SpARC: Sparsity Activation Regularization for Consistency

Stanford CS224N Custom Project

Sarthak Consul
Dept. of Computer Science
Stanford University
`sarthakc@stanford.edu`

Samar Khanna
Dept. of Computer Science
Stanford University
`samar99@stanford.edu`

Julia Xu
Dept. of Computer Science
Stanford University
`juliaxu@stanford.edu`

Abstract

Transformers, while powerful language models, have a tendency to lack logical consistency in their beliefs [1]. Large language models such as T5 may simultaneously believe “A bird can fly" and “A bird cannot fly" [2]. In this work, we aim to finetune large language models to make them logically consistent. Specifically, we show that sparsifying model activations and increasing model modularity yields gains in both model accuracy and consistency. To achieve these improvements, we design a method of jointly penalising model activations through the L1 norm and employing a contrastive similarity loss between pairs of “similar" and “dissimilar" facts. We perform qualitative analysis of the model attention weights and find a correlation between increased sparsity and increased consistency. Code is available at `https://github.com/SConsul/SpARC/`.

1 Key Information

- TA mentor: Eric Mitchell
- No External Collaborators, No External Mentors, No sharing of project

2 Introduction

Language models store large amounts of linguistic data and contain extensive world knowledge from which relational knowledge can be captured [3]. Although it is possible to recover factual and commonsense knowledge, the problem that arises the ability to maintain a consistent set of beliefs in pretrained language models (PTLM) [1]. During pre-training, logical consistency in language models can be difficult to achieve [2, 4]. Logical consistency is the notion that a model will reason consistently over a set of implications. For example, given a set of implications: “A lion is a mammal” and “A mammal is a vertebrate,” we want to correctly answer the question: “Is a lion a vertebrate?” Logical consistency is an interesting and important problem because we want to create language models that are consistent and non-contradictory for more accurate and reliable outputs.

BeliefBank [1] attempts to address the issue of logical consistency by embedding a PTLM with a systemic notion of belief and consistent knowledge of the world. They augment the model with a system of global memory, which is a component that does not require finetuning the model. However, this work doesn’t directly target the issue of enforcing consistent beliefs during training. BeliefBank additionally requires a targeted and specific dataset containing facts and constraints, which may not be readily available. Instead, we enforce consistency during the finetuning stage, without explicitly requiring constraints. By enforcing logical consistency in the model during learning, we aim to address the root of the issue. We build on the idea of [5], which argues that imposing modularity in neural networks would allow for the modules to be reused for a greater consistency in networks. We finetune a pretrained MACAW-large [6] transformer with an auxiliary loss during the that penalizes the network for non-sparse activations when given similar prompts.

The goal of our project is to enforce logical consistency on a pretrained language model (PTLM). We regularise model activations to make PTLMs logically consistent such that similar beliefs correspond to similar sub-parts of the network. In regular PTLMs, the noisiness of encoding a large variety of inputs in any sub-part of the network results in poor consistency. We test the hypothesis that network modularity will prevent the model from conflating inconsistent facts. From our experiments, we conclude that modular activations promise better logical consistency. Our original contributions are summarized as follows:

1. Build upon the BeliefBank dataset from given consistency graphs and facts to create a dataset suited for consistency evaluation. Establish accuracy and consistency baselines.
2. Finetune a raw QA model baseline via adapters [7] that leads to increased accuracy and consistency over finetuning the entire model, or finetuning just the final layer.
3. Achieve a 4.3% gain in consistency over the finetuned baseline using our L1 and similarity-based loss metrics.
4. Perform qualitative analysis of model attention weights and demonstrate a correlation between increased sparsity and increased consistency.

3 Related Work

Logical Consistency Language models can be trained to perform consistent logical reasoning [1, 8, 9]. [8] investigates how logical reasoning can be improved by combining explicit natural language inputs with implicit pre-trained knowledge. The focus of our work is on logical consistency, but there are other approaches in consistency such as with paraphrased questions. Language models may be asked the same question phrased in different ways and produce different answers, and [9] examines and attempts to address these consistency issues through introducing a new dataset and pretraining with a novel loss.

