Vrije Universiteit Amsterdam

Master Thesis

A Benchmark to Evaluate the Lateral Thinking Skills of Visual Question Answering Models through Rebus Puzzles

Author: Koen Kraaijveld (2673132)

1st supervisor: Filip Ilievski
2nd reader: Kaixin Ma

A thesis submitted in fulfillment of the requirements for the VU Master of Science degree in Artificial Intelligence

June 17, 2024

Abstract. Visual question answering (VQA) has seen a great deal of progress in recent years, particularly with the development of synthetic, abstract visual reasoning benchmarks designed to evaluate the vertical thinking skills of vision-language models. Effective problem-solving, however, also combines lateral thinking, which remains relatively understudied and has only been addressed by a few text-only benchmarks in NLP. Toward bridging this gap, we propose a synthetic, multiple-choice benchmark that challenges the lateral thinking skills of vision-language models in VQA. The benchmark comprises 600 unique rebus puzzles partitioned into puzzles that include icons and those that do not. Through an experimental analysis of nine SOTA open-source MLLMs, we show that most models achieve above random performance with an average accuracy of 42.13% and 49.8% for non-icon and icon puzzles, respectively. All code is publicly available and can be found at: `Koen-Kraaijveld/rebus-puzzles`.

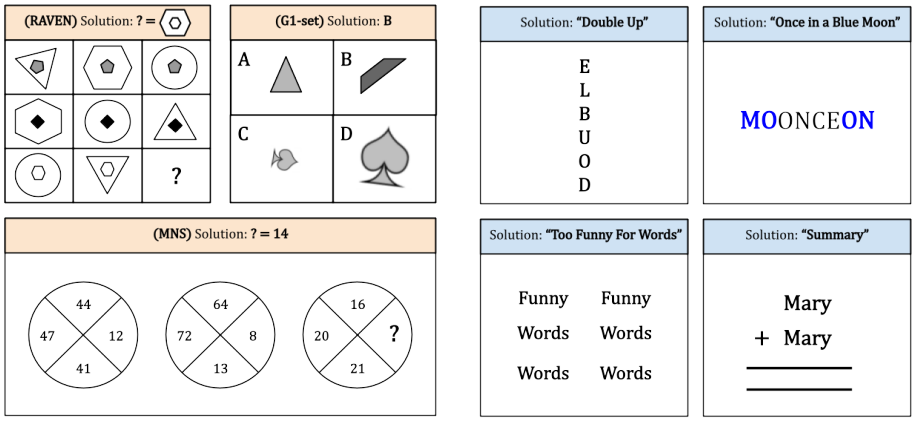
Keywords: Lateral Thinking · Visual Question Answering · Benchmark · Computer Vision · Natural Language Processing · Rebus Puzzles · Visual Puzzles

1 Introduction

To be an effective problem solver is to combine two modes of thinking [11]. On one hand, *vertical* thinking is an analytical search process that rewards logic, rules and rationality. It optimizes on correctness by narrowing down on quality solutions and rejecting suboptimal ones [16]. On the other hand, *lateral* thinking [12] is explorative, multidirectional and creative [16, 11]. It expands the solution space by considering unconventional ones.

Consider the puzzles in Figure 1. The left-hand side (1a) shows puzzles where a test-taker must either fill in an answer for the question mark (top left, bottom) or find the odd-one-out symbol (top right). Vertical thinking is more useful in this instance, as the puzzles demand a sequential process that systematically filters possible solutions based on patterns from the given examples. On the right (1b), four rebus puzzles are shown that each convey an idiom or word. Here, a test-taker needs to use their understanding of language, semantics and culture to arrive at the answer through a multi-step reasoning process. Visual, spatial, verbal and numerical cues (that ordinarily would be ignored in other puzzle types) must be interpreted in unconventional ways, a process that lends itself to lateral thinking.

Synthetic benchmarks in visual question answering (VQA) [1] have been useful in exploring the vertical thinking skills of vision-language models. As a task that sits at the crossroads of computer vision (CV) and natural language processing (NLP), these benchmarks cover visual reasoning [21, 24, 40], abstract visual reasoning [8, 28, 29, 20], and commonsense reasoning [7]. Despite this progress, there remains little work in gauging lateral thinking skills. Moreover, the work that has been done has largely been conducted in NLP.[19, 17]. An approach to extend this to CV still requires development.



(a) Three examples of visual puzzles that require vertical thinking to solve. The three puzzle types are Raven's Progressive Matrices from RAVEN [39] (top left), odd-one-out puzzles from G1-set [30] (top right), and visual math puzzles from Machine Number Sense (MNS) [42] (bottom).

(b) Four examples of rebus puzzles based on generic words or idioms. These were adapted from `https://www.rebuses.co/`.

Fig. 1: Examples of puzzles that require vertical thinking (left) or lateral thinking (right) to solve.

To this end, we aim to study *how well VQA models perform on visual puzzles that require lateral thinking to solve*. We propose a multiple-choice synthetic VQA benchmark of 600 puzzles. Rebus puzzles are chosen as the puzzle type because they are a reasonably well-studied class of puzzle that has been known to lend themselves to lateral thinking [35, 36, 27]. By analyzing puzzles from online rebus databases and the literature, we design a taxonomy of rules. Each of these rules uniquely manipulates the attributes or spatial relationships of text and icons (i.e., Unicode characters representing emojis) in a puzzle. Input data consists of idioms, common phrases and compound words. These are either scraped from online sources or downloaded from existing datasets. For puzzle generation, we implement a two-phase puzzle generation pipeline: first, by parsing an input word/phrase into a graph representation that can be reasoned about, and second by passing this into an image generation module. To generate distractors for each question, we compute a weighted average over the orthographic and semantic similarity between a puzzle's correct answer and the elements visible in that puzzle. As a last step, we partition the benchmark into puzzles that contain only text and ones that contain at least one icon. Using the final benchmark, we perform an experimental analysis of nine open source pretrained VQA models in a zero shot setting. The results show that most models achieve well above

random performance, maintaining an average accuracy of 42.13% and 49.8% for non-icon and icon puzzles, respectively

In summary, our contributions are as follows: 1) a taxonomy of rules to classify and reason about rebus puzzles, along with a pipeline to generate them, 2) a novel multiple-choice benchmark containing 600 puzzles partitioned into icon and non-icon instances, and 3) an experimental analysis involving the benchmark of several SOTA open-source VQA models.

We divide the thesis into the following sections. Section 2 covers related work, detailing the main works of visual puzzles in VQA and lateral thinking puzzles in NLP. Section 3 covers the methodology by describing the rule taxonomies, puzzle generation pipeline, distractor generation, data filtering and by presenting the final benchmark. Section 5 presents the results across three experiments, along with a discussion of them in Section 6. Lastly, Section 7 ends the thesis with a summary of the results.

2 Related Work

2.1 VQA with Visual Brainteasers

Many related works that study visual brainteasers do so through abstract visual reasoning (AVR) problems (see Figure 1a). The most studied visual brainteaser in Deep Learning is Raven's Progressive Matrices (RPM) [33], with significant contributions stemming from the benchmarks PGM [5] and RAVEN [39]. More recently, the Abstraction and Reasoning Corpus (ARC) [8] has emerged as a popular visual brainteaser benchmark that tries to measure how efficiently the test-taker learns new skills from limited demonstrations. A comprehensive overview of more AVR problems is given in [29].

