responses to the odd-one-out task

indicates that categories such as “food“, “dessert“ and “drink“ are perceived as quite similar, resulting in elevated similarity scores within the matrix. In contrast, categories like “animal“ and “office supply“ exhibit minimal relatedness, leading to substantially lower values in the corresponding cells. Consequently, the probability of these categories appearing together in the same triplet and being considered relevant or similar is markedly reduced.



	animal	body part	clothing	clothing accessory	container	dessert	drink	electronic device	food	furniture	home decor	kitchen appliance	kitchen tool	medical equipment	musical instrument	office supply	part of car	plant	sports equipment	tool	toy	vehicle	weapon
animal	0.54	0.38	0.31	0.30	0.29	0.26	0.26	0.26	0.31	0.28	0.32	0.21	0.29	0.29	0.31	0.30	0.26	0.43	0.33	0.32	0.36	0.32	0.32
body part	0.38	0.59	0.40	0.36	0.29	0.23	0.25	0.26	0.27	0.28	0.29	0.24	0.30	0.35	0.31	0.34	0.32	0.34	0.34	0.35	0.33	0.29	0.35
clothing	0.31	0.40	0.52	0.44	0.32	0.24	0.26	0.29	0.24	0.34	0.32	0.23	0.30	0.35	0.34	0.32	0.31	0.27	0.36	0.34	0.37	0.31	0.34
clothing accessory	0.30	0.36	0.44	0.47	0.37	0.29	0.30	0.34	0.29	0.36	0.39	0.27	0.38	0.38	0.38	0.41	0.35	0.29	0.37	0.40	0.39	0.29	0.36
container	0.29	0.30	0.32	0.38	0.42	0.34	0.39	0.38	0.33	0.41	0.38	0.38	0.40	0.35	0.38	0.42	0.34	0.31	0.35	0.40	0.36	0.32	0.36
dessert	0.26	0.23	0.24	0.30	0.33	0.58	0.53	0.28	0.48	0.27	0.33	0.30	0.34	0.28	0.27	0.34	0.22	0.28	0.27	0.29	0.31	0.22	0.25
drink	0.26	0.26	0.26	0.30	0.39	0.53	0.64	0.33	0.46	0.30	0.33	0.32	0.40	0.31	0.31	0.37	0.26	0.29	0.30	0.33	0.32	0.25	0.28
electronic device	0.26	0.26	0.29	0.33	0.38	0.27	0.32	0.46	0.27	0.38	0.32	0.40	0.37	0.35	0.37	0.38	0.36	0.25	0.34	0.37	0.34	0.33	0.34
food	0.31	0.27	0.24	0.29	0.33	0.48	0.46	0.27	0.47	0.27	0.32	0.29	0.34	0.28	0.28	0.34	0.23	0.34	0.29	0.31	0.30	0.22	0.26
furniture	0.28	0.28	0.34	0.35	0.41	0.27	0.31	0.38	0.27	0.50	0.38	0.37	0.36	0.34	0.40	0.37	0.33	0.28	0.34	0.37	0.35	0.33	0.33
home decor	0.32	0.30	0.32	0.39	0.38	0.33	0.34	0.33	0.32	0.38	0.42	0.31	0.37	0.32	0.38	0.39	0.32	0.36	0.34	0.38	0.39	0.29	0.34
kitchen appliance	0.22	0.24	0.23	0.28	0.36	0.30	0.35	0.41	0.29	0.39	0.30	0.66	0.38	0.28	0.32	0.30	0.35	0.24	0.27	0.34	0.27	0.26	0.29
kitchen tool	0.30	0.29	0.30	0.38	0.40	0.34	0.38	0.37	0.34	0.35	0.37	0.40	0.48	0.37	0.41	0.42	0.32	0.29	0.36	0.45	0.37	0.27	0.38
medical equipment	0.29	0.35	0.35	0.38	0.34	0.27	0.30	0.34	0.28	0.33	0.32	0.28	0.38	0.37	0.35	0.38	0.31	0.27	0.36	0.38	0.35	0.30	0.37
musical instrument	0.31	0.31	0.34	0.39	0.38	0.27	0.31	0.38	0.28	0.40	0.39	0.33	0.39	0.35	0.51	0.39	0.32	0.29	0.37	0.41	0.39	0.31	0.38
office supply	0.30	0.33	0.33	0.40	0.41	0.34	0.37	0.38	0.34	0.37	0.39	0.29	0.42	0.38	0.40	0.48	0.34	0.32	0.38	0.44	0.38	0.29	0.37
part of car	0.26	0.32	0.31	0.34	0.34	0.22	0.25	0.36	0.23	0.33	0.31	0.36	0.32	0.32	0.32	0.34	0.49	0.27	0.33	0.35	0.32	0.39	0.35
plant	0.43	0.34	0.27	0.29	0.31	0.28	0.29	0.26	0.34	0.29	0.36	0.24	0.28	0.27	0.30	0.32	0.27	0.53	0.31	0.32	0.32	0.30	0.29
sports equipment	0.33	0.34	0.36	0.37	0.36	0.28	0.31	0.34	0.28	0.34	0.34	0.26	0.37	0.37	0.37	0.38	0.33	0.31	0.43	0.39	0.39	0.36	0.38
tool	0.32	0.35	0.34	0.40	0.40	0.29	0.33	0.37	0.31	0.38	0.38	0.34	0.46	0.38	0.42	0.44	0.35	0.32	0.39	0.47	0.38	0.31	0.41
toy	0.35	0.32	0.37	0.38	0.36	0.31	0.31	0.34	0.30	0.36	0.38	0.26	0.37	0.37	0.39	0.37	0.32	0.31	0.38	0.38	0.43	0.33	0.37
vehicle	0.32	0.28	0.31	0.29	0.33	0.22	0.25	0.33	0.22	0.33	0.29	0.26	0.28	0.30	0.31	0.29	0.39	0.30	0.35	0.32	0.34	0.52	0.36
weapon	0.32	0.35	0.34	0.36	0.36	0.24	0.28	0.34	0.26	0.34	0.34	0.29	0.38	0.36	0.37	0.38	0.35	0.29	0.38	0.41	0.37	0.36	0.43

Figure 5.9 A matrix with some color-coded cells that represent each category similarity based on GPT-2 family perception

A similar process was conducted using outcomes from the GPT-2 family of models, specifically chosen for their better performance. Consequently, a second heatmap

representing the model's perception of category similarities is displayed in Figure 5.9.