We base our work off the BeliefBank dataset, which is described in detail in Section 4.1. BeliefBank [1] improves consistency of a PTLM through adding an external global memory layer that tracks beliefs. There are two components that manage the beliefs of this external memory - a reasoning component which updates beliefs that contradict one another and a feedback component that poses questions to the model using known of beliefs. The reasoning component is a weighted MaxSAT solver, and the two objectives are either to flip beliefs to minimize constraint violations or to not flip beliefs to retain the raw model's beliefs. In the constraint solving problem, each belief is assigned a weight. Given these beliefs, their weights, and their constraints, the constraint solver must output a set of truth assignments for each of these beliefs such that the weighted sum of violated constraints is minimized. The intended purpose of feedback mechanism is to remind the model of relevant beliefs when new questions are posed.

Activation Regularization The idea of reducing redundant learning in neural networks by inducing sparsity in the network has been commonly explored. Previous works have promoted sparsity in the activations through approaches such as using ReLU activations [10, 11] or promoting small activations by penalizing the L1 norm [12] or L2 norm of the activations [13]. The increase in network sparsity do lead to improved model accuracies, however the resulting networks fail to exhibit functional modularity.

Modularity Neural networks with explicitly designed modularity [14, 15] have shown great generalization capabilities. While [16, 17, 18] have shown that neural networks can be clustered to some degree, [5] demonstrates that popular neural network architectures (such as CNNs and transformers) are not implicitly functionally modular and that their weight-sharing is heavily dependent on the input interface rather than the similarity amongst the tasks. The authors of [5] argue that neural networks are trained inefficiently by showing how separate portions of networks end up learning similar rules and blame such redundancy as a cause for the lack of generalizability of neural networks.

4 Approach

In our approach, we generate question and answer data, construct a custom dataset from given consistency graphs and facts, establish initial baseline and finetuned results, and encourage modularity.

We encourage modularity through the two main methods: (1) sparsity via L1 loss and (2) encouraging modularity through functional similarity.

Macaw We base our approach on Macaw [19], a pretrained general-purpose question-answering model. In particular, we use Macaw-large (770 million parameters) for our experiments. Macaw is built on top of the text-to-text transformer, T5 [20], a pretrained encoder-decoder model. It is intended to serve as a general question-answering system, and it can produce sensible answers to a variety of questions, even outside its training data. It is also able to generate questions from answers, generate multiple-choice options given an answer and a question, and provide rough explanations for its answers to questions. These features make it relevant to our work, since we want to address the issue of logical inconsistency in an already adept question answering model.

4.1 Data Processing

The BeliefBank dataset [1] consists of constraints and silver facts. Constraints take the form of a graph, relating entities or properties of entities with different relation types (eg: (“IsA, Lion", yes-yes, "hasPart, Fur" translates to “if X is a lion, X has fur"). The data also has silver facts, relating specific entities (eg: “albatross") to nodes in the constraint graphs (eg: “albatross": “isA, Mammal": No, “isA, Bird": Yes). We design question templates according to English grammar rules based on the patterns of nodes, links, and constraints. We process the data into train, validation, and test sets as follows:

1. Convert the silver facts into questions, using standard grammar rules (eg: IsA,tulip $\rightarrow$ IsA,flower becomes “Is a tulip a flower?"). Each question sentence is prepended with '$answer$; $mcoptions$ = (A) Yes (B) No ; $question$ =' which prompts Macaw to provide an answer for the given question as a 'Yes' or 'No' answer. Randomly sample 80% of the question bank into a training set of 10,083 questions. 20% of the facts (10% as validation and 10% as test) are held out so the model is forced to leverage its pre-trained world knowledge on unseen node-entity relations.
2. Traverse the constraint graph to discover every path of $X \rightarrow Y \rightarrow \ldots \rightarrow Z$ or $W \rightarrow X \rightarrow \ldots \rightarrow Y \nrightarrow Z$, adding every edge along such a path to an evaluation set of 5896 questions. We stop at $Y \nrightarrow Z$ edges since we can't make any further implications. We randomly and evenly split the evaluation set into a validation and test set.
3. For each of the 583 nodes starting with “IsA", we make an edge to all other 1846 nodes in the graph, creating a dense set of 1,076,218 questions (with no labels). These will be posed to finetuned models to evaluate the model's logical consistency.