Crucially, what all these brainteasers have in common is that they rely on vertical thinking. They are not designed to be solved through out-of-the-box approaches and as such are not able to measure a test-taker's lateral thinking skills.

2.2 Textual Lateral Thinking Puzzles

Predominantly, the work that has been done in designing lateral thinking benchmarks has been focused on NLP. Jiang et. al [19] introduce BrainTeaser, a multiple-choice lateral thinking benchmark that includes 1,100 puzzles web crawled from online sources. In particular, the authors try to measure the success with which instruction-based and commonsense LLMs are able to bypass commonsense associations to arrive at novel solutions to lateral thinking puzzles. Similarly, Huang et. al [17] present a benchmark consisting of 300 lateral thinking brainteasers. The authors frame the benchmark as an interactive game between two LLMs in which the LLM under evaluation must solve the puzzle presented by the host LLM (typically a more advanced model such as GPT-4).

Although these works make contributions to being some of the earliest work on lateral thinking benchmarks in NLP, they do not broaden the scope to include visual brainteasers in CV.

3 Methodology

In this section, we first give an overview of the main methods used to generate the benchmark, after which we explain the data selection and collection process. Then, we explain the taxonomies of rules used in the puzzle and present the pipeline used to generate these puzzles (including distractor generation). Lastly, we detail the process used for data filtering and present key statistics of the final benchmark.

3.1 Overview

The benchmark we create is based on rebus puzzles due to it's emphasis on lateral problem solving [35, 36, 27]. Moreover, the structure of rebus puzzles is simple enough to allow for a generative approach. This offers several benefits through:

1. **Access to metadata.** We can access metadata about each puzzles, providing additional possibilities during experiments.
2. **Fairness.** Current MLLMs are trained using huge snapshots of web data. By creating new puzzles, we aim to alleviate the memorization problem of MLLMs using rebus puzzles they might have seen during training.
3. **Copyright concerns.** Ensures all puzzles are free from copyright issues.

We split the benchmark into two partitions: puzzles that only contain text and puzzles that contain at least one icon. This isolates the type of visual recognition that models use for the puzzles. Non-icon puzzles exclusively require optical character recognition (OCR), while icon puzzles contain data that resembles what is typically seen during training for VQA models.

Figure 2 shows an overview of the modules and pipelines in our approach. The following sections describes these components in detail.

3.2 Data Selection and Collection

As rebus puzzles are typically built around common phrases, idiomatic expressions or compound words [35, 36], we select data covering these sources. For idioms, we scrape them from publicly available sources[1], yielding a total of 9745 idioms. For compound words, we use the Large Database of English Compounds (LaDEC) [14]. This dataset is clean, feature engineered, and consists of 8957 compounds. Lastly, we add custom compounds and phrases manually and by prompting ChatGPT. All combined, there are 18836 candidate answers to generate puzzles from.

Additionally, we also collect sources such as homophones and icons[2]. These are scraped from online sources, or manually added by recognizing common homophones and idioms found in rebus puzzle databases.

[1] Wiktionary: `en.wiktionary.org/w/index.php?title=Category:English_idioms`; The Idioms: `www.theidioms.com`

[2] Due to the size of these datasets, we refer to the code for 1) `icons` and 2) `homophones`. The icons are scraped from publicly available data on `unicode.org`.

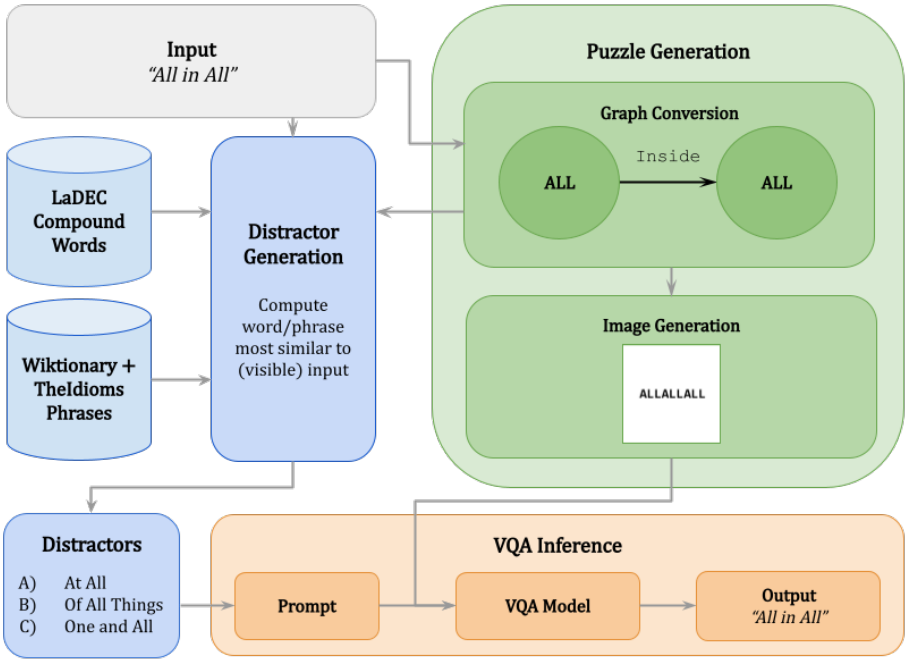


Fig. 2: Module overview and pipeline of our approach. The input is first converted into an attributed, directed graph representation through keyword extraction. The nodes are the elements that will be rendered to the final image (i.e., the elements that are visible to the test-taker) and the edges are the rules that define relationships between the rendered elements. This graph is then converted into the puzzle image. After this, distractor generation is performed by computing the top 3 most similar options to the original input out of a combined pool of words and phrases. The input and its distractors are formatted into a multiple choice question prompt and passed, together with the puzzle image, to the VQA model for inference.

3.3 Data Generation

Approach Overview Rebus puzzles consist of elements (i.e., either text or icons) whose appearance and position are manipulated and encoded through hidden rules. The test-taker is expected to decode these rules using their knowledge of language and semantics to solve the puzzle. What sets rebus puzzles apart is that these rules are designed to use visual, spatial, verbal and numerical cues in such a way that they challenge conventional thinking. The use of lateral thinking here stems from being able to generate unlikely solutions from cues that would ordinarily be ignored in other puzzle types.

A key observation is that the rules present in the puzzle will always be determined by specific keywords in the answer that is encoded by the puzzle. Figure

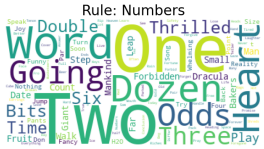
(a) Rebus puzzle that manipulates size (solution: *Big Deal*)

(b) Rebus puzzle that manipulates color (solution: *Tickled Pink*)

(c) Rebus puzzle that manipulates numbers (solution: *Two Left Feet*)

(d) Word cloud for rebus puzzle answers that encode a rule for size.

(e) Word cloud for rebus puzzle answers that encode a rule for color.



(f) Word cloud for rebus puzzle answers that encode a rule for numbers.

Fig. 3: Examples showing how the words in a rebus puzzle answer relate to the puzzle itself. For each column, the top half image shows an example of a rebus puzzle that encodes the rule from the bottom half. The bottom half image is a word cloud consisting of all rebus puzzle answers that encode the specified rule. These puzzles and their answers were all scraped from `rebuses.co`, each of which has a tag showing which rule is encoded.

