Neural Analogical Matching

Anonymous Authors[1]

Abstract

Analogy is core to human cognition. It allows us to solve problems based on prior experience, it governs the way we conceptualize new information, and it even influences our visual perception. The importance of analogy to humans has made it an active area of research in the broader field of artificial intelligence, resulting in data-efficient models that learn and reason in human-like ways. While analogy and deep learning have generally been considered independently of one another, the integration of the two lines of research seems like a promising step towards more robust and efficient learning techniques. As part of the first steps towards such an integration, we introduce the Analogical Matching Network: a neural architecture that learns to produce analogies between structured, symbolic representations that are largely consistent with the principles of Structure-Mapping Theory.

1. Introduction

Analogical reasoning is a form of inductive reasoning that cognitive scientists consider to be one of the cornerstones of human intelligence (Gentner, 2003; Hofstadter, 2001; 1995). Analogy shows up at nearly every level of human cognition, from low-level visual processing (Sagi et al., 2012) to abstract conceptual change (Gentner et al., 1997). Problem solving using analogy is common, with past solutions being used to solve new problems (Holyoak et al., 1984; Novick, 1988). Analogy also facilitates learning and understanding by allowing people to generalize specific situations into increasingly abstract schemas (Gick & Holyoak, 1983).

Many different theories have been proposed for how humans perform analogy (Mitchell, 1993; Chalmers et al., 1992; Gentner, 1983; Holyoak et al., 1996). One of the most influential theories is Structure-Mapping Theory (SMT) (Gentner, 1983), which posits that analogy involves the alignment of structured representations of objects or situations subject to certain constraints. In this work, we introduce the Analogical Matching Network (AMN), a neural architecture that learns to produce analogies between symbolic representations that are largely consistent with SMT.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review.

2. Related Work

Many different computational models of analogy have been proposed (Holyoak & Thagard, 1989; O'Donoghue & Keane, 1999; Forbus et al., 2017), each instantiating a different cognitive theory of analogy. The differences between them are compounded by the computational costs of analogical reasoning, a provably NP-HARD problem (Veale & Keane, 1997). Many of the early approaches to analogy were connectionist (Gentner & Markman, 1993). The STAR architecture of (Halford et al.) used tensor product representations of structured data to perform simple analogies of the form $R(x, y) \Rightarrow S(f(x), f(y))$. Drama (Eliasmith & Thagard, 2001) was an implementation of the multi-constraint theory of analogy (Holyoak et al., 1996) that employed a holographic representation similar to tensor products to embed structure. LISA (Hummel & Holyoak, 1997; 2005) was a hybrid symbolic connectionist approach to analogy. It staged the mapping process temporally, generating mappings from elements of the compared representations that were activated at the same time. Only a few recent deep learning works incorporated cognitive theories of analogy (Hill et al., 2019; Zhang et al., 2019). Generally, prior deep learning work has only considered analogy as solving simple problems of the form $A : B :: C : D$ (Mikolov et al., 2013; Reed et al., 2015). Still, such prior works made progress in applying analogy to more perceptual data, e.g., language.

3. Structure-Mapping Theory

In Structure-Mapping Theory (SMT) (Gentner, 1983), analogy centers around the structural alignment of *relational representations* (see Figure 1). A relational representation is a set of logical expressions constructed from entities (e.g., `sun`), attributes (e.g., `YELLOW`), functions (e.g., `TEMPERATURE`), and relations (e.g., `GREATER`). Structural alignment is the process of producing a *mapping* between two relational representations (referred to as the *base* and *target*). A mapping is a triple $\langle M, C, S \rangle$, where M is a

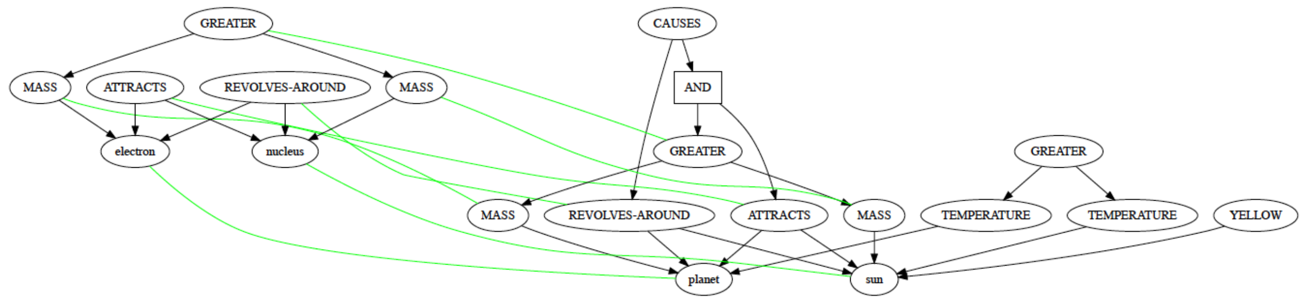

Figure 1. Graph representations for models of the atom (left) and solar system (right). Light green edges indicate correspondences

set of *correspondences* between the base and target, C is a set of *candidate inferences* (i.e., inferences about the target derived from the structure of the base), and S is a *structural evaluation score* for the quality of M. Correspondences are pairs of expressions or entities between the base and target. While entities can correspond irrespective of their labels, there are more rigorous criteria for matching expressions.

SMT asserts that M should satisfy the following: *1) One-to-One*: Each element of the base and target can be a part of *at most* one correspondence. *2) Parallel Connectivity*: Two expressions can be in a correspondence with each other only if their arguments are also in a correspondence with each other. *3) Tiered Identicality*: Relations of expressions in a correspondence must match identically, but functions need not be identical if their correspondence would support structural connectivity. *4) Systematicity*: Preference should be given to mappings with more deeply nested expressions.