4.2 Finetuning

We generate initial baseline accuracy and consistency scores for on the raw Macaw model based on our custom dataset. We then finetune the model using three different approaches. We test finetuning the model on all 770 million parameters and on freezing all layers except for the last one and only finetune the last layer of the network. However, our training dataset is not large, so the model is susceptible to overfitting. To reduce the risk of overfitting on our small training corpus, we inserted adapter layers [7], which are small layers in between the Macaw block that introduce a learnt perturbation to every intermediate block in the network (see Figure 1). In particular we adopt the adapter structure proposed in [21] where 2 hidden layers are introduced between each Macaw block. We then freeze the original model weights, finetuning only the adapter module weights.

4.3 L1 Sparsity

Modularity in the network necessitates that activations due to any input be sparse. We aim to regularize the activations of our network by promoting sparsity in the activatations. We encourage this sparsity by penalising the L1 norm of the activations of select layers of the network. The L1 norm is commonly used in machine learning as a regularization method to enhance sparsity [22]. In order to induce sparsity, we construct an auxiliary L1 loss that promotes sparse activations through penalizing the L1 norm of the activations.

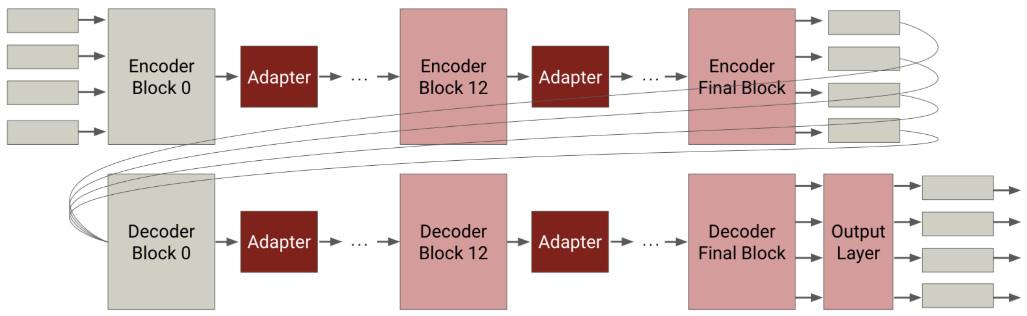


Figure 1: MACAW-large architecture with adapter modules

4.3.1 L1 Loss

The auxiliary loss function promoting L1 sparsity is:

$$L1\ Loss = \sum_{x} \sum_{layer} \sum_{k=1}^{N} \sum_{i=1}^{T} ||A_k(t_{x,i})||_1 \tag{1}$$

Here, $A_k(t_{x,i})$ is the normalized activation of the last layer in the k^{th} block corresponding to the i^{th} token of input x.

4.4 Similarity

While the L1 loss can make the activations sparse, an additional requirement of functional similarity is needed to achieve modularity. Our goal is to enforce that similar beliefs lead to similar sub-parts of the model network being activated and dissimilar beliefs lead to different sub-parts of the network being activated. In order to achieve this goal, we generate similar question pairs based on three different similarity heuristics and construct an auxiliary similarity loss.

4.4.1 Similarity Loss

To encourage the network be modular in its activations, we adopt the Noise Contrastive Estimator (NCE) loss [23], which is of the form:

$$\sum_{(x,x')\in +ve} -\log\left(\frac{\exp(\sum_i \sum_j A_k(t_{x,i})^\top A_k(t_{x',j}))}{\exp(\sum_i \sum_j A_k(t_{x,i})^\top A_k(t_{x',j})) + \sum_{x,x''\in -ve} \exp(1 - \sum_i \sum_j A_k(t_{x,i})^\top A_k(t_{x'',j}))}\right) \tag{2}$$

Here, t_x stands for input x's token, and $A_k(t_x, i)$ is the normalized vector corresponding to activation of the k^{th} layer at token index t_x. The inputs (x, x') are paired together as similar facts (denoted by $+ve$ in the equation) while (x, x'') are paired together as negative facts (denoted by $-ve$ in the equation). The NCE loss thus aims to make the activations of similar facts align with each other while separating the activations of dissimilar facts. The above loss term is added to the training objective.