3 gives three examples from an online rebus database of how the keywords in a rebus puzzle's answer influence the elements in that puzzle. This observation forms the basis of our approach to puzzle generation.

The generation process itself consists of two phases. First, we parse a rebus puzzle answer into a graph that encodes the puzzle's elements and rules. Then, we use Matplotlib's text-in-image features to generate the final image from this graph.

Defining the Rules With respect to the possible rules that can be present in a puzzle, we select 18. These are chosen by reviewing rules from online guides and databases of rebus puzzles,[3] and from a rebus categorization scheme outlined by Salvi et. al [35]. We then filter the rules down to a suitable subset. This process is done on the basis that each rule uniquely manipulates an element through visual, spatial, verbal and numerical properties, yet that each rule is still simple enough to be generated automatically and can be mixed with other rules in the same puzzle.

[3] Sources: (1) `rebuses.co`, (2) `eslvault.com`, and (3) `amagiclassroom.com`

We then categorize the puzzles according to how they manipulate elements in a puzzle. These are as follows:

- Individual. Defines the individual characteristics of an element in a rebus. Examples include changing color, size, or character order. Due to the nature of the parsing algorithm, at most only one *Individual* rule can be applied to a single element. Multiple of these elements can still be present in the same puzzle.
- Relational. Defines the positioning between two elements in a rebus. Examples include placing an element beside/inside/above/outside another element.
- Modifier. A set of rules designed to be mutually-inclusive with other *Individual* rules. Examples include repeating an element multiple times, or substituting it with a phonetically similar element.

Figure 4 shows the taxonomies used to organize all the rules across all three categories, as well as examples on what each rule looks like visually. For a complete list of all keywords that trigger these rules, see Appendix A.2.

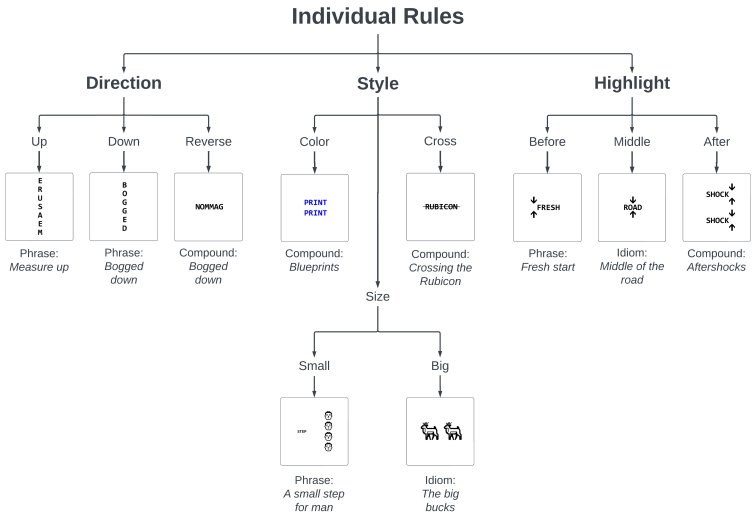
Graph Parsing Algorithm We first explain the details of the graph itself that is used to represent the rebus puzzles. This graph is a directed, attributed path graph. All nodes in the graph are elements that will be rendered to the graph. The attributes of each node determine what that element will look like (i.e., which *Individual* or *Modifier* rules it contains). Any edge connecting two nodes determines a *Relational* rule between them, with the edge's attribute specifying which rule it is. Figure 5 shows two examples of rebus puzzles that are represented as graphs along with the final image of the puzzle.

The parsing algorithm used to produce a graph differs for compounds and phrases. For both approaches, we first ignore a specific set of words by removing them from the input (see Appendix A.1 for the complete list). These are stopwords that do not overlap with any of the triggered keywords (e.g., "to" is a stopword, but triggers the *Repetition: Two* rule phonetically).

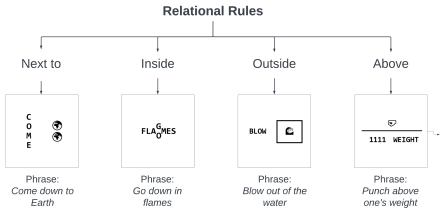
For compounds, their structure is simplest and only requires a graph with a single node. Figure 6 shows a step-by-step example of the algorithm. In cases where both constituent words of a compound trigger a rule, we generate both graph interpretations of the input.

For phrases, we use the following process:

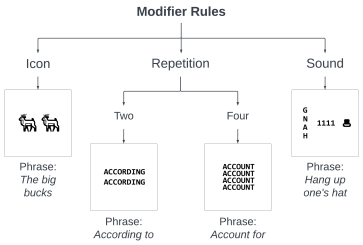
1. First, we identify keywords in the phrase belonging to a *Relational* rule.
2. If a keyword match is found, we split the phrase into two subphrases at the position of the matched word. If no match is found, we retain the original phrase without splitting.
3. We then apply the same algorithm used for the compound word parser to every pair of words within each (sub)phrase, starting from the beginning. The compound word parser algorithm generates graphs with a single node for each word pair.



(a) Individual rules

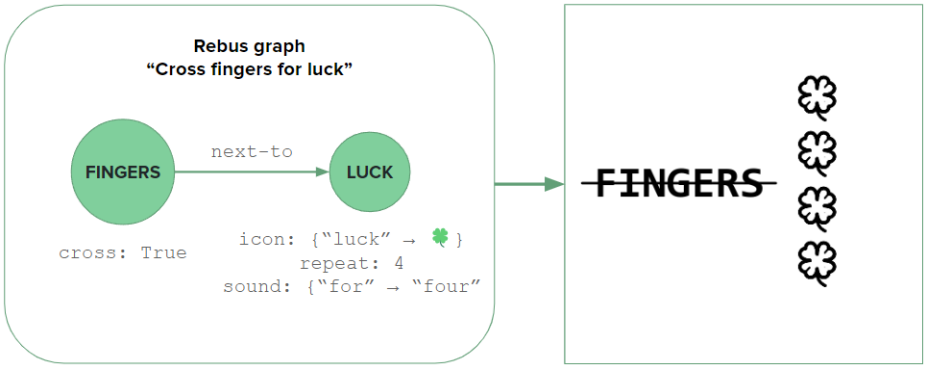


(b) Relational rules

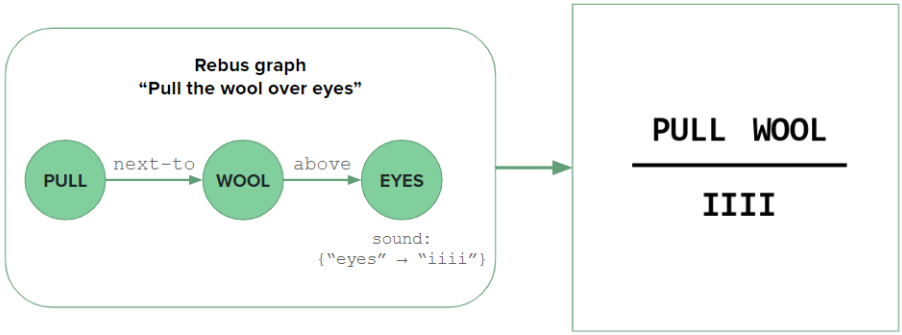


(c) Modifier rules

Fig. 4: Three taxonomies that classify and organize the (a) *Individual*, (b) *Relational*, and (c) *Modifier* rules used to manipulate the appearance and position of elements in a rebus puzzle. For each rule, we present an example puzzle along with its answer, both of which are taken directly from the benchmark.