In this work, the base and target expressions are considered semi-ordered directed-acyclic graphs (DAGs) $G_B = \langle V_B, E_B \rangle$ and $G_T = \langle V_T, E_T \rangle$, with V_B and V_T being sets of nodes and E_B and E_T being sets of edges. Each node corresponds to an element in the base or target, with its label being its relation, function, attribute, or entity name.

4. Model

4.1. Model Components

Given a base $G_B = \langle V_B, E_B \rangle$ and target $G_T = \langle V_T, E_T \rangle$, AMN produces a set of correspondences $M \subseteq V_B \times V_T$ and a set of candidate inferences $I \in V_B \setminus \{b_i : \langle b_i, t_j \rangle \in M\}$. Our architecture uses Transformers (Vaswani et al., 2017) and pointer networks (Vinyals et al., 2015) and takes inspiration from the work of (Kool et al., 2018).

Representing Structure: AMN first parses both the base and target into two separate graphs, a *label graph* and a *signature graph*. The label graph is used to get an estimate of the structural similarity of two expressions. To generate the label graph, AMN first substitutes each entity node's label with a generic entity token (reflecting that entity labels have no utility for producing matchings). Then, each function and predicate node is assigned a randomly chosen generic label (from a fixed set of labels) based off of its arity and orderedness. Assignments are made consistently across the entire graph, e.g., *every* instance of the function `MASS` across *both* the base and target would be given the same generic replacement label. This substitution means the original label is not used during matching, which allows AMN to generalize to unseen symbols. In addition, a signature graph is constructed which represents nodes by their object identities. To construct the signature graph, AMN replaces each distinct entity with a unique identifier (drawn from a fixed set of possible identifiers). It then assigns each function / predicate a new label based on arity and orderedness. Unlike the label graph, two differently labeled symbols would be given the same label if they have the same properties.

AMN uses two separate DAG LSTMs (Crouse et al., 2019) to embed the nodes of the label and signature graphs (equations in Appendix 6.3.1). The set of label structure embeddings is written as $L_V = \{l_v : v \in V\}$ and the set of signature embeddings is written as $S_V = \{s_v : v \in V\}$. Before passing these embeddings to the next step, each element of S_V is scaled to unit length, i.e. s_v becomes $s_v / \|s_v\|$.

Correspondence Selector: We utilize the set of embedding pairs for each node of V_B and V_T, writing l_v to denote the label structure embedding of node v taken from L_V and s_v the signature embedding of node v taken from S_V. We first define the set of unprocessed correspondences $\mathcal{C}^{(0)}$

$$\begin{aligned} \hat{\mathcal{C}} &= \{\langle b, t \rangle \in V_B \times V_T : \|l_b - l_t\| \leq \epsilon\} \\ \mathcal{C}^{(0)} &= \{\langle [l_b; l_t; s_b; s_t], s_b, s_t \rangle : \langle b, t \rangle \in \hat{\mathcal{C}}\} \end{aligned}$$

where $[\cdot;\cdot]$ denotes vector concatenation and ϵ is the tiered identicality threshold that governs how much the subgraphs rooted at two nodes may differ and still be considered for correspondence. The first element of each correspondence in $\mathcal{C}^{(0)}$, i.e., $h_c = [l_b; l_t; s_b; s_t]$, is passed through the N-layered Transformer encoder (equations in Appendix 6.3.3).

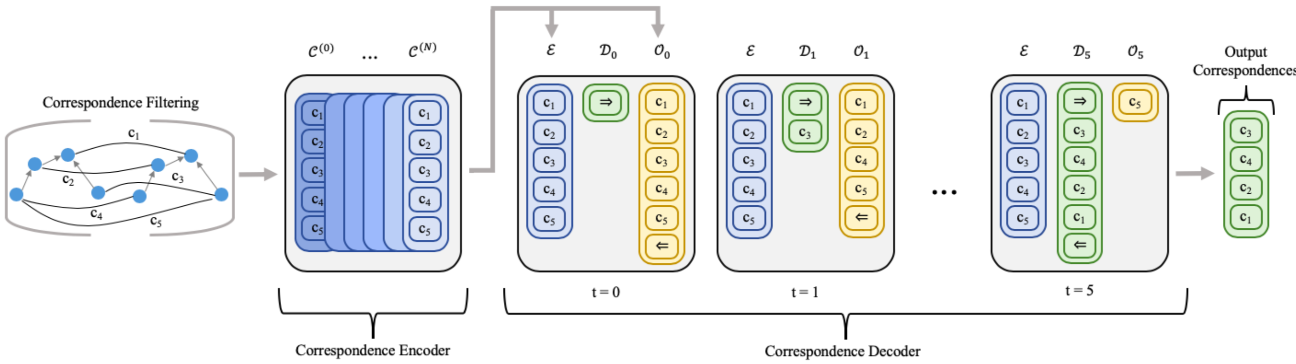

Figure 2. The correspondence selection process, where $\Rightarrow$ and $\Leftarrow$ are the start and stop tokens and $\mathcal{E}$, $\mathcal{D}_t$, and $\mathcal{O}_t$ are the sets of encoded, selected, and remaining correspondences

This produces a set of encoded correspondences as $\mathcal{E} = \{\langle h_c^{(N)}, s_b, s_t \rangle \in \mathcal{C}^{(N)}\}$.