Upon carefully reviewing the two figures (5.8 and 5.9), it can be observed that the selected LLM model exhibits fewer dependencies within the categories, resulting in a generally lighter table. This indicates that the model perceives separate categories as more distinct compared to human perception. Additionally, there are more pronounced distinctions among similar categories. For example, "kitchen appliance" is perceived as less similar to "kitchen tool", and "vehicle" is less similar to "part of car", compared to the results obtained from human responses. This suggests that the LLM model lacks a nuanced understanding of these concepts.

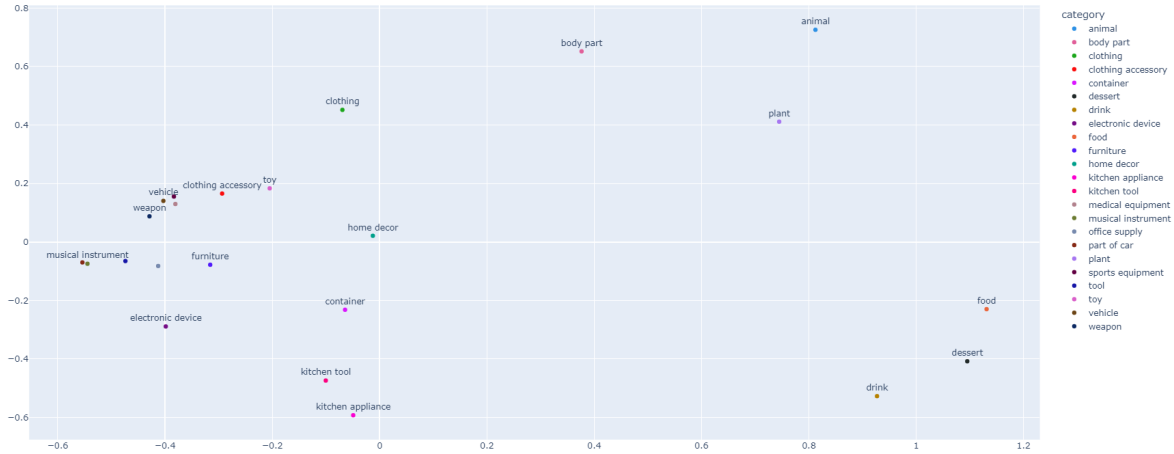


Figure 5.10 The map shows patterns and inner similarity of categories for human perception. It was created with the use of Principal Component Analysis (PCA) method and the similarity matrix where each category is represented as a vector where each dimension equals to similarity to relevant category

Figure 5.11 The map shows patterns and inner similarity of categories for GPT-2 family perception. It was created with the use of Principal Component Analysis (PCA) method and the similarity matrix where each category is represented as a vector where each dimension equals to similarity to relevant category

The final stage involves representing these category similarities in a 2D space. This was achieved by applying the Principal Component Analysis (PCA) dimensionality reduction technique to each row from the two previously mentioned matrices. The results of this analysis are displayed in Figures 5.10 and 5.11.

In this case, it is much more straightforward to notice some dependencies among the proposed categories. Both humans and the model easily categorize and understand the most basic concepts: “food“, “dessert“ and “drink“. These categories are located close to each other in each graph and, at the same time, farther from other concepts. Similarly, “body part“, “animal“ and “plant“ are situated separately and farther from all other points, indicating they are not deeply related to other categories.

However, both methods exhibit problematic points. Notably, the cloth related categories and “vehicle“ are located near each other, suggesting a stronger similarity than expected. A more significant issue is the GPT-2 model’s perception that “toy“ and “medical equipment“ are related. This misconception could lead to problematic outcomes if the model is used without careful oversight.

Chapter 6

CONCLUSION

6.1 Summary

Large Language Models (LLMs) have increasingly become integral to everyday life, significantly enhancing various aspects of human interaction and productivity. Their advanced natural language processing capabilities facilitate seamless communication between humans and machines, empowering people with real-time language translation, personalized digital assistants, tutoring or generating educational content tailored to individual learning styles, automated creative content generation, etc. These models aid in improving accessibility for individuals with disabilities through speech-to-text and text-to-speech functionalities. In the healthcare sector, they assist in diagnosing medical conditions through the analysis of patient data and medical literature.

The adaptability and extensive application potential of LLMs underscore their importance as transformative tools in modern society, driving efficiency and innovation across diverse domains. As previously mentioned, LLMs are utilized in various fields, and thus, it is crucial for them to accurately comprehend the basic concepts that surround us.

In today's technological landscape, numerous types of Large Language Models (LLMs) exist, yet none have fully embodied the ideal of a highly intelligent AI of the new era. Despite their impressive advancements, these models still exhibit notable limitations and disadvantages that hinder their performance. Many LLMs continue to make errors in areas such as semantics, logic and factual accuracy. Additionally, contemporary expectations for AI are evolving; people increasingly desire AI systems that emulate human behavior rather than function as perfectly logical robots. This shift underscores the importance of subjectivity, empathy and nuanced understanding in AI interactions, as human behavior is often neither purely logical nor consistently effective. Consequently, there remains a significant journey ahead in refining LLMs to meet these expectations, involving continuous improvements and breakthroughs in AI development.

To achieve robust solutions using LLMs, proper performance evaluation is essential. The primary goal of such evaluation is to identify the main weaknesses and limitations of current models to enhance their future capabilities. Various metrics have been developed to assess model outputs, checking their robustness and overall performance. Some

evaluation methods have become so popular that entire platforms have been created around them, making these tools accessible to everyone.