(a) *Cross fingers for luck*



(b) *Pull wool over eyes*

Fig. 5: Two examples of how rebus puzzles are represented as graphs. For each figure (a) and (b), the answer is parsed into a graph representation (left) and then converted into the resulting image (right). Each node represents an element that contains text that will be rendered to the final image. Node attributes are the *Individual* properties of the element's text, while edges are the *Relational* properties between elements.

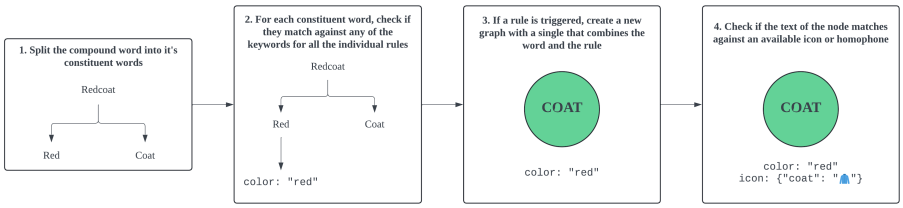


Fig. 6: Figure showing the simplest case of how a compound word is parsed into a graph step-by-step.

4. We connect these nodes into a path graph where all edges represent the *Next To* relational rule. The order of the nodes is maintained according to their position of the matched word pair in the original (sub)phrase. If the compound word parser generates two graphs for a word pair, we compute all possible combinations of path graphs from that word pair.
5. If the phrase was split at the start, then there are two subgraphs for each subphrase. These are reconnected using the relational rule identified in step 1.

Image Generation Image generation is performed using a graph as input. This will generate a 400×400 pixel image through Matplotlib's text-in-image capabilities. The font used for text is Consolas, as it is monospaced, which is convenient for spatial calculations. To render icons, we use the Segoe UI Emoji font.

The general approach involves selecting a template from four options based on the *Relational* rules present in the graph, as these define the positional properties of each element. Each template consists of x, y coordinates and a multiplier to control the font size. Three of the four templates are points (up to three) spaced roughly equally along the x-axis in the middle of the image. The remaining template is used for graphs involving the *Above* relational rule. The points in these templates are filled in with text/icons represented by the nodes of the graph. We sequentially change the appearance of these elements according to the attributes of that element's respective node.

Additionally, there are constraints that restrict the number of rendered elements in the image. This is intended to improve the readability by reducing clutter and the risk of overlapping elements. These are as follows:

- Graphs with nodes that are all connected with the *Next To* relational rule will not be rendered if it has more than three nodes.
- Graphs that contain an *Above* relational rule will not be rendered if either the top and bottom portions exceed two nodes.
- Graphs that contain an *Inside* relational rule will not be rendered if either the inside portion has more than one node, or the outside portion has more than two nodes.
- Graphs that contain an *Outside* relational rule will not be rendered if either the inside or outside portions has more than one node.

Distractor Generation Distractor generation consists of semi-automatically selecting the three most similar compounds or idioms/phrases to the input. We do not opt for a data augmentation approach. This is because, in the case of idioms, the test-taker often does not even have to look at the puzzle to discern the correct idiom when contrasted with three other non-idiomatic options.

For each input, we start by identifying the elements that are visible in the puzzle generated by that input. This is performed by converting the input to its graph representation and extracting the text/icon associated with each node. In the case of icons, we convert this back to its original text from its Unicode counterpart to enable similarity computations. If the input is a compound, we

split it into its constituent words. This is because we compute word overlap similarity and one of the constituent words is what will be extracted as a visible word from the graph.

Similarity is measured through word overlap on the visible words (Jaccard similarity) and semantic analysis on the entire input (BERT embeddings). To balance these two metrics, we use a manually tuned weighted average between them. The weight for word overlap is set to 0.8. We prioritize computing the similarity on visible words through word overlap to increase the difficulty, as these are the words the test-taker has to work with in order to solve the puzzle. However, this approach on its own struggles to select relevant distractors, since most visible words only occur once or too many times. Therefore, we aim to balance this issue with semantic similarity through sentence BERT embeddings[4]. This also has the added bonus of selecting phrases containing synonyms of the words in the original input, thereby increasing the challenge.

Data Filtering As a final step, we filter through all generated puzzles and distractors. First, we remove unreadable puzzles that were broken from the generation process (e.g., elements that are too large, overlapping text/icons, text/icon overflow, etc.). Then, we remove puzzles in which the only rule used is replacing text with icons. These puzzles are more similar to emoji puzzles than a rebus, and we do not consider them to require a sufficient degree of lateral thinking. We prioritize puzzles that feature denser graphs with respect to the number of edges and rules per node. We expect these puzzles to challenge a test-taker more, as it requires them to decode more about an element to understand its use in a puzzle.

For distractors, we perform minimal data filtering. We only replace a distractor if it is too similar to another by a few insignificant stopwords (i.e., stopwords that don't trigger any rules). In such a case, we replace it with the next most similar distractor (if it is not too similar for the same reason). If two words are the singular and plural forms of each other, we do not replace either, as plurality is used to trigger the *Repetition* rules.

Final Benchmark Table 1 shows key statistics on the benchmark. While there is a roughly even distribution of puzzles with and without icons, the complexity between them varies. Icon puzzles tend to feature more elements. This can be attributed to the difference in the answer length, as longer answers feature more chances that a word can be replaced with an icon.

Table 2 shows the frequency distribution of puzzles containing a specific rule. Here, we see that while puzzles without icons tend to contain more *Individual* rules, they also contain less *Relational* rules. This highlights that non-icon puzzles focus more on the appearance of elements, rather than the spatial relationships between them.

[4] We use the sentence similarity model `all-MiniLM-L6-v2`. This is the most downloaded sentence similarity model available on Huggingface.

Table 1: Table showing key statistics of the benchmark.

Statistic	Without icons	With icons	All
Number of puzzles	300	387	687
Mean answer length (# words)	2.77	3.89	3.40
Number of single node graphs	165	37	202
Number of double node graphs	101	254	355
Number of triple node graphs	34	96	130
Mean number of nodes per graph	1.57	2.17	1.91
Mean number of edges per graph	0.57	1.17	0.91

4 Experimental Setup

4.1 Model Selection

The models we will compare are vision-language models trained for VQA tasks. These are all open-source pretrained models available from online sources. The models are as follows:

- CLIP [32], a seminal multi-modal vision-language model. We use this model as our baseline due to its prevalence across CV research and because it is used as a foundation for many of the models used in these experiments. As CLIP is not a VQA model, we switch the task for this model to image classification where it must match the image of a puzzle to the correct answer from the four available choices.
- BLIP-2 [23], a vision-language model that leverages the OPT [41] or Flan-T5 [9] LLMs and which achieves SOTA performance on zero-shot VQA tasks. We use the OPT-2.7b, OPT-6.7b and Flan-T5-XXL-11b variants.
- InstructBLIP [10], an instruction tuned alternative based on a pretrained BLIP-2 model that uses the Vicuna-7b [43] LLM.
- Fuyu-8b [6], a multimodal text and image transformer that achieves competitive performance on VQA tasks.
- QwenVL [3], a 7 billion visual multimodal version of the Qwen LLM [2]. We use the chat variant.
- CogVLM [37], a 17 billion parameter visual language model that achieves SOTA performance on several VQA benchmarks.
- Llava [26], a large multimodal model that achieves SOTA performance on several vision benchmarks despite it's lack of billion scale data. We use the 13b (v1.5) and 34b (v1.6) variants, in which the 34b model is quantized to 4-bit.