The Transformer decoder (equations in Appendix 6.3.3) will select some subset of the set of correspondences that produces the best analogical match (see Figure 2). The layers of attention-based transformations are performed on only the initial elements of each tuple, i.e., h_d in $\langle h_d, s_b, s_t \rangle$. We let $\mathcal{D}_t$ be the processed set of all selected correspondences (after the N attention layers) and $\mathcal{O}_t$ be the set of all remaining correspondences at timestep t (with $\mathcal{D}_0 = \{\texttt{START-TOK}\}$ and $\mathcal{O}_0 = \mathcal{E} \cup \{\texttt{END-TOK}\}$). The decoder generates compatibility scores α_{od} between each pair of elements, i.e., $\langle o, d \rangle \in \mathcal{O}_t \times \mathcal{D}_t$. These are combined with the signature embedding similarities to produce a final compatibility π_{od}

$$\pi_{od} = \text{FFN}\big(\big[\tanh(\alpha_{od}); s_{b_o}^{\top} s_{b_d}; s_{t_o}^{\top} s_{t_d}\big]\big)$$

where FFN is a two layer feed-forward network with ELU activations (Clevert et al., 2015). Recall that the signature components, i.e. s_b and s_t, were scaled to unit length. Thus, we would expect closeness in the original graph to be reflected by dot-product similarity and identicality to be indicated by a maximum value dot-product, i.e. $s_{b_o}^{\top} s_{b_d} = 1$ or $s_{t_o}^{\top} s_{t_d} = 1$. For each $o \in \mathcal{O}_t$, we compute its value as

$$v_o = \text{FFN}\big(\big[\max_d \pi_{od}; \min_d \pi_{od}; \sum_d \frac{\pi_{od}}{|\mathcal{D}_t|}\big]\big)$$

where FFN is a two layer feed-forward network with ELU activations. From these, a softmax produces probabilities and the most probable element is added to $\mathcal{D}_{t+1}$. When $\texttt{END-TOK}$ is selected, the set of correspondences M returned are the node pairs in $V_B \times V_T$ associated with $\mathcal{D}$.

Candidate Inference Selector: The output of the correspondence selector is a set of correspondences M. The candidate inferences associated with M are drawn from the nodes of the base graph V_B that were *not* used in M. Let V_{in} and V_{out} be the subsets of V_B that were and were not used in M. AMN first extracts the signature embeddings for both sets, i.e., $\mathcal{S}_{in} = \{s_b : b \in V_{in}\}$ and $\mathcal{S}_{out} = \{s_b : b \in V_{out}\}$.

AMN will select elements from $\mathcal{S}_{out}$ to return. Like before, we let $\mathcal{D}_t$ be the set of all selected elements from $\mathcal{S}_{out}$ and $\mathcal{O}_t$ be the set of all remaining elements from $\mathcal{S}_{out}$ at timestep t. AMN computes compatibility scores between pairs of output options with candidate inference and previously selected nodes, i.e. α_{od} for each $\langle o, d \rangle \in \mathcal{O}_t \times (\mathcal{D}_t \cup \mathcal{S}_{in})$. The compatibility scores are given by a simple single-headed attention computation (see Appendix 6.3.2). The compatibilities are used directly to compute a value v_o for each element of $\mathcal{O}_t$. AMN computes the value for a node o as

$$\alpha'_{od} = \tanh(\alpha_{od})$$
$$v_o = \text{FFN}\big(\big[\max_d \alpha'_{od}; \min_d \alpha'_{od}; \sum_d \frac{\alpha'_{od}}{|\mathcal{D}_t|}\big]\big)$$

A softmax is used and the most probable element is added to $\mathcal{D}_{t+1}$, ending when $\texttt{END-TOK}$ is selected.

4.2. Model Scoring

Structural Match Scoring: In order to avoid counting erroneous correspondence predictions towards the score of the output correspondences M, we first identify all correspondences that are either degenerate or violate the constraints of SMT. Degenerate correspondences are between constants with no higher-order structural support in M (i.e., if either has no parent participating in a correspondence in M). Let the valid subset of M be M_{val}. A *root correspondence* m is one such that there does not exist another correspondence m' such that $m' \in M_{val}$ and a node in m' is an ancestor of a node in m. For $m = \langle b, t \rangle$ in M_{val}, its score $s(m)$ is given as the size of the subgraph rooted at b in the base. The structural match score for M is the sum of scores for all root correspondences. This repeatedly counts nodes appearing in the dependencies of multiple correspondences, which leads to higher scores for more interconnected matchings.

Table 1. AMN correspondence prediction results for performance ratio, solution type rate (↑ better), and error rate (↓ better), and AMN candidate inference prediction results

Domain	r	Struct. Perf.	Larger	Equiv.	Err. Free	1-to-1	PC	Degen.	CI F1	CI Rec.	CI Prec.	CI Acc.
Synthetic	1	0.702	0.000	0.308	0.342	0.007	0.106	0.018	0.901	0.866	0.969	0.860
Synthetic	16	0.948	0.001	0.671	0.684	0.006	0.021	0.009	0.899	0.867	0.964	0.860
Oddity	1	0.775	0.062	0.404	0.483	0.152	0.223	0.000	0.971	0.968	0.991	0.962
Oddity	16	0.957	0.075	0.492	0.571	0.130	0.139	0.000	0.991	0.995	0.993	0.991
Moral DM	1	0.617	0.014	0.017	0.076	0.001	0.169	0.030	0.889	0.817	0.987	0.816
Moral DM	16	0.968	0.081	0.210	0.352	0.000	0.039	0.015	0.897	0.832	0.984	0.830
Geometric	1	0.870	0.066	0.539	0.654	0.041	0.116	0.000	0.940	0.928	0.989	0.924
Geometric	16	1.038	0.069	0.707	0.783	0.029	0.043	0.000	0.960	0.954	0.993	0.951

Structural Evaluation Maximization: Dynamically assigning labels to each example allows AMN to handle never-before-seen symbols, but its randomness can lead to variability in terms of outputs. AMN combats this by running each test problem r times and returning the predicted match M that maximizes the structural evaluation score, i.e., $M = \arg\max_{M_r} s(M_r)$. Notably, AMN does not attempt to alter or correct the mapping it chooses this way, so unlike SME, the mapping it returns can include SMT violations.

5. Experiments

AMN was trained on 100,000 synthetic analogy examples. A single example consisted of base and target graphs, a set of correspondences between the base and target, and a set of nodes from the base to be considered candidate inferences.