Certain approaches leverage other modern technologies and AI to facilitate automated evaluation. These methods use additional AI tools to generate tasks for LLMs and subsequently assess their performance. Despite these advancements, most benchmarks still require significant human interaction and evaluation. This human involvement ensures that the evaluations account for nuances and complexities that automated systems might miss, but it also highlights the need for more sophisticated, integrated evaluation frameworks in the future.

In this study, an advanced strategy for evaluating the performance of Large Language Models (LLMs) was proposed. The main idea is to compare the capabilities of popular models in natural text processing to human performance. To achieve this, basic concepts from everyday life were used as benchmarks. The study assessed the models' understanding of these concepts, their relationships and their similarities and then compared the models' performance to human perception. This approach aims to highlight discrepancies in different spheres and areas for improvement in how LLMs comprehend and process natural language.

To perform the evaluation, the *THINGS* dataset was utilized. This dataset was constructed from a list of the most popular concepts that surround us. From this list, the most popular categories were selected and expanded with precise examples (names of specific items). In later stages, a common visual representation was provided for each item. The final dataset consisted of different image combinations of three items (triplets). The main task was to demonstrate understanding of the selected concepts in each triplet by identifying their similarities and selecting the most dissimilar item (odd-one-out problem). For each combination human answers were provided to facilitate easier evaluation in the subsequent stages.

At the beginning, all images were converted back to text using three different models for image captioning. These models, each with distinct architectures, provided varying descriptions. The BLIP and ViT models generated relatively short descriptions, whereas GPT-4V produced more detailed and longer texts. Additionally, the descriptions were supplemented with the original item names, ensuring clarity and context.

In the next stage, an examination of the current most popular pretrained Large Language Models was conducted. Various models were compared, resulting in a final list of 24 models from 8 model families (Bert, Albert, DistilBert, Roberta, GPT-2, GPT Embedding, XLM, XLNet). To ensure a fair evaluation, the selected models represent a diverse range of types, architectures, sizes and training data.

Then, the input text (image captioning from each approach) was passed to the LLMs without any pretraining. At this point, a zero-shot learning strategy was chosen to assess the models' primary understanding of concepts without any additional fine-tuning. This approach evaluates how well the models can be integrated into basic environments without further modifications. To enhance the robustness of the investigation, the last five hidden states of each model were considered and vector representations (embeddings) of the input text were extracted. These vectors were then compared within each triplet using Cosine and Euclidean similarity measures. The final performance results were based on the evaluation of all triplets.

It is important to mention that all results were compared with human labels, which do not necessarily indicate the ground truth and are not always highly reliable. Consequently, common performance evaluation metrics may not be applicable in this context, necessitating a more flexible and open approach. Additionally, the *THINGS* dataset is designed in such a way that distinguishing the odd-one-out item can be sometimes challenging even for humans, making the task inherently subjective and adding complexity to the comparison of LLMs' performance.

Several key findings from the experiment are detailed in the EXPERIMENTS Section. These insights highlight the strengths and limitations of current LLMs in understanding and processing basic concepts compared to human perception. The results provide a comprehensive overview of how well these models perform in various scenarios, offering valuable information for future improvements and research in the field of Large Language Models.

Overall, the results show that emulating human reasoning in terms of some basic concepts understanding remains a challenging task for the models. Most of them achieve an accuracy between 40% and 50%, which is better than a Random Classifier (with an accuracy of 33% for a triplet problem) but still far from ideal. This means that, on average, **only** 40% **to** 50% **of the triplet answers matched those labeled by humans** (noting that this does not necessarily indicate the ground truth).

Still, the analysis reveals that in cases where the triplet comparison was not straightforward, **people often were uncertain of their answers as well**. The input dataset includes several examples of triplets that were reviewed multiple times, with each person providing a different answer for the most dissimilar object. Consequently, the average confidence rate for selecting the right odd-one-out item is approximately 40%.

Additionally, a more detailed analysis indicates that models **receiving longer and more detailed descriptions from the GPT-4V model generally yield better results**. Their average accuracy in selecting the correct odd-one-out is higher. In most cases, longer descriptions are more effective than the exact item names generated from the *THINGS* dataset. This suggests that, in terms of this comparison, the models resemble human behavior: the more explanation they receive, the easier it is for them to understand each concept deeply and make similar decisions.

In terms of different LLM architectures, there are notable distinctions as well. Firstly, larger model architectures require more time and computational resources to extract embeddings and use them for item comparisons. However, a larger model size does not necessarily guarantee more precise results. The analysis did not reveal any other dependencies, such as those based on the training dataset or number of hidden states. Interestingly, models from the **GPT-2 family generally exhibited the most effective performance** (see Table 5.1).

Lastly, after considering image categories, several conclusions can be drawn. First, the training dataset does not significantly help the model understand certain types of data better. **Nuanced relationships among categories remain challenging for the model to define**, leading to decision-making based on a more polarized perception. Additionally, both approaches easily grasp some simple categories, while more complex ones may be misconceived by LLMs, potentially leading to serious issues.

The results of this experiment make it easier to determine if a selected model is suitable for the task that the LLM is supposed to perform. They also help identify the model's main weaknesses, allowing for targeted improvements during fine-tuning and deployment phases. Understanding these aspects ensures that the model can be optimized for better performance in its intended application.

6.2 Further Enhancements

The examination encompasses a comprehensive comparison of various models, evaluating multiple aspects of Large Language Model (LLM) performance. This includes assessing the models' ability to comprehend fundamental concepts without any fine-tuning, their overall performance and their similarity to human perception. Additionally, the examination considers the impact of different input types and the influence of architectural variations on model performance.

The evaluation process is inherently complex, involving numerous metrics and criteria to provide a holistic assessment of each model's capabilities. Despite the rigorous nature of this evaluation, there is always room for refinement and the introduction of new methodologies. Continuous advancements in the field necessitate the ongoing development of evaluation techniques to ensure they remain robust and relevant.