The resultant loss function for training is thus:

$$Loss = CrossEntropy(y, \hat{y}) + \lambda_1 L1\ Loss + \lambda_2 NCE \tag{3}$$

4.4.2 Question Similarity

It is not immediately obvious what constitutes “similar" or “dissimilar" pairs of facts. Similarity between two sentences can be considered in different ways such as lexical similarity, where the sentences may share common words, or semantic similarity, where the sentences may share common meaning or intent [24]. Sentence similarity can be calculated based on word-to-word, sentence structure, or word vector representation. We leverage the fact that we have a constraint graph relating facts to determine whether two questions are similar. In our approach, we determine similar, or positive pairs, of inputs in three ways: linked-based, adjacency-based, and SimCSE-based. Negative pairs for training were generated by pairing all inputs with all the other inputs, except its positive pair, in the minibatch. From the training dataset, we generate a dataset of positive pairs from each of these three similarity metrics.

Linked-Based Similarity Linked-based similarity is determined by relationships between single-hop facts. If we have questions generated by the constraints $X \rightarrow Y$ and $Y \rightarrow Z$, then we determine that these questions are similar. For example, we have a positive pair "Is a poodle a dog?" and "Do dogs have ears?". Through linked-based similarity, our aim is to capture the similarity between questions that arises from implication relationships. We aim to induce the language model to remain consistent in answering questions given a known set of implications.

Adjacency-Based Similarity In adjacency-based similarity, we determine two questions to be similar if they relate to the same entity. These are questions generated by the constraints $X \rightarrow A$ and $X \rightarrow B$. An example of a positive pair would be "Is a plant a living thing?" and "Is a plant a vertebrate?" The purpose of adjacency-based similarity is to capture similarity that arises from asking questions about the same entity. Our aim is for the model to learn be consistent in answering questions about an entity.

SimCSE-Based Similarity In many circumstances, a ground-truth constraint graph would be unavailable, so we generate similar pairs of questions based on a pretrained SimCSE model [25]. SimCSE learns sentence embeddings through constrastive learning in both an unsupervised manner, where the input sentence predicts itself, and supervised manner, where entailment pairs are positive and contradition pairs are negative. The inputs to the SimCSE model are encoded into embeddings, and these sentence embeddings can be compared for similarity. For every pair of inputs in the training set, we encode the questions into sentence embeddings and compute their cosine similarity score. Pairs with high cosine similarity scores are set to be positive pairs. For example, we have the positive pair: "Is a dog a pet?" and "Does a puppy have paws?" We set the similarity score threshold to be the average cosine similarity of the pairs in the linked-based dataset.

4.4.3 Token-Index-Based Activations

The current formulation of the NCE loss allows for the model to improve on similarity via the pad token activations. To remove the effect of pad token activations, we modify Equation 2 to only consider the dot product of activations of the the 2 inputs at particular indices i and j. [26] investigate and identify where knowledge is encoded in transformer parameters, and the authors find that the last token of an entity has the greatest influence in transformer outputs. Inspired by these findings, the indices i and j were tested amongst: (1) the indices corresponding to the last token of the common entity in a pair of questions (e.g. "Lion" in "Is lion a mammal?" and "Do lions have tails?"), (2) the indices corresponding to the "EOS" token, and (3) the indices corresponding to the last of the special tokens for "$answer$"

5 Experiments

5.1 Data

We use a custom QA dataset (generation details in section 4.1) which is built on constraints and facts given by BeliefBank [1]. The QA dataset contains $10,083$ train questions, $5,896$ validation and test questions, and $1,076,218$ unlabeled questions for constraint checking. For each question, we are given the ground-truth answer. The inputs to the model are natural language questions Q. The output A is a "Yes"/"No" answer. The model M takes in a question $q \in Q$ and predicts an answer A.

5.2 Evaluation method

We quantitatively evaluate the performance of our models on two metrics: its F1 score and consistency.

F1 score is the harmonic mean of the model's precision and recall and is expressed as:

$$F1 = \frac{TP}{TP + 0.5(FP + FN)}$$

Consistency is the fraction of multi-hop beliefs that for which the model correctly believes the single hop fact. For example, if a model believes "a carp is a fish" and "a fish has gills," then the model is consistent if it concludes that "a carp has gills." We implement the following:

1. Predict all possible yes-yes edges in the graph to construct an adjacency matrix of the model's belief graph.
2. Find all the multi-hop paths in the belief graph by computing the powers of the adjacency matrix. [1]. As checking for all lengths of paths theoretically requires computing all powers of the adjacency matrix, as a metric, we restrict ourselves to finding upto 9-hop connections.
3. For every pair of nodes, we can compute the existence of a multi-length hop between them and check if the single hop connection is also believed by the model.