Though all training was done with synthetic data, we evaluated the effectiveness of AMN on both synthetic data and data used in previous analogy experiments. The corpus of previous analogy examples was taken from the public release of SME[1]. Importantly, AMN was *not* trained on the corpus of existing analogy examples (AMN never learned from a real-world analogy example). In fact, there was *no* overlap between the symbols used in that corpus and the symbols used for the synthetic data. The four domains tested in this work are the *Synthetic*, *Visual Oddity*, *Moral Decision Making*, *Geometric Analogies* domains. Each are described in Appendix 6.2 and examples of AMN's output for each domain can be found in Appendix 6.4.

5.1. Results and Discussion

Table 1 shows the results for AMN across different values of r, where r denotes the re-run hyperparameter detailed in Section 4.2. When evaluating on the synthetic data, the comparison set of correspondences was given by the data generator; whereas when evaluating on the three other analogy domains, the comparison set of correspondences was given by the output of SME. It is important to note that we are using SME as our stand-in for SMT (as it is the most widely accepted computational model of SMT). Thus, we do *not* want significantly different results from SME, e.g. substantially higher or lower structural evaluation scores. Candidate inference prediction performance was measured relative to the set of correspondences AMN generated.

[1]http://www.qrg.northwestern.edu/software/sme4/index.html

Analysis: The left side of Table 1 shows the average ratio of AMN's performance (labeled Struct. Perf.), as measured by structural evaluation score, against the comparison method's performance (i.e., data generator correspondences or SME) across domains. As can be seen, AMN was around 95-104% of SME's performance in terms of structural evaluation score on the three preexisting domains, which indicates that it was finding similar structural matches.

The middle-left of Table 1 gives us the best sense of how well AMN modeled SMT. We observe AMN's performance in terms of the proportion of *larger*, *equivalent*, and *error-free* matches it produces (labeled Larger, Equiv., and Err. Free, respectively). Error-free matches do not contain degenerate correspondences or SMT constraint violations, whereas equivalent and larger matches are both error-free and have the same / larger structural evaluation score as compared to gold set of correspondences. The Equiv. column provides the best indication that AMN could model SMT. It shows that $\gtrapprox 50\%$ of AMN's outputs were SMT-satisfying analogical matches with the *exact same* structural score as SME in two of the three non-synthetic domains.

The right side of Table 1 shows the frequency of the different types of errors, including violations of the one-to-one and parallel connectivity constraints, and degenerate correspondences (labeled 1-to-1, PC, and Degen., respectively). Importantly, degenerate correspondences were not an issue for any domain, which verifies that AMN leveraged higher-order relational structure when generating matches.

The candidate inference (CI) metrics (averaged across all problems) shows that AMN was fairly effective in predicting candidate inferences. The high accuracy scores across domains indicate that AMN could capture the notion structural support for candidate inferences.

References

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Chalmers, D. J., French, R. M., and Hofstadter, D. R. High-level perception, representation, and analogy: A critique of artificial intelligence methodology. *Journal of Experimental & Theoretical Artificial Intelligence*, 4(3):185–211, 1992.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Crouse, M., Abdelaziz, I., Cornelio, C., Thost, V., Wu, L., Forbus, K., and Fokoue, A. Improving graph neural network representations of logical formulae with subgraph pooling. *arXiv preprint arXiv:1911.06904*, 2019.

Dehaene, S., Izard, V., Pica, P., and Spelke, E. Core knowledge of geometry in an amazonian indigene group. *Science*, 311(5759):381–384, 2006.

Dehghani, M., Tomai, E., Forbus, K., Iliev, R., and Klenk, M. Moraldm: A computational modal of moral decision-making. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, 2008a.

Dehghani, M., Tomai, E., Forbus, K. D., and Klenk, M. An integrated reasoning approach to moral decision-making. In *AAAI*, pp. 1280–1286, 2008b.

Eliasmith, C. and Thagard, P. Integrating structure and meaning: A distributed model of analogical mapping. *Cognitive Science*, 25(2):245–286, 2001.

Evans, T. G. A program for the solution of a class of geometric-analogy intelligence-test questions. Technical report, AIR FORCE CAMBRIDGE RESEARCH LABS LG HANSCOM FIELD MASS, 1964.

Forbus, K., Usher, J., Lovett, A., Lockwood, K., and Wetzel, J. Cogsketch: Sketch understanding for cognitive science research and for education. *Topics in Cognitive Science*, 3(4):648–666, 2011.

Forbus, K. D., Ferguson, R. W., Lovett, A., and Gentner, D. Extending sme to handle large-scale cognitive modeling. *Cognitive Science*, 41(5):1152–1201, 2017.

Gentner, D. Structure-mapping: A theoretical framework for analogy. *Cognitive science*, 7(2):155–170, 1983.

Gentner, D. Why we're so smart. *Language in mind: Advances in the study of language and thought*, 195235, 2003.

Gentner, D. and Markman, A. B. Analogy–watershed or waterloo? structural alignment and the development of connectionist models of analogy. In *Advances in neural information processing systems*, pp. 855–862, 1993.

Gentner, D., Brem, S., Ferguson, R. W., Markman, A. B., Levidow, B. B., Wolff, P., and Forbus, K. D. Analogical reasoning and conceptual change: A case study of johannes kepler. *The journal of the learning sciences*, 6 (1):3–40, 1997.

Gick, M. L. and Holyoak, K. J. Schema induction and analogical transfer. 1983.

Halford, G. S., Wilson, W. H., Guo, J., Gayler, R. W., Wiles, J., and Stewart, J. Connectionist implications for processing capacity limitations in analogies.

Hill, F., Santoro, A., Barrett, D. G., Morcos, A. S., and Lillicrap, T. Learning to make analogies by contrasting abstract relational structure. *International Conference on Learning Representations*, 2019.

Hofstadter, D. *Fluid concepts and creative analogies: Computer models of the fundamental mechanisms of thought.* Basic books, 1995.