Improvements to the evaluation process may encompass a variety of enhancements. These include the incorporation of additional metrics, the use of more diverse datasets and the inclusion of a broader range of captioning methods or models for text processing. Furthermore, considering even deeper layers within the models and applying additional scoring metrics can provide a more nuanced understanding of their capabilities.

There is always the potential to improve the input dataset of concepts and triplets. Enhancements can include adding new features and descriptions to the items, implementing support for additional languages, and making the dataset more robust by repeatedly verifying the correctness of the answers. These improvements can lead to a more comprehensive and accurate dataset, ultimately contributing to better model understanding and evaluation.

Testing alternative calculation techniques could also yield valuable insights. For instance, instead of relying on average pooling applied to the last hidden state to obtain phrase representations, other pooling methods or classification vectors could be employed. This could involve experimenting with max pooling, attention-based pooling or other sophisticated aggregation methods that might capture different aspects of the data more effectively.

Moreover, models can be slightly modified through fine-tuning to better align with specific requirements. This approach suggests that their performance with few-shot learning is likely to improve. However, several questions arise regarding the extent of effort required to achieve satisfactory outcomes.

One important consideration is determining the level of fine-tuning necessary to produce a model that performs well for a given task. This involves assessing the trade-off between the amount of data and computational resources needed for fine-tuning versus the performance gains achieved. Evaluating different fine-tuning strategies, such as transfer learning, domain adaptation and task-specific adjustments, can provide insights into the most efficient methods for enhancing model performance. Additionally, the

effectiveness of providing models with a clear outline of the main general categories needs to be examined.

Another potential enhancement involves automating the evaluation strategy and leveraging other available AI tools. This can facilitate the generation of evaluation tasks, setting up frameworks for real-time competitions among models and systematically assess the behavior and performance of the tested LLMs. Automating these processes may significantly increase the efficiency and scalability of the evaluation, ensuring consistent, robust and objective comparisons while reducing costs by minimizing human intervention.

At the end, it also makes sense to perform the same evaluation and comparison of models' understanding of concepts by solving a different type of problem, rather than relying solely on the odd-one-out approach. This could involve using tasks such as analogy completion, concept categorization or contextual reasoning, providing a more comprehensive assessment of the models' capabilities.

By integrating these improvements, the evaluation framework can be enhanced to provide a more comprehensive and detailed assessment of LLM performance and better capture the nuances of human language understanding and generation. This continuous refinement is crucial for keeping pace with the rapid advancements in the field of natural language processing, ensuring that evaluation techniques remain robust, relevant and capable of identifying the most effective models and methodologies.

Bibliography

1. BALLY, Charles; SECHEHAYE, Albert. *Course in general linguistics Ferdinand de Saussure*. McGraw-Hill Book Company, 1966.
2. WEIZENBAUM, Joseph. ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*. 1966, vol. 9, no. 1, pp. 36–45.
3. COLLINS, Michael. Statistical machine translation: IBM models 1 and 2. *Columbia Columbia Univ*. 2011.
4. TARWANI, Kanchan M; EDEM, Swathi. Survey on recurrent neural network in natural language processing. *Int. J. Eng. Trends Technol.* 2017, vol. 48, no. 6, pp. 301–304.
5. PATWARDHAN, Narendra; MARRONE, Stefano; SANSONE, Carlo. Transformers in the real world: A survey on nlp applications. *Information*. 2023, vol. 14, no. 4, p. 242.
6. GILLIOZ, Anthony; CASAS, Jacky; MUGELLINI, Elena; ABOU KHALED, Omar. Overview of the Transformer-based Models for NLP Tasks. In: *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2020, pp. 179–183.
7. MINAEE, Shervin; MIKOLOV, Tomas; NIKZAD, Narjes; CHENAGHLU, Meysam; SOCHER, Richard; AMATRIAIN, Xavier; GAO, Jianfeng. Large language models: A survey. *arXiv preprint arXiv:2402.06196*. 2024.
8. THIRUNAVUKARASU, Arun James; TING, Darren Shu Jeng; ELANGOVAN, Kabilan; GUTIERREZ, Laura; TAN, Ting Fang; TING, Daniel Shu Wei. Large language models in medicine. *Nature medicine*. 2023, vol. 29, no. 8, pp. 1930–1940.
9. BROWN, Tom; MANN, Benjamin; RYDER, Nick; SUBBIAH, Melanie; KAPLAN, Jared D; DHARIWAL, Prafulla; NEELAKANTAN, Arvind; SHYAM, Pranav; SASTRY, Girish; ASKELL, Amanda, et al. Language models are few-shot learners. *Advances in neural information processing systems*. 2020, vol. 33, pp. 1877–1901.

10. DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. 2018.

11. KOROTEEV, MV. BERT: a review of applications in natural language processing and understanding. *arXiv preprint arXiv:2103.11943*. 2021.

12. RAVICHANDIRAN, Sudharsan. *Getting Started with Google BERT: Build and train state-of-the-art natural language processing models using BERT*. Packt Publishing Ltd, 2021.

13. RADFORD, Alec; WU, Jeffrey; AMODEI, Dario; AMODEI, Daniela; CLARK, Jack; BRUNDAGE, Miles; SUTSKEVER, Ilya. Better language models and their implications. *OpenAI blog*. 2019, vol. 1, no. 2.

14. KALYAN, Katikapalli Subramanyam. A survey of GPT-3 family large language models including ChatGPT and GPT-4. *Natural Language Processing Journal*. 2023, p. 100048.

15. ROUMELIOTIS, Konstantinos I; TSELIKAS, Nikolaos D. Chatgpt and open-ai models: A preliminary review. *Future Internet*. 2023, vol. 15, no. 6, p. 192.

16. LEWIS, Mike; LIU, Yinhan; GOYAL, Naman; GHAZVININEJAD, Marjan; MOHAMED, Abdelrahman; LEVY, Omer; STOYANOV, Ves; ZETTLEMOYER, Luke. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*. 2019.