5.3 Experimental details

The input to our model are text questions padded to a length of 64 and output is set to length 8. We enforce the output to be multiple choice options "Yes" or "No." In our experiments, we train using the AdamW optimizer [27] with a fixed learning rate of 3×10^{-4}, betas of (0.9, 0.95), and weight decay of 0.1. We run our experiments on a single NVIDIA Tesla V100 16GB GPU for an average of approximately 3 hours per experiment. We train our model with a batch size of 16 for 10 epochs for all experiments. We conduct the following experiments to establish baseline results: (1) using raw, non-finetuned MACAW-large model, (2) finetuning on all layers of the model and only the last layer of the model, and (3) finetuning with adapters.

L1 Sparsity The value of λ_1 is chosen to balance the L1 sparsity loss with cross-entropy loss. [26] altered the activations of internal neurons to understand the flow of information through a transformer and examined the downstream causal effects of disabling selected multi-layer perceptron layers in transformer language models. The authors argue that altering the activations of middle blocks of a transformer have the greatest influence in a model's output. Building upon the idea by [26], we enforce L1 sparsity on specific layers of the model. We tested enforcing L1 sparsity on the following layers: (1) the middle block of the encoder and/or decoder, (2) all the layers of the encoder and/or decoder, and (3) the final block of the encoder or decoder.

Similarity Loss We conduct experiments on linked-based, adjacency-based, and SimCSE-based similarity datasets. The value of λ_1 and λ_2 are chosen experimentally to balance L1 sparsity loss, similarity loss, and cross-entropy loss. We run L1 sparsity and similarity experiments with adapters. Based on the layers that resulted in the high accuracy and/or consistency scores in the L1 sparsity experiments, we choose the following layers for the similarity loss experiments: (1) middle block of the decoder and (2) final layer of the encoder or decoder. Additionally, we conduct experiments on the token-index-based activations with: (1) common word, (2) "EOS", (3) "\$answer\$" tokens.

5.4 Results

The quantitative results of our experiments on the test set are given in Table 1, which outlines the accuracy and consistency scores of our experiments.

6 Analysis

From our results in table 1, we can immediately conclude:

- Finetuning the adapter model significantly outperforms finetuning the entire model, or finetuning just the last layer, in terms of both accuracy and consistency.

[1]Refer to `https://www.um.edu.mt/library/oar/bitstream/123456789/24439/1/powers%20of%20the%20adjacency%20matrix.pdf` for proof

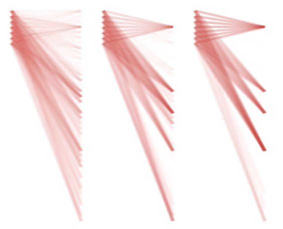
Table 1: Quantitative Results of Experiments

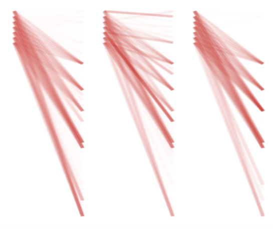
Method	Layer for Aux. Loss	λ_1	λ_2	F1 (%)	Consis. (%)
Raw Model	-	-	-	85.75	52.36
Finetune all parameters	-	0	0	88.89	74.15
Finetune last layer	-	0	0	86.39	48.93
Finetune using Adapters	-	0	0	93.24	86.70
L1 Sparsity	Output Layer	10^{-5}	0	86.50	49.48
L1 Sparsity	Enc. Block 12	10^{-5}	0	91.10	80.45
L1 Sparsity	Enc. Final Block	10^{-5}	0	92.58	86.11
L1 Sparsity	Dec. Block 12	10^{-5}	0	**94.90**	**89.59**
L1 Sparsity	Dec. Final Block	10^{-5}	0	93.91	81.35
L1 Sparsity	Enc. & Dec. Block 12	10^{-5}	0	90.89	74.86
L1 Sparsity	All Enc.+ Dec. Layers	2×10^{-7}	0	87.05	69.68
L1 Sparsity	All Enc. Layers	5×10^{-7}	0	93.94	82.09
Similarity (Linked)	Enc. Final Block	5×10^{-7}	2×10^{-3}	92.29	78.44
Similarity (Linked)	Dec. Block 12	5×10^{-7}	2×10^{-3}	92.88	79.15
Similarity (Linked)	Dec. Final Block	5×10^{-7}	2×10^{-3}	92.34	71.75
Similarity (Adj.)	Enc. Final Block	5×10^{-7}	2×10^{-3}	94.79	**91.00**
Similarity (Adj.)	Dec. Block 12	5×10^{-7}	2×10^{-3}	94.85	82.25
Similarity (Adj.)	Dec. Final Block	5×10^{-7}	2×10^{-3}	95.21	88.85
Similarity (SimCSE)	Enc. Final Block	5×10^{-7}	2×10^{-3}	94.99	90.93
Similarity (SimCSE)	Dec. Block 12	5×10^{-7}	2×10^{-3}	92.06	80.97
Similarity (SimCSE)	Dec. Final Block	5×10^{-7}	2×10^{-3}	**96.81**	89.92
Common Token	Enc. Block 12	5×10^{-7}	2×10^{-3}	95.22	83.15
EOS Token	Enc. Final Block	5×10^{-7}	2×10^{-3}	93.30	83.51