Hofstadter, D. R. Analogy as the core of cognition. *The analogical mind: Perspectives from cognitive science*, pp. 499–538, 2001.

Holyoak, K. J. and Thagard, P. Analogical mapping by constraint satisfaction. *Cognitive science*, 13(3):295–355, 1989.

Holyoak, K. J., Junn, E. N., and Billman, D. O. Development of analogical problem-solving skill. *Child development*, pp. 2042–2055, 1984.

Holyoak, K. J., Holyoak, K. J., and Thagard, P. *Mental leaps: Analogy in creative thought*. 1996.

Hummel, J. E. and Holyoak, K. J. Distributed representations of structure: A theory of analogical access and mapping. *Psychological review*, 104(3):427, 1997.

Hummel, J. E. and Holyoak, K. J. Relational reasoning in a neurally plausible cognitive architecture: An overview of the lisa project. *Current Directions in Psychological Science*, 14(3):153–157, 2005.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kool, W., Van Hoof, H., and Welling, M. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

Lovett, A. and Forbus, K. Cultural commonalities and differences in spatial problem-solving: A computational analysis. *Cognition*, 121(2):281–287, 2011.

Lovett, A. and Forbus, K. Modeling multiple strategies for solving geometric analogy problems. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 34, 2012.

Lovett, A., Tomai, E., Forbus, K., and Usher, J. Solving geometric analogy problems through two-stage analogical mapping. *Cognitive science*, 33(7):1192–1231, 2009.

Mikolov, T., Yih, W.-t., and Zweig, G. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pp. 746–751, 2013.

Mitchell, M. *Analogy-making as perception: A computer model*. 1993.

Novick, L. R. Analogical transfer, problem similarity, and expertise. *Journal of Experimental Psychology: Learning, memory, and cognition*, 14(3):510, 1988.

O'Donoghue, T. V. D. and Keane, M. Computability as a limiting cognitive constraint: Complexity concerns in metaphor comprehension about which cognitive linguists should be aware. In *Cultural, Psychological and Typological Issues in Cognitive Linguistics: Selected papers of the bi-annual ICLA meeting in Albuquerque, July 1995*, volume 152, pp. 129. John Benjamins Publishing, 1999.

Reed, S. E., Zhang, Y., Zhang, Y., and Lee, H. Deep visual analogy-making. In *Advances in neural information processing systems*, pp. 1252–1260, 2015.

Sagi, E., Gentner, D., and Lovett, A. What difference reveals about similarity. *Cognitive science*, 36(6):1019–1050, 2012.

Tai, K. S., Socher, R., and Manning, C. D. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1556–1566, 2015.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Veale, T. and Keane, M. T. The competence of sub-optimal theories of structure mapping on hard analogies. In *IJCAI (1)*, pp. 232–237, 1997.

Vinyals, O., Fortunato, M., and Jaitly, N. Pointer networks. In *Advances in neural information processing systems*, pp. 2692–2700, 2015.

Zhang, C., Gao, F., Jia, B., Zhu, Y., and Zhu, S.-C. Raven: A dataset for relational and analogical visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5317–5327, 2019.

6. Appendix

6.1. Model Details

In the DAG LSTM, the node embeddings were 32-dimensional vectors and the edge embeddings were 16-dimensional vectors. For all Transformer components, our model used multi-headed attention with 2 attention layers each having 4 heads. In each multi-headed attention layer, the query and key vectors were projected to 128-dimensional vectors. The feed forward networks used in the Transformer components had one hidden layer with a dimensionality twice that of the input vector size. The feed forward networks used to compute the values in the correspondence selector used two 64-dimensional hidden layers.

Training Loss: As both the correspondence and candidate inference components use a softmax, the loss function is categorical cross entropy. Teacher forcing is used to guide the decoder to select the correct choices during training. The losses for both the correspondence and candidate inference components are summed together to produce the final loss which is minimized with Adam (Kingma & Ba, 2014).

6.2. Data Details

Synthetic Data: To generate a synthetic example (see Figure 3) for training, we first generate a set of random graphs C, which will form the basis for the correspondences. Next, we construct the base B by further generating graphs around C. Likewise, for the target T we also build another set of graphs around the C. The graphs of C are then used to form the correspondences between the base and target. Any element in B that is an ancestor of a node from C or a descendent of such an ancestor is considered a candidate inference.

Experimental Domains: We describe each domain used in this paper here (a more detailed description can be found in (Forbus et al., 2017))

1. *Synthetic*: this domain consisted of 1000 examples generated with the same parameters as the training data.
2. *Visual Oddity*: this problem setting was initially proposed to explore cultural differences to geometric rea-

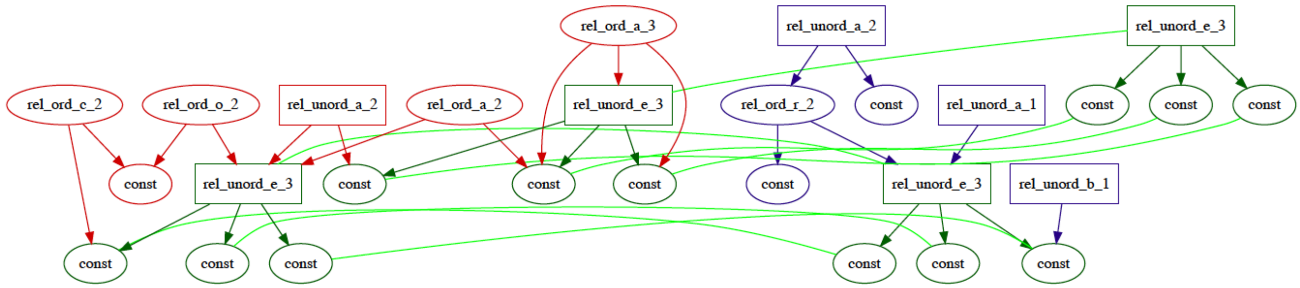

Figure 3. Synthetic example with a base (red), target (blue), and shared subgraphs (green)

soning in (Dehaene et al., 2006). The work of (Lovett & Forbus, 2011) modeled the findings of the original experiment computationally, and from their work we extracted 3405 analogical comparisons.