17. RAY, Partha Pratim. ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems*. 2023.

18. PICHAI, Sundar; HASSABIS, Demis. Introducing Gemini: our largest and most capable AI model. *Google. Retrieved December*. 2023, vol. 8, p. 2023.

19. ZHAO, Haiyan; CHEN, Hanjie; YANG, Fan; LIU, Ninghao; DENG, Huiqi; CAI, Hengyi; WANG, Shuaiqiang; YIN, Dawei; DU, Mengnan. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*. 2023.

20. CHANG, Yupeng; WANG, Xu; WANG, Jindong; WU, Yuan; YANG, Linyi; ZHU, Kaijie; CHEN, Hao; YI, Xiaoyuan; WANG, Cunxiang; WANG, Yidong, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*. 2024, vol. 15, no. 3, pp. 1–45.

21. KASNECI, Enkelejda; SESSLER, Kathrin; KÜCHEMANN, Stefan; BANNERT, Maria; DEMENTIEVA, Daryna; FISCHER, Frank; GASSER, Urs; GROH, Georg; GÜNNEMANN, Stephan; HÜLLERMEIER, Eyke, et al. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and individual differences*. 2023, vol. 103, p. 102274.

22. CARLINI, Nicholas; NASR, Milad; CHOQUETTE-CHOO, Christopher A; JAGIELSKI, Matthew; GAO, Irena; KOH, Pang Wei W; IPPOLITO, Daphne; TRAMER, Florian; SCHMIDT, Ludwig. Are aligned neural networks adversarially aligned? *Advances in Neural Information Processing Systems*. 2024, vol. 36.

23. WANG, Boshi; YUE, Xiang; SUN, Huan. Can ChatGPT defend its belief in truth? evaluating LLM reasoning via debate. In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. 2023, pp. 11865–11881.
24. WANG, Yidong; YU, Zhuohao; ZENG, Zhengran; YANG, Linyi; WANG, Cunxiang; CHEN, Hao; JIANG, Chaoya; XIE, Rui; WANG, Jindong; XIE, Xing, et al. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. *arXiv preprint arXiv:2306.05087*. 2023.
25. LI, Jiwei; CHEN, Xinlei; HOVY, Eduard; JURAFSKY, Dan. Visualizing and understanding neural models in NLP. *arXiv preprint arXiv:1506.01066*. 2015.
26. WU, Zhiyong; CHEN, Yun; KAO, Ben; LIU, Qun. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. *arXiv preprint arXiv:2004.14786*. 2020.
27. JAWAHAR, Ganesh; SAGOT, Benoıt; SEDDAH, Djamé. What does BERT learn about the structure of language? In: *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*. 2019.
28. PETERS, Matthew E; NEUMANN, Mark; ZETTLEMOYER, Luke; YIH, Wen-tau. Dissecting contextual word embeddings: Architecture and representation. *arXiv preprint arXiv:1808.08949*. 2018.
29. KOKALJ, Enja; ŠKRLJ, Blaž; LAVRAČ, Nada; POLLAK, Senja; ROBNIK-ŠIKONJA, Marko. BERT meets shapley: Extending SHAP explanations to transformer-based classifiers. In: *Proceedings of the EACL Hackashop on News Media Content Analysis and Automated Report Generation*. 2021, pp. 16–21.
30. MUTTENTHALER, Lukas; DIPPEL, Jonas; LINHARDT, Lorenz; VANDERMEULEN, Robert A; KORNBLITH, Simon. Human alignment of neural network representations. *arXiv preprint arXiv:2211.01201*. 2022.
31. HEBART, Martin N; DICKTER, Adam H; KIDDER, Alexis; KWOK, Wan Y; CORRIVEAU, Anna; VAN WICKLIN, Caitlin; BAKER, Chris I. THINGS: A database of 1,854 object concepts and more than 26,000 naturalistic object images. *PloS one*. 2019, vol. 14, no. 10, e0223792.
32. STOINSKI, Laura M; PERKUHN, Jonas; HEBART, Martin N. THINGSplus: New norms and metadata for the THINGS database of 1854 object concepts and 26,107 natural object images. *Behavior Research Methods*. 2023, pp. 1–21.
33. SINAPOV, Jivko; STOYTCHEV, Alexander. The odd one out task: Toward an intelligence test for robots. In: *2010 IEEE 9th International Conference on Development and Learning*. IEEE, 2010, pp. 126–131.
34. SCHÖNMANN, Inés; KÜNSTLE, David-Elias; WICHMANN, Felix A. Using an Odd-One-Out Design Affects Consistency, Agreement and Decision Criteria in Similarity Judgement Tasks Involving Natural Images. *Journal of Vision*. 2022, vol. 22, no. 14, pp. 3232–3232.
35. STRINGHAM, Nathan; IZBICKI, Mike. Evaluating word embeddings on low-resource languages. In: *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*. 2020, pp. 176–186.

36. MOHAMMADI, Salman; UHRENHOLT, Anders Kirk; JENSEN, Bjørn Sand. Odd-one-out representation learning. *arXiv preprint arXiv:2012.07966*. 2020.

37. RUIZ, Philippe E. Building and solving odd-one-out classification problems: A systematic approach. *Intelligence*. 2011, vol. 39, no. 5, pp. 342–350.

38. LAROCHELLE, Hugo; ERHAN, Dumitru; BENGIO, Yoshua. Zero-data learning of new tasks. In: *AAAI*. 2008, vol. 1, p. 3. No. 2.

39. WANG, Wei; ZHENG, Vincent W; YU, Han; MIAO, Chunyan. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*. 2019, vol. 10, no. 2, pp. 1–37.

40. POURPANAH, Farhad; ABDAR, Moloud; LUO, Yuxuan; ZHOU, Xinlei; WANG, Ran; LIM, Chee Peng; WANG, Xi-Zhao; WU, QM Jonathan. A review of generalized zero-shot learning methods. *IEEE transactions on pattern analysis and machine intelligence*. 2022, vol. 45, no. 4, pp. 4051–4070.