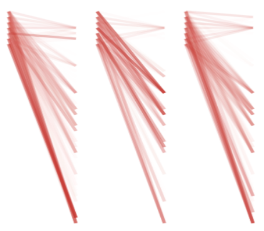
- Enforcing only L1 sparsity led to minor improvements in accuracy and consistency (1.7% in F1 score over the adapter baseline, and 2.88% in consistency). This result suggests sparsity is useful to consistency.
- Surprisingly, enforcing sparsity and similarity on the model using the linked-based similarity dataset suffers the most in consistency, dropping around 10% in consistency. Enforcing sparsity and similarity using the adjacency-based similarity model yields the best consistency at 91.00%, and enforcing using the SimCSE model yields the best F1 score, at 96.81%.

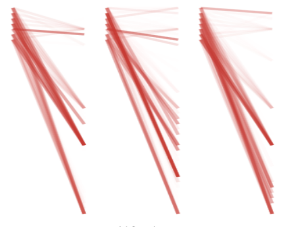
The results provide an indication that sparsity and modularity promote consistency. However, it is not obvious why the adjacency and SimCSE based similarity datasets outperform the linked-based similarity dataset, since one would think that the linked-based dataset is directly providing examples of logical implications to the model. To understand these results better and to investigate whether the model weights were indeed becoming more sparse, we visualized the model's attention weights using BertViz [28] (Figure 2). As plotted from left 2a, to right 2f, we see an increase in the sparsity of the attention weights. The raw model's attention weights are the most diffuse, which is reflected in a poor consistency showing of 52.36%. The model trained on the adjacency-based similarity dataset has the sparsest activations, and also the highest consistency at 91.00%. Moreover, the linked-based similarity dataset, which underperformed in consistency, has more diffuse weights. We thus have an empirical signal linking sparsity with increased consistency.

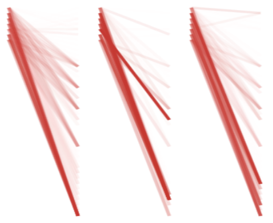
An explanation for why the linked-based similarity dataset is underperforming could be a lack of diversity in its similarity pairs. Whereas the adjacency dataset can include pairs such as "A fish does not have paws" and "A fish cannot preach sermons" (which we will denote a no-no pair), we cannot add no-no or no-yes implications to the linked-based dataset. That is, if we have "Is fish a mammal?" (no), and "Can a mammal have offspring?" (yes), we cannot necessarily conclude yes/no about "Can a fish have offspring?" (which would be a no-yes pair). Another point to make is that the adjacency dataset provides knowledge about an entity (eg: outgoing edges from the common entity, a "fish"), and more intuitively captures information grounded at that entity. While a pair in


(a) Raw Model


(b) Similarity (Link)


(c) Finetuned


(d) L1 sparsity


(e) Similarity (SimCSE)

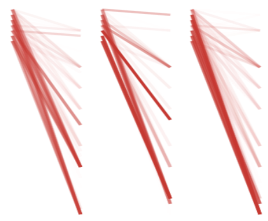

(f) Similarity (Adj.)