3. *Moral Decision Making*: this domain was taken from the work of (Dehghani et al., 2008a), who introduced a computational model of moral decision making driven by SME. From the works of (Dehghani et al., 2008a;b), we extracted 420 analogical comparisons.

4. *Geometric Analogies*: this domain originated from (Evans, 1964). Each problem was an incomplete analogy between manually encoded geometric figures. In (Lovett et al., 2009; Lovett & Forbus, 2012) it was shown that the analogy problems could be solved with structure-mapping over automatic encodings (produced by the CogSketch system (Forbus et al., 2011)). From that work we extracted 866 analogies.

6.3. Background

6.3.1. DAG LSTMs

DAG LSTMs extend Tree LSTMs (Tai et al., 2015) to DAG-structured data. As with Tree LSTMs, DAG LSTMs compute each node embedding as the aggregated information of all their immediate predecessors (the equations for the DAG LSTM are identical to those of the Tree LSTM). The difference between the two is that DAG LSTMs stage the computation of a node's embedding based on the order given by a topological sort of the input graph. Batching of computations is done by grouping together updates of independent nodes (where two nodes are independent if they are neither ancestors nor predecessors of one another). As in (Crouse et al., 2019), for a node, v, its initial node embedding, s_v, is assigned based on its label and arity. The DAG LSTM then computes the final embedding h_v to be

$$\begin{aligned}
i_v &= \sigma\big(W_i s_v + \sum_{w \in \mathcal{P}(v)} U_i^{(e_{vw})} h_w + b_i\big) \\
o_v &= \sigma\big(W_o s_v + \sum_{w \in \mathcal{P}(v)} U_o^{(e_{vw})} h_w + b_o\big) \\
\hat{c}_v &= \tanh\big(W_c s_v + \sum_{w \in \mathcal{P}(v)} U_c^{(e_{vw})} h_w + b_c\big) \\
f_{vw} &= \sigma\big(W_f s_v + U_f^{(e_{vw})} h_w + b_f\big) \\
c_v &= i_v \odot \hat{c}_v + \sum_{w \in \mathcal{P}(v)} f_{vw} \odot c_w \\
h_v &= o_v \odot \tanh\big(c_v\big)
\end{aligned}$$

where $\odot$ is element-wise multiplication, σ is the sigmoid function, $\mathcal{P}$ is the predecessor function that returns the arguments for a node, $U_i^{(e_{vw})}$, $U_o^{(e_{vw})}$, $U_c^{(e_{vw})}$, and $U_f^{(e_{vw})}$ are learned matrices per edge type. i and o represent input and output gates, c and $\hat{c}$ are memory cells, and f is a forget gate.

6.3.2. Multi-Headed Attention

The multi-headed attention (MHA) mechanism of (Vaswani et al., 2017) is used in our work to compare correspondences against one another. In this work, MHA is given two inputs, a query vector q and a list of key vectors to compare the query vector against $\langle k_1, \ldots, k_n \rangle$. In N-headed attention, N separate attention transformations are computed. For transformation i we have

$$\hat{q}_i = W_i^{(q)} q, k_{ij} = W_i^{(k)} k_j, v_{ij} = W_i^{(v)} k_j$$

$$w_{ij} = \frac{\hat{q}_i^\top k_{ij}}{\sqrt{b_{\hat{q}}}}$$

$$\alpha_{ij} = \frac{\exp\left(w_{ij}\right)}{\sum_{j'} \exp(w_{ij'})}$$

$$q_i = \sum_j \alpha_{ij} \hat{q}_i$$

where each of $W_i^{(q)}$, $W_i^{(k)}$, and $W_i^{(v)}$ are learned matrices and $b_{\hat{q}}$ is the dimensionality of $\hat{q}_i$. The final output vector q' for input q is then given as a combination of its N transformations

$$q' = \sum_{i=1}^{N} W_i^{(o)} q_i$$

where each $W_i^{(o)}$ is a distinct learned matrix for each i. In implementation, the comparisons of query and key vectors are batched together and performed as efficient matrix multiplications.

6.3.3. Transformer Encoder-Decoder

The Transformer-based encoder-decoder is given two inputs, a comparison set $\mathcal{C}$ and an output set $\mathcal{O}$. At a high level, $\mathcal{C}$ will be encoded into a new set $\mathcal{E}$, which will inform a selection process that picks elements of $\mathcal{O}$ to return. In the context of pointer networks, the set $\mathcal{O}$ begins as the encoded input set (i.e., $\mathcal{O} \equiv \mathcal{E}$).

Encoder: First, the elements of $\mathcal{C}$, i.e. $h_c \in \mathcal{C}$, are passed through N layers of an attention-based transformation. For element h_c in the i-th layer (i.e., $h_c^{(i-1)}$) this is performed as follows

$$\begin{aligned} \hat{h}_c &= \text{LN}\big(h_c^{(i-1)} + \text{MHA}_{\mathcal{C}}^{(i)}\big(h_c^{(i-1)}, \langle h_1^{(i-1)}, \ldots, h_j^{(i-1)} \rangle\big)\big) \\ h_c^{(i)} &= \text{LN}\big(\hat{h}_c + \text{FFN}^{(i)}\big(\hat{h}_c\big)\big) \end{aligned}$$

where LN denotes the use of layer normalization (Ba et al., 2016), $\text{MHA}_{\mathcal{C}}^{(i)}$ (Appendix 6.3.2) denotes the use of self multi-headed attention for layer i (i.e., attention between $h_c^{(i)}$ and the other elements of $\mathcal{C}^{(i-1)}$), and $\text{FFN}^{(i)}$ is a two-layer feed-forward neural network with ELU (Clevert et al., 2015) activations. After N layers of processing, the set of encoded inputs $\mathcal{E}$ is given by $\mathcal{E} = \mathcal{C}^{(N)}$

Decoder: With encoded comparison elements $\mathcal{E}$ and a set of potential outputs $\mathcal{O}$, the objective of the decoder is to use $\mathcal{E}$ to inform the selection of some subset of output options $\mathcal{D} \subseteq \mathcal{O}$ to return. Decoding happens sequentially; at each timestep $t \in \{1, \ldots, n\}$ the decoder selects an element from $\mathcal{O} \cup \{\texttt{END-TOK}\}$ (where `END-TOK` is a learned triple) to add to $\mathcal{D}$. If `END-TOK` is chosen, the decoding procedure stops and $\mathcal{D}$ is returned.