41. CHALKIDIS, Ilias; FERGADIOTIS, Manos; KOTITSAS, Sotiris; MALAKASIOTIS, Prodromos; ALETRAS, Nikolaos; ANDROUTSOPOULOS, Ion. An empirical study on large-scale multi-label text classification including few and zero-shot labels. *arXiv preprint arXiv:2010.01653*. 2020.

42. PELICON, Andraž; PRANJIĆ, Marko; MILJKOVIĆ, Dragana; ŠKRLJ, Blaž; POLLAK, Senja. Zero-shot learning for cross-lingual news sentiment classification. *Applied Sciences*. 2020, vol. 10, no. 17, p. 5993.

43. KANG, Hyunwook; HAZARIKA, Devamanyu; KIM, Dongho; KIM, Jihie. Zero-Shot Visual Emotion Recognition by Exploiting BERT. In: *Proceedings of SAI Intelligent Systems Conference*. Springer, 2022, pp. 485–494.

44. SIVARAJKUMAR, Sonish; WANG, Yanshan. Healthprompt: A zero-shot learning paradigm for clinical natural language processing, 2022. *URL https://arxiv.org/abs/2203.05061*. [N.d.].

45. CHEN, Qi; WANG, Wei; HUANG, Kaizhu; COENEN, Frans. Zero-shot text classification via knowledge graph embedding for social media data. *IEEE Internet of Things Journal*. 2021, vol. 9, no. 12, pp. 9205–9213.

46. SUN, Xiaohong; GU, Jinan; SUN, Hongying. Research progress of zero-shot learning. *Applied Intelligence*. 2021, vol. 51, pp. 3600–3614.

47. HERDADE, Simao; KAPPELER, Armin; BOAKYE, Kofi; SOARES, Joao. Image captioning: Transforming objects into words. *Advances in neural information processing systems*. 2019, vol. 32.

48. SHARMA, Himanshu; AGRAHARI, Manmohan; SINGH, Sujeet Kumar; FIROJ, Mohd; MISHRA, Ravi Kumar. Image captioning: a comprehensive survey. In: *2020 International Conference on Power Electronics & IoT Applications in Renewable Energy and its Control (PARC)*. IEEE, 2020, pp. 325–328.

49. LI, Zewen; LIU, Fan; YANG, Wenjie; PENG, Shouheng; ZHOU, Jun. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*. 2021, vol. 33, no. 12, pp. 6999–7019.

50. SHERSTINSKY, Alex. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*. 2020, vol. 404, p. 132306.
51. DING, Zihan; HUANG, Yanhua; YUAN, Hang; DONG, Hao. Introduction to reinforcement learning. *Deep reinforcement learning: fundamentals, research and applications*. 2020, pp. 47–123.
52. REITER, Ehud; BELZ, Anja. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*. 2009, vol. 35, no. 4, pp. 529–558.
53. WANG, Haoran; ZHANG, Yue; YU, Xiaosheng. An overview of image caption generation methods. *Computational intelligence and neuroscience*. 2020, vol. 2020.
54. FENG, Yang; MA, Lin; LIU, Wei; LUO, Jiebo. Unsupervised image captioning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4125–4134.
55. MATHUR, Pratistha et al. A survey on various deep learning models for automatic image captioning. In: *Journal of Physics: Conference Series*. IOP Publishing, 2021, vol. 1950, p. 012045. No. 1.
56. CHANG, Yeong-Hwa; CHEN, Yen-Jen; HUANG, Ren-Hung; YU, Yi-Ting. Enhanced image captioning with color recognition using deep learning methods. *Applied Sciences*. 2021, vol. 12, no. 1, p. 209.
57. IYER, Sethurathienam; CHATURVEDI, Shubham; DASH, Tirtharaj. Image captioning-based image search engine: An alternative to retrieval by metadata. In: *Soft Computing for Problem Solving: SocProS 2017, Volume 2*. Springer, 2019, pp. 181–191.
58. AHSAN, Hiba; BHALLA, Nikita; BHATT, Daivat; SHAH, Kaivankumar. Multi-modal image captioning for the visually impaired. *arXiv preprint arXiv:2105.08106*. 2021.
59. IMAI, Toru; HOMMA, Shinichi; KOBAYASHI, Akio; OKU, Takahiro; SATO, Shoei. Speech recognition with a seamlessly updated language model for real-time closed-captioning. In: *Eleventh Annual Conference of the International Speech Communication Association*. 2010.
60. GUPTA, Sonal; MOONEY, Raymond. Using closed captions as supervision for video activity recognition. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2010, vol. 24, pp. 1083–1088. No. 1.
61. WIENEKE, Arika E; BOWLES, Erin JA; CRONKITE, David; WERNLI, Karen J; GAO, Hongyuan; CARRELL, David; BUIST, Diana SM. Validation of natural language processing to extract breast cancer pathology procedures and results. *Journal of pathology informatics*. 2015, vol. 6, no. 1, p. 38.
62. MORANTE, Roser; BLANCO, Eduardo. Recent advances in processing negation. *Natural Language Engineering*. 2021, vol. 27, no. 2, pp. 121–130.

63. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N; KAISER, Łukasz; POLOSUKHIN, Illia. Attention is all you need. *Advances in neural information processing systems*. 2017, vol. 30.

64. RADFORD, Alec; WU, Jeffrey; CHILD, Rewon; LUAN, David; AMODEI, Dario; SUTSKEVER, Ilya, et al. Language models are unsupervised multitask learners. *OpenAI blog*. 2019, vol. 1, no. 8, p. 9.

65. LIU, Yinhan; OTT, Myle; GOYAL, Naman; DU, Jingfei; JOSHI, Mandar; CHEN, Danqi; LEVY, Omer; LEWIS, Mike; ZETTLEMOYER, Luke; STOYANOV, Veselin. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*. 2019.

66. LAN, Zhenzhong; CHEN, Mingda; GOODMAN, Sebastian; GIMPEL, Kevin; SHARMA, Piyush; SORICUT, Radu. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*. 2019.