Table 2: Table showing the sample size of puzzles that contain a specified rule. The *Direction* rules are omitted because these either do not function with icons, or become functionally identical to other rules when combined with icons.

Rule Type	**Rule**	**Rule sample size**	
		Without icons	With icons
Individual	Direction: Up	27	N/A
	Direction: Down	20	N/A
	Direction: Reverse	20	N/A
	Highlight: Before	13	10
	Highlight: Middle	12	10
	Highlight: After	18	10
	Size: Big	12	11
	Size: Small	13	10
	Color	16	13
	Cross	19	12
Relational	Next to	111	245
	Inside	23	106
	Above	26	83
	Outside	10	19
Modifier	Sound	170	158
	Repetition: Two	109	87
	Repetition: Four	41	31

Lastly, we also include the instruction tuned Mistral-7b (v2) [18] model to use in text-only, question answering (QA) auxiliary experiments. All experiments use these models in zero-shot settings. During inference, we do not alter the hyperparameters that were provided in the documentation of their Huggingface pages outside of the generated output token length where necessary.

Resources All models were used on the DAS-6 cluster [4]. Almost all models were run on a Nvidia RTX A6000, except BLIP-2 OPT-2.6b which was used on an RTX A10.

4.2 Evaluation Metrics

The evaluation metric we use is accuracy due its fairness and popularity across other multiple-choice benchmarks [19, 44, 13]. We define it as the percentage of puzzles correctly solved and is computed for icon and non-icon puzzles separately.

To extract answers from a model's output, we use regex to check for choice symbols (e.g., "A") if they are present and then perform exact string matching to the correct answer/symbol. In the larger, more flexible models we manually annotate the answers given in their outputs and extract them with regex. We also ignore outputs that give more than one answer.

5 Results

Our experiments focus on three questions: 1) How well can MLLMs solve rebus puzzles that require lateral thinking? 2) Do MLLMs perform better on prompts that supply more information about the puzzle? 3) Are there types of puzzles that MLLMs are better at solving?

Overall Performance Table 3 shows the performance of each model when solving the benchmark, divided into puzzles that include/exclude icons. The prompt we use includes a short description on the nature of the rebus puzzle (see Table 9 in Appendix B.1) as a middle ground between an uninformative and overly informative prompt.

For puzzles without icons, we can observe that larger models tend to perform the best on the benchmark, but still fail to outperform the baseline for the most part. For puzzles with icons, a similar trend is seen, yet most models are able achieve a higher accuracy than the baseline. Overall, all models (bar one) are better at solving puzzles that include icons. For both partitions, the best performing model, BLIP-2 Flan-T5-XXL, achieves the highest performance, although this is not the largest model. The two smallest models achieve the worse performance with near-random answers.

Additionally, most models struggle to achieve a uniform answer distribution, as the most common answer tends to be A or D. While larger models tend to show a more uniform answer distribution, some small models (Fuyu and Qwen-VL) are also included. This suggests that these inductive biases are not always tied to model size.

Performance on Different Prompts Table 9 shows the performance of each model for different prompts (see Table 9 in Appendix B.1) and for puzzles with/without icons. Each type of prompt supplies the model with varying levels of information

Table 3: Table showing the results for each model for puzzles with/without icons. The accuracy and most commonly picked answer is reported. Highest results in each column are in bold and underlined.

Model (Size)	Puzzles without icons		Puzzles with icons	
	Accuracy (%)	Most common answer (%)	Accuracy (%)	Most common answer (%)
CLIP (baseline)	49.33	C → 27.67	52.71	D → 26.1
BLIP-2 OPT (2.7b)	21.33	A → 78.66	24.03	A → 82.24
BLIP-2 OPT (6.7b)	24.33	**A → 97.59**	23.26	**A → 99.22**
InstructBLIP Vicuna (7b)	44.00	D → 65.33	51.94	D → 55.04
Qwen-VL (7b)	45.67	C → 28.83	56.07	D → 36.58
Fuyu (8b)	29.67	A → 29.92	35.40	C → 27.41
BLIP-2 Flan-T5-XXL (11b)	**58.33**	A → 40.75	73.13	A → 32.38
Llava (13b)	50.67	A → 37.70	54.01	D → 38.79
CogVLM (17b)	48.00	C → 37.25	58.40	D → 37.5
Llava (34b)	50.00	A → 33.33	**68.99**	A → 32.49

on a given puzzle (1 = least information, 4 = most information)[5]. Prompts 3 and 4 give the model a description of the graph, in addition to the image. Prompt 3 only describes the nodes of the graph and its attributes, while prompt 4 describes the nodes, edges and attributes of both.

Adding information on the nature the puzzle (i.e., from prompt 1 to 2) leads to a small improvement (+4.66% and +4.49% for non-icon and icon puzzles, respectively, averaged across all models excluding CLIP and Mistral). A two-sample, two-tailed Wilcoxon signed-rank test (significance level $\alpha = 0.05$) shows that these differences are not significant (p-values are 0.74 and 0.46 for non-icon and icon puzzles, respectively).

Furthermore, we can observe that the addition of the graph description helps improve the performance of almost all models. For non-icon and puzzles, this improvement is on average 9.38% and 7.6%, respectively, across all models (excluding CLIP and Mistral). This difference is not significant according to a Wilcoxon signed-rank test with p-values of 0.008 and 0.04 for non-icon and icon puzzles, respectively.

However, there is no substantial difference by including information on the *Relational* rules through the edges of the graph, with an average improvement

[5] For simplicity, we henceforth refer to these prompts with this numbering scheme. For a more descriptive naming scheme, see Appendix B.1.

0.83% and 2.65% for non-icon and icon puzzles, respectively, across all models except CLIP and Mistral. This suggests that the models rely more on the elements they see in the puzzle, rather than the spatial relationships between them.

Lastly, we see that Mistral achieves a strong performance despite relying solely on the graph description. This also suggests that the VQA models are prioritizing the graph description over the image of the puzzle.

Table 4: Table showing results for four prompts that supply the model with varying levels of information (see Appendix B.1). Results are partitioned into puzzles that include/exclude icons, and only the accuracy is reported. The results from Table 3 are also shown with prompt 2. Results for Mistral are also reported, but only including the graph description (no image is passed). Highest results in each column are in bold and underlined

Model Type	**Model (Size)**	**Accuracy (%) Puzzles without icons**				**Accuracy (%) Puzzles with icons**			
		Prompt				**Prompt**			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**
VQA	BLIP-2 OPT (2.7b)	18.00	21.33	25.33	25.00	23.00	24.03	21.45	24.55
	BLIP-2 OPT (6.7b)	24.67	24.33	24.67	24.00	21.45	23.26	23.26	24.29
	InstructBLIP Vicuna (7b)	45.33	44.00	48.00	47.00	54.26	51.94	56.33	49.35
	Qwen-VL (7b)	25.67	45.67	57.00	56.33	40.83	56.07	62.79	58.91
	Fuyu (8b)	29.00	29.67	38.33	36.67	34.11	35.40	42.89	41.60
	BLIP-2 Flan-T5-XXL (11b)	54.67	**58.33**	**80.67**	**80.00**	67.18	**73.13**	**90.18**	**90.96**
	Llava (13b)	55.33	50.67	64.00	63.00	60.98	54.01	69.25	70.28
	CogVLM (17b)	51.33	48.00	59.00	59.67	57.36	58.40	65.63	62.53
	Llava (34b)	**55.67**	50.00	-	-	**70.54**	68.99	-	-
QA	Mistral (7b)	N/A	N/A	65.67	67.67	N/A	N/A	71.58	74.94

Performance for Different Rules Tables 5 and 6 show results for how often a puzzle containing a specific rule is solved for icon and non-icon puzzles, respectively. These results are averaged across models[6], as well as showing performance across all four prompts (see 9 in Appendix B.1).