Let $\mathcal{D}_t$ be the set of elements that have been selected by timestep t and $\mathcal{O}_t$ be the remaining unselected elements at timetstep t. First, $\mathcal{D}_t$ is processed with an N-layered attention-based transformation. For an element $h_d^{(i-1)}$ this is given by

$$\begin{aligned} \acute{h}_d &= \text{LN}\big(h_d^{(i-1)} + \text{MHA}_{\mathcal{D}}^{(i)}\big(h_d^{(i-1)}, \langle h_1^{(i-1)}, \ldots, h_j^{(i-1)} \rangle\big)\big) \\ \hat{h}_d &= \text{LN}\big(\acute{h}_d + \text{MHA}_{\mathcal{E}}^{(i)}\big(\acute{h}_d, \langle h_1^{(i-1)}, \ldots, h_l^{(i-1)} \rangle\big)\big) \\ h_d^{(i)} &= \text{LN}\big(\hat{h}_d + \text{FFN}^{(i)}\big(\hat{h}_d\big)\big) \end{aligned}$$

where $\text{MHA}_{\mathcal{D}}^{(i)}$ denotes the use of self multi-headed attention, $\text{MHA}_{\mathcal{E}}^{(i)}$ denotes the use of multi-headed attention against elements of $\mathcal{E}$, and $\text{FFN}^{(i)}$ is a two-layer feed-forward neural network with ELU activations. We will consider the already selected outputs to be the transformed selected outputs, i.e., $\mathcal{D}_t = \mathcal{D}_t^{(N)}$. For a pair, $\langle h_o, h_d \rangle \in \mathcal{O}_t \times \mathcal{D}_t$, we compute their compatibility as α_{od}

$$q_{od} = W_q h_d^{(n)}, k_{od} = W_k h_o$$
$$\alpha_{do} = \frac{q_{od}^{\top} k_{od}}{\sqrt{b_o}}$$

where W_q and W_k are learned matrices, b_o is the dimensionality of h_o, and FFN is a two layer feed-forward network with ELU activations. This defines a matrix $H \in \mathbb{R}^{|\mathcal{O}_t| \times |\mathcal{D}_t|}$ of compatibility scores. One can then apply some operation (e.g., max pooling) to produce a vector of values $v_t \in \mathbb{R}^{|\mathcal{O}_t|}$ which can be fed into a softmax to produce a distribution over options from $\mathcal{O}_t$. The highest probability element δ^* from the distribution is then added to the set of selected outputs, i.e., $\mathcal{D} = \mathcal{D}_t \cup \{\delta^*\}$.

6.4. AMN Example Outputs

For the outputs from the non-synthetic domains (all but the first figure), only small subgraphs of the original graphs are shown (the original graphs were too large to be displayed)