67. WEBSTER, Jonathan J; KIT, Chunyu. Tokenization as the initial phase in NLP. In: *COLING 1992 volume 4: The 14th international conference on computational linguistics*. 1992.

68. SONG, Xinying; SALCIANU, Alex; SONG, Yang; DOPSON, Dave; ZHOU, Denny. Fast wordpiece tokenization. *arXiv preprint arXiv:2012.15524*. 2020.

69. LI, Yang; YANG, Tao. Word embedding for understanding natural language: a survey. *Guide to big data applications*. 2018, pp. 83–104.

70. LAI, Siwei; LIU, Kang; HE, Shizhu; ZHAO, Jun. How to generate a good word embedding. *IEEE Intelligent Systems*. 2016, vol. 31, no. 6, pp. 5–14.

71. MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. 2013.

72. HOFMANN, Valentin; PIERREHUMBERT, Janet B; SCHÜTZE, Hinrich. Dynamic contextualized word embeddings. *arXiv preprint arXiv:2010.12684*. 2020.

73. PENNINGTON, Jeffrey; SOCHER, Richard; MANNING, Christopher D. Glove: Global vectors for word representation. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

74. ATHIWARATKUN, Ben; WILSON, Andrew Gordon; ANANDKUMAR, Anima. Probabilistic fasttext for multi-sense word embeddings. *arXiv preprint arXiv:1806.02901*. 2018.

75. GUO, Meng-Hao; XU, Tian-Xing; LIU, Jiang-Jiang; LIU, Zheng-Ning; JIANG, Peng-Tao; MU, Tai-Jiang; ZHANG, Song-Hai; MARTIN, Ralph R; CHENG, Ming-Ming; HU, Shi-Min. Attention mechanisms in computer vision: A survey. *Computational visual media*. 2022, vol. 8, no. 3, pp. 331–368.

76. BAAN, Joris; HOEVE, Maartje ter; WEES, Marlies van der; SCHUTH, Anne; RIJKE, Maarten de. Understanding multi-head attention in abstractive summarization. *arXiv preprint arXiv:1911.03898*. 2019.

77. GLOROT, Xavier; BENGIO, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

78. YAO, Benjamin Z; YANG, Xiong; LIN, Liang; LEE, Mun Wai; ZHU, Song-Chun. I2t: Image parsing to text description. *Proceedings of the IEEE*. 2010, vol. 98, no. 8, pp. 1485–1508.

79. FU, Kun; JIN, Junqi; CUI, Runpeng; SHA, Fei; ZHANG, Changshui. Aligning where to see and what to tell: Image captioning with region-based attention and scene-specific contexts. *IEEE transactions on pattern analysis and machine intelligence*. 2016, vol. 39, no. 12, pp. 2321–2334.

80. SOCHER, Richard; FEI-FEI, Li. Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 966–973.

81. KUZNETSOVA, Polina; ORDONEZ, Vicente; BERG, Tamara L; CHOI, Yejin. Treetalk: Composition and compression of trees for image descriptions. *Transactions of the Association for Computational Linguistics*. 2014, vol. 2, pp. 351–362.

82. ORDONEZ, Vicente; KULKARNI, Girish; BERG, Tamara. Im2text: Describing images using 1 million captioned photographs. *Advances in neural information processing systems*. 2011, vol. 24.

83. DEVLIN, Jacob; GUPTA, Saurabh; GIRSHICK, Ross; MITCHELL, Margaret; ZITNICK, C Lawrence. Exploring nearest neighbor approaches for image captioning. *arXiv preprint arXiv:1505.04467*. 2015.

84. DONAHUE, Jeffrey; ANNE HENDRICKS, Lisa; GUADARRAMA, Sergio; ROHRBACH, Marcus; VENUGOPALAN, Subhashini; SAENKO, Kate; DARRELL, Trevor. Long-term recurrent convolutional networks for visual recognition and description. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 2625–2634.

85. JOHNSON, Justin; KARPATHY, Andrej; FEI-FEI, Li. Densecap: Fully convolutional localization networks for dense captioning. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4565–4574.

86. YAO, Ting; PAN, Yingwei; LI, Yehao; MEI, Tao. Exploring visual relationship for image captioning. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 684–699.

87. YANG, Xu; TANG, Kaihua; ZHANG, Hanwang; CAI, Jianfei. Auto-encoding scene graphs for image captioning. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 10685–10694.

88. LIU, Huan; WANG, Guangbin; HUANG, Ting; HE, Ping; SKITMORE, Martin; LUO, Xiaochun. Manifesting construction activity scenes via image captioning. *Automation in Construction*. 2020, vol. 119, p. 103334.

89. YANG, Zhenyu; LIU, Qiao. Att-bm-som: A framework of effectively choosing image information and optimizing syntax for image captioning. *IEEE Access*. 2020, vol. 8, pp. 50565–50573.

90. YENDURI, Gokul; RAMALINGAM, M; SELVI, G Chemmalar; SUPRIYA, Y; SRIVASTAVA, Gautam; MADDIKUNTA, Praveen Kumar Reddy; RAJ, G Deepti; JHAVERI, Rutvij H; PRABADEVI, B; WANG, Weizheng, et al. GPT (Generative Pre-trained Transformer)–A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions. *IEEE Access*. 2024.

91. GUO, Longteng; LIU, Jing; ZHU, Xinxin; YAO, Peng; LU, Shichen; LU, Hanqing. Normalized and geometry-aware self-attention network for image captioning. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 10327–10336.

92. CORNIA, Marcella; STEFANINI, Matteo; BARALDI, Lorenzo; CUCCHIARA, Rita. Meshed-memory transformer for image captioning. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 10578–10587.

93. PAN, Yingwei; YAO, Ting; LI, Yehao; MEI, Tao. X-linear attention networks for image captioning. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 10971–10980.

94. LIU, Wei; CHEN, Sihan; GUO, Longteng; ZHU, Xinxin; LIU, Jing. Cptr: Full transformer network for image captioning. *arXiv preprint arXiv:2101.10804*. 2021.