[6] In computing this average, we exclude the baseline and models showing highly non-uniform answer distributions where the most common answer is picked more than

For performance on non-icon puzzles, we can observe that models tend to be more accurate when *Relational* rules are involved, rather than the *Individual* rules (mean improvement of 11.74% across all prompts). This performance is also more consistent, with a std. dev. of 6.8% for the *Relational* rules and 9.16% for the *Individual* rules (averaged across all prompts). The inclusion of *Modifier* rules when combined with *Individual* rules leads to a small improvement (mean improvement of 5.38%).

For the *Individual* rules, the models perform best with the *Color* rule, while struggling most with the *Direction: Reverse* (near-random performance). Notably, the models also fail to perform consistently on the relational *Inside* rule, while performing well on the others in the same category.

For performance on icon puzzles, models also tend to perform better on *Relational* vs. *Individual* rules, although this improvement is smaller than for non-icon puzzles (mean improvement of 7.66% across all prompts). The consistency across both categories is approximately equal (both with a std. dev. of ≈ 7.5, averaged across all prompts). The average improvement in performance with the *Modifier* rules is 10.42%, almost double that of non-icon puzzles.

The models struggle excel most at solving the *Middle* rule, while struggling most with the *Before* rule, both of which are from the *Highlight* rule. Interestingly, when icons are involved they do not struggle with the *Inside* relational rule, but rather the *Outside* one.

Notably, while there is a minimal improvement (1.44%) for the *Relational* rules between prompts 3 and 4 (i.e., when information on which *Relational* rule is passed), there is a substantial degradation in performance (-7.14%) for the *Above* rule. The opposite happens for the *Outside* rule during icon puzzles (+6.01%).

6 Discussion

The benchmark presented in this thesis introduces 687 multiple-choice rebus puzzles that require lateral thinking to solve. We demonstrate a systematic approach to generate these puzzles through a two-phase process: first, by parsing a word or phrase into a graph representation that can be reasoned about, and second, by using this representation to generate an image of the puzzle. Moreover, we present an experimental analysis of nine open-source MLLMs using the benchmark.

Eight out of nine models perform better on puzzles with icons, possibly because these MLLMs are not trained for OCR. Even the best performing models in our experiments (BLIP-2 Flan-T5-XXL and both Llava models) are trained on real-world image datasets [25, 22]. Icons may make the puzzles easier for these models as they are more familiar with such data.

All models perform better with the addition of a description of the graph. This improvement is greater for puzzles without icons ($\approx 2\%$), further suggesting that these models struggle with OCR as they perform better when given the

50% of the time (InstructBLIP, BLIP 2 OPT 2.7b and 6.7b). We also exclude Mistral as it does not have access to the puzzle image.

Table 5: Table showing percentage of puzzles (without icons) solved correctly for each rule. These results are averaged across all the models for each prompt. Note: This only considers if a model is able to solve a given puzzle when a rule is present in that puzzle. It does not consider whether the model is able to solve that rule specifically, irrespective of whether or not it can solve the puzzle as a whole.

Rule Type	**Rule**	**Puzzles without icons Puzzles solved correctly (%)**			
		Prompt 1	**Prompt 2**	**Prompt 3**	**Prompt 4**
Individual	Direction: Up	39.68	36.51	52.91	56.08
	Direction: Down	52.86	52.14	65.71	65.71
	Direction: Reverse	27.14	23.57	45.00	41.43
	Highlight: Before	45.06	43.96	47.25	47.25
	Highlight: Middle	30.95	51.19	47.62	46.43
	Highlight: After	46.82	44.44	41.27	43.65
	Size: Big	51.19	46.43	57.14	55.95
	Size: Small	30.77	36.26	39.56	41.76
	Color	59.82	62.50	61.61	63.39
	Cross	38.35	42.11	53.38	52.63
Relational	Next to	60.23	62.16	63.19	61.52
	Inside	44.72	40.37	55.28	54.04
	Above	64.28	60.99	61.54	54.4
	Outside	61.43	74.29	60.00	64.29
Modifier	Sound	52.44	55.04	57.31	56.05
	Repetition: Two	50.59	51.64	50.33	50.06
	Repetition: Four	45.65	54.01	53.66	54.01

text directly through the question. The minimal improvement ($\approx 0.83\%$ and $\approx 2.65\%$) when expanding a models access to include *Relational* rules implies they rely mostly on the elements in a puzzle alone, rather than the spatial relationships between them.

Table 6: Table showing percentage of puzzles (with icons) solved correctly for each rule. These results are averaged across all the models for each prompt. All *Direction* rules are omitted as the use of icons do not work with this rule or are functionally identical to other rules. Note: This only considers if a model is able to solve a given puzzle when a rule is present in that puzzle. It does not consider whether the model is able to solve that rule specifically, irrespective of whether or not it can solve the puzzle as a whole.

Rule Type	**Rule**	**Puzzles with icons Puzzles solved correctly (%)**			
		Prompt 1	**Prompt 2**	**Prompt 3**	**Prompt 4**
Individual	Highlight: Before	34.29	38.57	41.43	51.43
	Highlight: Middle	60.00	61.43	60.00	62.86
	Highlight: After	42.86	42.86	55.71	57.14
	Size: Big	40.26	46.75	45.45	58.57
	Size: Small	57.14	61.43	57.14	48.05
	Color	52.75	46.15	56.04	53.85
	Cross	38.10	48.81	45.24	42.86
Relational	Next to	58.25	61.11	62.10	60.17
	Inside	61.32	63.75	63.75	64.55
	Above	59.38	66.27	62.65	61.79
	Outside	42.86	45.11	43.61	49.62
Modifier	Sound	56.87	60.40	61.12	59.04
	Repetition: Two	59.60	65.03	60.59	59.60
	Repetition: Four	56.68	60.83	65.44	62.67

As a whole, the results can be compared to REBUS, a rebus puzzle benchmark by Gritsevskiy et al. [15], which features hand-crafted puzzles. These puzzles demand world knowledge (e.g., geography, culture, names, marine animals) and more vertical thinking techniques, such as string manipulation. In their experiments, six out of the nine models used in our experiments achieved less than 3% accuracy. As our benchmark requires only some cultural knowledge through idioms and avoids vertical thinking, this highlights how these factors can be used to increase the challenge of rebus puzzles.

Limitations and Future Work There is a notable variability in the properties of puzzles with and without icons (see Table 1), as they are both disjoint sets. This makes it challenging to draw meaningful conclusions about model performance across these puzzle types. To address this, a subset of puzzles should be reconfigured to bridge this gap. One approach is to convert icon puzzles back to their original text-based versions. This produces two nearly identical puzzles: one with icons and one with text.