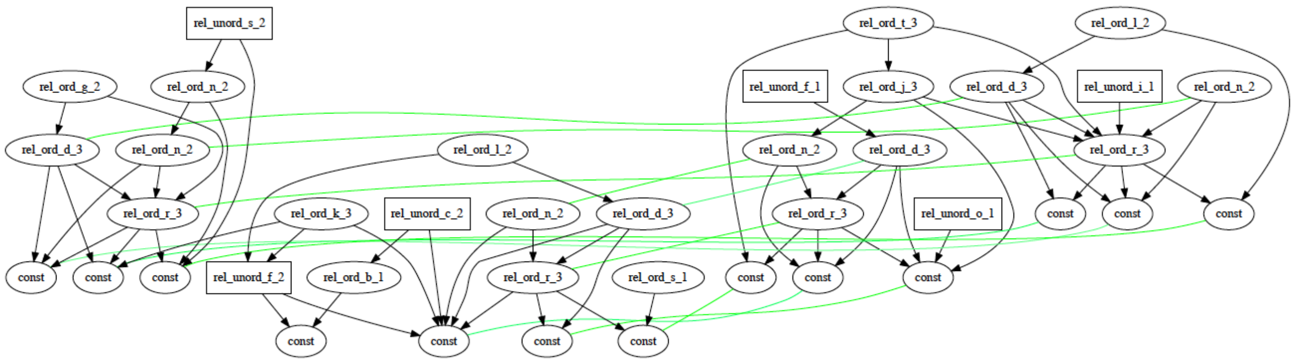

Figure 4. AMN output for an example from the Synthetic domain

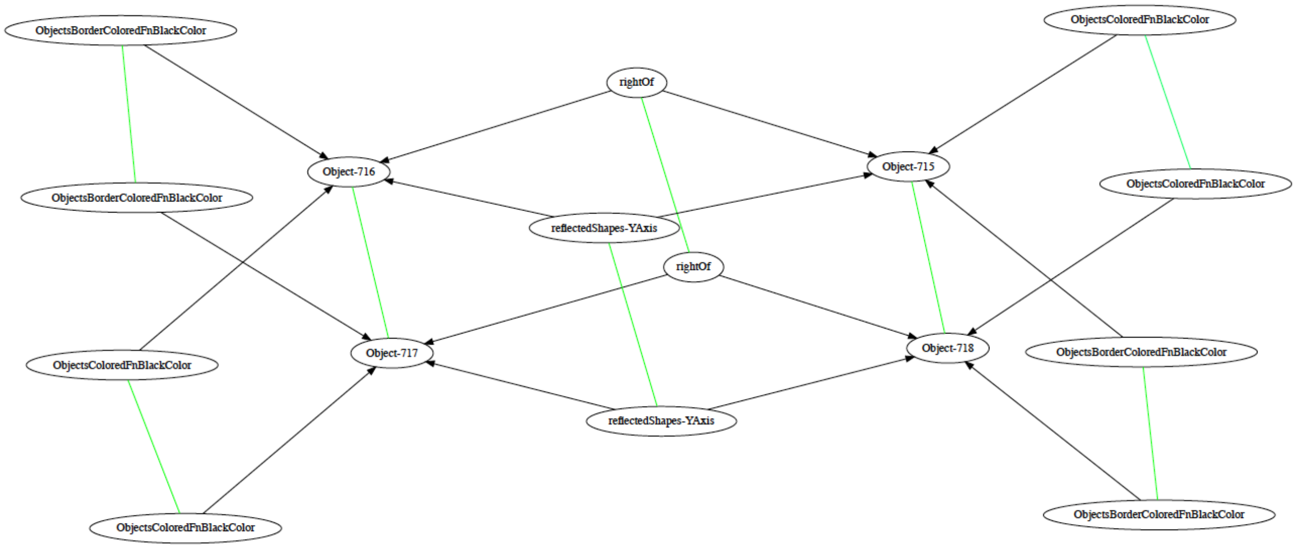

Figure 5. AMN output for an example from the Visual Oddity domain

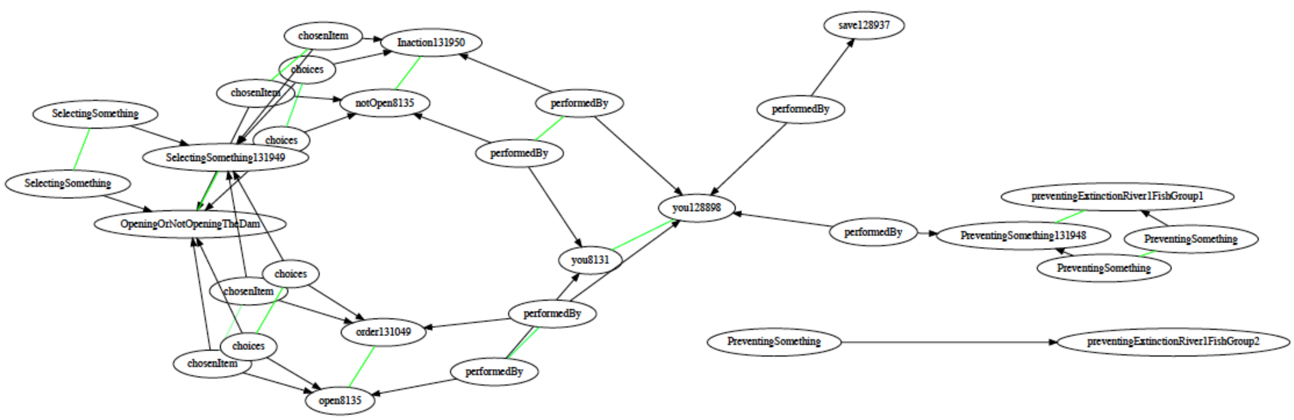

Figure 6. AMN output for an example from the Moral Decision Making domain

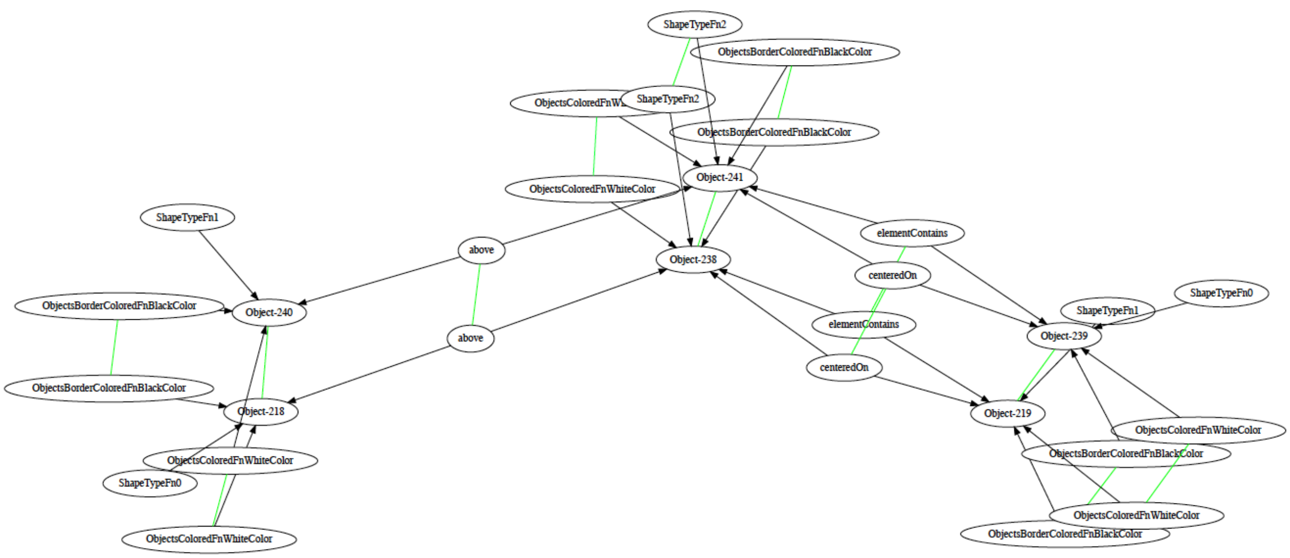

Figure 7. AMN output for an example from the Geometric Analogies domain