95. TOUVRON, Hugo; CORD, Matthieu; DOUZE, Matthijs; MASSA, Francisco; SABLAYROLLES, Alexandre; JÉGOU, Hervé. Training data-efficient image transformers & distillation through attention. In: *International conference on machine learning*. PMLR, 2021, pp. 10347–10357.

96. CHANG, Yupeng; WANG, Xu; WANG, Jindong; WU, Yuan; YANG, Linyi; ZHU, Kaijie; CHEN, Hao; YI, Xiaoyuan; WANG, Cunxiang; WANG, Yidong, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*. 2023.

97. PETERS, Matthew E; NEUMANN, Mark; IYYER, Mohit; GARDNER, Matt; CLARK, Christopher; LEE, Kenton; ZETTLEMOYER, Luke. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*. 2018.

98. HOWARD, Jeremy; RUDER, Sebastian. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*. 2018.

99. SANH, Victor; DEBUT, L; CHAUMOND, J; WOLF, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. arXiv 2019. *arXiv preprint arXiv:1910.01108*. 2019.

100. TEAM, Gemini; ANIL, Rohan; BORGEAUD, Sebastian; WU, Yonghui; ALAYRAC, Jean-Baptiste; YU, Jiahui; SORICUT, Radu; SCHALKWYK, Johan; DAI, Andrew M; HAUTH, Anja, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*. 2023.

101. TOUVRON, Hugo; LAVRIL, Thibaut; IZACARD, Gautier; MARTINET, Xavier; LACHAUX, Marie-Anne; LACROIX, Timothée; ROZIÈRE, Baptiste; GOYAL, Naman; HAMBRO, Eric; AZHAR, Faisal, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*. 2023.

102. ANTHROPIC, AI. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*. 2024, vol. 1.

103. OUYANG, Long; WU, Jeffrey; JIANG, Xu; ALMEIDA, Diogo; WAINWRIGHT, Carroll; MISHKIN, Pamela; ZHANG, Chong; AGARWAL, Sandhini; SLAMA, Katarina; RAY, Alex, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*. 2022, vol. 35, pp. 27730–27744.

104. SCHULMAN, John; WOLSKI, Filip; DHARIWAL, Prafulla; RADFORD, Alec; KLIMOV, Oleg. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*. 2017.

105. RAFAILOV, Rafael; SHARMA, Archit; MITCHELL, Eric; MANNING, Christopher D; ERMON, Stefano; FINN, Chelsea. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*. 2024, vol. 36.

106. CHEN, Zixiang; DENG, Yihe; YUAN, Huizhuo; JI, Kaixuan; GU, Quanquan. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*. 2024.

107. RADFORD, Alec; KIM, Jong Wook; HALLACY, Chris; RAMESH, Aditya; GOH, Gabriel; AGARWAL, Sandhini; SASTRY, Girish; ASKELL, Amanda; MISHKIN, Pamela; CLARK, Jack, et al. Learning transferable visual models from natural language supervision. In: *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

108. RAMESH, Aditya; PAVLOV, Mikhail; GOH, Gabriel; GRAY, Scott; VOSS, Chelsea; RADFORD, Alec; CHEN, Mark; SUTSKEVER, Ilya. Zero-shot text-to-image generation. In: *International conference on machine learning*. Pmlr, 2021, pp. 8821–8831.

109. LIU, Yixin; ZHANG, Kai; LI, Yuan; YAN, Zhiling; GAO, Chujie; CHEN, Ruoxi; YUAN, Zhengqing; HUANG, Yue; SUN, Hanchi; GAO, Jianfeng, et al. Sora: A Review on Background, Technology, Limitations, and Opportunities of Large Vision Models. *arXiv preprint arXiv:2402.17177*. 2024.

110. KOO, Ryan; LEE, Minhwa; RAHEJA, Vipul; PARK, Jong Inn; KIM, Zae Myung; KANG, Dongyeop. Benchmarking cognitive biases in large language models as evaluators. *arXiv preprint arXiv:2309.17012*. 2023.

111. DOSOVITSKIY, Alexey; BEYER, Lucas; KOLESNIKOV, Alexander; WEISSENBORN, Dirk; ZHAI, Xiaohua; UNTERTHINER, Thomas; DEHGHANI, Mostafa; MINDERER, Matthias; HEIGOLD, Georg; GELLY, Sylvain, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. 2020.

112. LI, Junnan; LI, Dongxu; XIONG, Caiming; HOI, Steven. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In: *International conference on machine learning*. PMLR, 2022, pp. 12888–12900.

113. MANN, Ben; RYDER, N; SUBBIAH, M; KAPLAN, J; DHARIWAL, P; NEELAKANTAN, A; SHYAM, P; SASTRY, G; ASKELL, A; AGARWAL, S, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*. 2020.

114. LAMPLE, Guillaume; CONNEAU, Alexis. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*. 2019.

115. YANG, Zhilin; DAI, Zihang; YANG, Yiming; CARBONELL, Jaime; SALAKHUTDINOV, Russ R; LE, Quoc V. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*. 2019, vol. 32.

116. BELKINA, Anna C; CICCOLELLA, Christopher O; ANNO, Rina; HALPERT, Richard; SPIDLEN, Josef; SNYDER-CAPPIONE, Jennifer E. Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. *Nature communications*. 2019, vol. 10, no. 1, p. 5415.

Enclosed Media Contents

- `README.md` ..... guide to navigate through the enclosed media content
- `data` ..... a folder with some precalculated data
  - `image_captioning.csv` ..... generated captions for all input images
  - `category_t_df.csv` ..... model output prediction for each triplet
  - `triplet_accuracy.pkl` ..... final results for original triplet dataset
  - `repeated_triplet_accuracy.pkl` ..... final results for repeated triplet dataset
- `code` ..... a folder with all necessary code
  - `dataset_processing_and_captioning.ipynb` ..... captions creation
  - `captions_processing.ipynb` ..... embeddings creation and scoring
  - `results_analysis.ipynb` ..... model performance evaluation
- `images` ..... a folder with some images from thesis