A limitation of our experiments is the prevalence of a non-uniform answer distribution across six out of nine models for both puzzles with/without icons. This issue is consistent across other multiple-choice question-answering tasks [20, 34, 31, 38]. Proposed solutions include multiple choice symbol binding [34], where models must return answers with their corresponding symbols (e.g., "A"). Another includes circular choices [38], where answer options are rearranged cyclically and only once the model has correctly answered all arrangements is a question considered correct.

Furthermore, the experiments suggest that to solve a puzzle, a model often relies more heavily on the OCR word extraction and matching of each text/icon in a puzzle image to the text in the question's choice options. Models already ignore the spatial relationships between elements (see Table 4), and it is possible this extends to their appearance (i.e., the attributes of each node in a graph). This makes it challenging to determine if models are truly using creative problem-solving or simply matching extracted text to the choices in the question without considering the context.

7 Conclusion

In summary, we present a synthetic, multiple choice benchmark consisting of 687 unique rebus puzzles that are partitioned into non-icon and icon puzzles. We develop a taxonomy of rules across three categories, which is used in two-phase puzzle generation process. Our results on nine open-source MLLMs show that most models achieve above random performance. Models tend to have a worse accuracy on puzzles without icons, which may be attributed to a dependence on OCR that most models are not explicitly trained for. Furthermore, model accuracy can be significantly enhanced by providing descriptive metadata of a puzzle in a prompt. However, this improvement stems only from providing information on the appearance of elements in a puzzle, rather than spatial interactions between them.

References

[1] Aishwarya Agrawal et al. *VQA: Visual Question Answering*. 2016. arXiv: 1505.00468 [cs.CL].

[2] Jinze Bai et al. "Qwen Technical Report". In: *arXiv preprint arXiv:2309.16609* (2023).

[3] Jinze Bai et al. "Qwen-VL: A Versatile Vision-Language Model for Understanding, Localization, Text Reading, and Beyond". In: *arXiv preprint arXiv:2308.12966* (2023).

[4] Henri Bal et al. "A Medium-Scale Distributed System for Computer Science Research: Infrastructure for the Long Term". In: *IEEE Computer* 49.5 (May 2016), pp. 54–63.

[5] David G. T. Barrett et al. *Measuring abstract reasoning in neural networks*. 2018. arXiv: 1807.04225 [cs.LG].

[6] Rohan Bavishi et al. *Fuyu-8B: A Multimodal Architecture for AI Agents*. Adept. Oct. 17, 2023. URL: https://www.adept.ai/blog/fuyu-8b/ (visited on 04/15/2024).

[7] Nitzan Bitton-Guetta et al. "Breaking Common Sense: WHOOPS! A Vision-and-Language Benchmark of Synthetic and Compositional Images". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2023, pp. 2616–2627.

[8] François Chollet. *On the Measure of Intelligence*. 2019. arXiv: 1911.01547 [cs.AI].

[9] Hyung Won Chung et al. *Scaling Instruction-Finetuned Language Models*. _eprint: 2210.11416. 2022.

[10] Wenliang Dai et al. "InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 49250–49267. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/9a6a435e75419a836fe47ab6793623e6-Paper-Conference.pdf.

[11] Edward De Bono. *Lateral thinking: a textbook of creativity*. London: Penguin Life, 2016. 260 pp. ISBN: 978-0-241-25754-8.

[12] Edward De Bono. *The Use of Lateral Thinking*. Intl Center for Creative Thinking, Jan. 1, 1971. 144 pp. ISBN: 978-0-14-013788-0.

[13] Ana Cláudia Akemi Matsuki de Faria et al. *Visual Question Answering: A Survey on Techniques and Common Trends in Recent Literature*. 2023. arXiv: 2305.11033 [cs.CV].

[14] Christina L. Gagné, Thomas L. Spalding, and Daniel Schmidtke. "LADEC: The Large Database of English Compounds". In: *Behavior Research Methods* 51.5 (Oct. 1, 2019), pp. 2152–2179. ISSN: 1554-3528. DOI: 10.3758/s13428-019-01282-6. URL: https://doi.org/10.3758/s13428-019-01282-6.

[15] Andrew Gritsevskiy et al. *REBUS: A Robust Evaluation Benchmark of Understanding Symbols*. 2024. arXiv: 2401.05604 [cs.CL].

[16] James Hernandez and Prathibha Varkey. "Vertical versus lateral thinking". In: *Physician executive* 34 (May 2008), pp. 26–8.

[17] Shulin Huang et al. "LatEval: An Interactive LLMs Evaluation Benchmark with Incomplete Information from Lateral Thinking Puzzles". In: (Aug. 2023).

[18] Albert Q. Jiang et al. *Mistral 7B*. 2023. arXiv: 2310.06825 [cs.CL].

[19] Yifan Jiang et al. *BRAINTEASER: Lateral Thinking Puzzles for Large Language Models*. 2023. arXiv: 2310.05057 [cs.CL].

[20] Yifan Jiang et al. *MARVEL: Multidimensional Abstraction and Reasoning through Visual Evaluation and Learning*. 2024. arXiv: 2404.13591.

[21] Justin Johnson et al. "CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[22] Ranjay Krishna et al. "Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations". In: *International Journal of Computer Vision* 123.1 (May 1, 2017), pp. 32–73. ISSN: 1573-1405. DOI: 10.1007/s11263-016-0981-7. URL: https://doi.org/10.1007/s11263-016-0981-7.

[23] Junnan Li et al. "BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models". In: *Proceedings of the 40th International Conference on Machine Learning*. ICML'23. Place: , Honolulu, Hawaii, USA, JMLR.org, 2023.

[24] Zechen Li and Anders Søgaard. "QLEVR: A Diagnostic Dataset for Quantificational Language and Elementary Visual Reasoning". In: *Findings of the Association for Computational Linguistics: NAACL 2022*. Ed. by Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz. Seattle, United States: Association for Computational Linguistics, July 2022, pp. 980–996. DOI: 10.18653/v1/2022.findings-naacl.73. URL: https://aclanthology.org/2022.findings-naacl.73.

[25] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 740–755. ISBN: 978-3-319-10602-1.

[26] Haotian Liu et al. "Visual Instruction Tuning". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 34892–34916. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/6dcf277ea32ce3288914faf369fe6de0-Paper-Conference.pdf.

[27] James N. MacGregor and J. Barton Cunningham. "Rebus puzzles as insight problems". In: *Behavior Research Methods* 40.1 (Feb. 1, 2008), pp. 263–268. ISSN: 1554-3528. DOI: 10.3758/BRM.40.1.263. URL: https://doi.org/10.3758/BRM.40.1.263.

[28] Mateusz Malinowski and Mario Fritz. *A Multi-World Approach to Question Answering about Real-World Scenes based on Uncertain Input*. 2015. arXiv: 1410.0210 [cs.AI].

[29] Mikołaj Małkiński and Jacek Mańdziuk. "A review of emerging research directions in Abstract Visual Reasoning". In: *Information Fusion* 91 (2023), pp. 713–736. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2022.11.011. URL: https://www.sciencedirect.com/science/article/pii/S1566253522002214.

[30] Jacek Mańdziuk and Adam Żychowski. "DeepIQ: A Human-Inspired AI System for Solving IQ Test Problems". In: International Joint Conference on Neural Networks. July 2019, pp. 1–8. DOI: 10.1109/IJCNN.2019.8851878.

[31] Pouya Pezeshkpour and Estevam Hruschka. *Large Language Models Sensitivity to The Order of Options in Multiple-Choice Questions*. 2023. arXiv: 2308.11483 [cs.CL].

[32] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020.

[33] JOHN C. RAVEN. "STANDARDIZATION OF PROGRESSIVE MATRICES, 1938". In: *British Journal of Medical Psychology* 19.1 (1941), pp. 137–150. DOI: https://doi.org/10.1111/j.2044-8341.1941.tb00316.x.

[34] Joshua Robinson and David Wingate. "Leveraging Large Language Models for Multiple Choice Question Answering". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=yKbprarjc5B.

[35] Carola Salvi et al. "Validation of Italian rebus puzzles and compound remote associate problems". In: *Behavior Research Methods* 48.2 (June 1, 2016), pp. 664–685. ISSN: 1554-3528. DOI: 10.3758/s13428-015-0597-9. URL: https://doi.org/10.3758/s13428-015-0597-9.

[36] Emma Threadgold, John E. Marsh, and Linden J. Ball. "Normative Data for 84 UK English Rebus Puzzles." In: *Frontiers in psychology* 9 (2018). Place: Switzerland, p. 2513. ISSN: 1664-1078. DOI: 10.3389/fpsyg.2018.02513.

[37] Weihan Wang et al. *CogVLM: Visual Expert for Pretrained Language Models*. 2023. arXiv: 2311.03079 [cs.CV].

[38] Fangyun Wei, Xi Chen, and Linzi Luo. "Rethinking Generative Large Language Model Evaluation for Semantic Comprehension". In: *ArXiv* abs/2403.07872 (2024). URL: https://api.semanticscholar.org/CorpusID:268363952.

[39] Chi Zhang et al. *RAVEN: A Dataset for Relational and Analogical Visual rEasoNing*. 2019. arXiv: 1903.02741 [cs.CV].

[40] Peng Zhang et al. "Yin and Yang: Balancing and Answering Binary Visual Questions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[41] Susan Zhang et al. *OPT: Open Pre-trained Transformer Language Models*. 2022. arXiv: 2205.01068 [cs.CL].

[42] Wenhe Zhang et al. "Machine Number Sense: A Dataset of Visual Arithmetic Problems for Abstract and Relational Reasoning". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34 (Apr. 2020), pp. 1332–1340. DOI: 10.1609/aaai.v34i02.5489.

[43] Lianmin Zheng et al. *Judging LLM-as-a-judge with MT-Bench and Chatbot Arena*. 2023. arXiv: 2306.05685 [cs.CL].

[44] Yuke Zhu et al. "Visual7W: Grounded Question Answering in Images". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

A Data Generation

A.1 Ignored Words

The list of ignored words is: "the", "a", "of", "is", "let", "and", "at".

A.2 Rule Keywords

Tables 7 and 8 show the keywords that trigger the rules belonging to the *Individual* and *Relational* categories. For the *Modifier* rule keywords, they exist for icons and homophones (see Section 3.2), and for the *Repetition* rules. For the latter, these are: "two" and "double" for the *Repetition: Two* rule, and only "four" for the *Repetition: Four* rule.

Table 7: Keywords that trigger each *Individual* rule.

Direction			**Highlight**			**Size**			
Up	**Down**	**Reverse**	**Before**	**Middle**	**After**	**Big**	**Small**	**Color**	**Cross**
Up	Down	Back	Before	Middle	After	Big	Little	Black	Cross
		Mirror	Begin	Mid	End	Large	Micro	Blue	Crossed
		Inverse	Start		Behind	Grand	Smaller	Orange	Crossing
		Rear	Left			Bigger	Smallest	Green	
		Left	Starting			Biggest	Miniature	Red	
		Flip	Beginning			Giant		Purple	
						Jumbo		Brown	
								Pink	
								Gray	
								Yellow	
								Gold	

Table 8: Keywords that trigger each *Relational* rule.

Next to	**Inside**	**Above**	**Outside**
Next	In	Above	Out
	Inside	Over	Outside
	Into	On	
		Upon	

B Prompt Strategies

B.1 Templates

The following information is passed to the prompts in Table 9, ordered by the amount of information they give on a puzzle:

1. No description on the nature of the puzzle. No description of the graph.
2. Description on the nature of the puzzle. No description of the graph.
3. Description on the nature of the puzzle. Description of the graph is passed, but only the nodes.
4. Description on the nature of the puzzle. Description of the full graph is passed, which includes both nodes and edges.

Table 9: Prompt templates used during experimentation, as seen in Tables 3-6.

Prompt	**Template**
Prompt 1 (no knowledge)	*[IMG]* Which word/phrase is conveyed in this image from the following options (either A, B, C, or D)? (A) {} (B) {} (C) {} (D) {}
Prompt 2 (puzzle knowledge)	*[IMG]* You are given a rebus puzzle. It consists of text or icons that is used to convey a word or phrase. It needs to be solved through creative thinking. Which word/phrase is conveyed in this image from the following options (either A, B, C, or D)? (A) {} (B) {} (C) {} (D) {}
Prompt 3 (graph node knowledge)	*[IMG]* You are given an image of a rebus puzzle. It consists of text or icons that is used to convey a word or phrase. It needs to be solved through creative thinking. You are also given a description of the graph representation of the puzzle. The nodes are elements that contain text or icons, which are then manipulated through the attributes of their node. The description is as follows: {} Which word/phrase is conveyed in this image and description from the following options (either A, B, C, or D)? (A) {} (B) {} (C) {} (D) {}
Prompt 4 (graph nodes + edges knowledge)	*[IMG]* You are given an image of a rebus puzzle. It consists of text or icons that is used to convey a word or phrase. It needs to be solved through creative thinking. You are also given a description of the graph representation of the puzzle. The nodes are elements that contain text or icons, which are then manipulated through the attributes of their node. The edges define spatial relationships between these elements. The description is as follows: {} Which word/phrase is conveyed in this image and description from the following options (either A, B, C, or D)? (A) {} (B) {} (C) {} (D) {}

B.2 Graph Descriptions

Table 10: Descriptions of graphs used in prompts 3 and 4 from Table 9. For clarity, the two examples from Figure 5 are used in this table.

Idiom	**Graph description (only nodes)**	**Graph description (nodes + edges)**
Cross fingers for luck	Node 1 attributes: (text: FINGERS, cross: True, repeat: 1) Node 2 attributes: (text: ☘, repeat: 4, icon: (luck: ☘), sound: (for: 4))	Node 1 attributes: (text: FINGERS, cross: True, repeat: 1) Node 2 attributes: (text: ☘, repeat: 4, icon: (luck: ☘), sound: (for: 4)) Edge 1: node 1 to node 2 (rule: NEXT-TO)
Pull the wool over eyes	Node 1 attributes: (text: PULL, repeat: 1) Node 2 attributes: (text: WOOL, repeat: 1) Node 3 attributes: (text: IIII, repeat: 1, sound: (eyes: iiii))	Node 1 attributes: (text: PULL, repeat: 1) Node 2 attributes: (text: WOOL, repeat: 1) Node 3 attributes: (text: IIII, repeat: 1, sound: (eyes: iiii)) Edge 1: node 1 to node 2 (rule: NEXT-TO) Edge 2: node 2 to node 3 (rule: ABOVE)