Figure 2: Cross-attention weights for applying L1 sparsity and similarity loss on the encoder final block. These images are visualizations for attention heads 1-3 at the final layer when the models are asked: “Is a poodle a dog?” In order from (a) to (f), increasing density of the attention weights correlates approximately to increasing consistency (from 52.36% for the (a) raw model to 91.00% for (f) adjacency-based similarity).

the linked-based dataset does have a common entity, (eg: “A lion is a mammal", and “A mammal is a vertebrate", the common entity is “mammal"), the first fact is an *incoming* edge to mammals, which concerns information more aptly attributed to *lions*, not mammals. Therefore, by forcing the model to recognise a pair of linked edges as similar, the model might find it difficult to note that the combined facts of "A lion is a mammal" and "A mammal is a vertebrate" is giving it longer-range knowledge about lions being vertebrates. On the other hand, the model is learning directly that “fish are not mammals" and that “fish have gills", which helps it encode information about “fish".

Moreover, we note that the final blocks of the encoder and decoder layers performed slightly better than the 12th (i.e. middle) blocks in experiments on the linked, adjacency, and SimCSE datasets. This is interesting, since [26] suggested that mid-layer embeddings have the most influence. This divergence could be explained through the impact of the adapter modules, since their impact on the activations is likelier to be felt near the “ends" of the encoder and decoder blocks.

Finally, we find that token-indexing approaches fail to improve consistency. We theorize this behavior is due a failure to find the optimal token and index position. The selected tokens we tested were likely insufficient and suboptimal indices by which to compute similarity. Additionally, we note that during our experiments, the similarity loss did not converge over our training epochs, and instead decreased very slowly. This is a limitation of the NCE loss as used in SimCLR, since it requires very large batch sizes for the negative pairs to contribute to effective learning. Our batch size of 16 was insufficient to provide a strong enough signal. Even so, we did note improvements in using the adjacent-based and SimCSE similarity datasets, as mentioned previously.

7 Conclusion

We find a strong correlation between the modularity of a model and its consistency. We demonstrate a viable approach for enforcing said modularity by means of an L1 norm penalty and a contrastive similarity loss on the mid-layer model activations to achieve a 4.3% gain in absolute consistency. This project is limited by its scale, both in the size of the dataset (with about 15,000 questions in total) as well as the batch size, which negatively impacts the similarity loss convergence. Directions for future work involve generating and running experiments on a larger, more complex dataset of question, directly enforcing sparsity through methods such as magnitude-based pruning, and exploring the use of momentum encoding (MoCo) to increase the number of negative samples beyond the minibatch size for improved similarity loss training.

References

[1] Nora Kassner, Oyvind Tafjord, Hinrich Schutze, and Peter Clark. Beliefbank: Adding memory to a pre-trained language model for a systematic notion of belief. In *EMNLP*, 2021.

[2] Nora Kassner and Hinrich Schütze. Negated LAMA: birds cannot fly. *CoRR*, abs/1911.03343, 2019.

[3] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China, November 2019. Association for Computational Linguistics.

[4] Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard H. Hovy, Hinrich Schütze, and Yoav Goldberg. Measuring and improving consistency in pretrained language models. *CoRR*, abs/2102.01017, 2021.

[5] Róbert Csordás, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Are neural nets modular? inspecting functional modularity through differentiable weight masks. *CoRR*, abs/2010.02066, 2020.

[6] Oyvind Tafjord and Peter Clark. General-purpose question-answering with Macaw. *ArXiv*, abs/2109.02593, 2021.

[7] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, 2020.

[8] Alon Talmor, Oyvind Tafjord, Peter Clark, Yoav Goldberg, and Jonathan Berant. Leap-of-thought: Teaching pre-trained models to systematically reason over implicit knowledge. *Advances in Neural Information Processing Systems*, 33:20227–20237, 2020.

[9] Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard H. Hovy, Hinrich Schütze, and Yoav Goldberg. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031, 2021.

[10] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[11] Hidenori Ide and Takio Kurita. Improvement of learning for cnn with relu activation by sparse regularization. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2684–2691. IEEE, 2017.

[12] Praveen Kulkarni, Joaquin Zepeda, Frederic Jurie, Patrick Pérez, and Louis Chevallier. Learning the structure of deep architectures using l1 regularization. In *British Machine Vision Conference, 2015*, 2015.

[13] Stephen Merity, Bryan McCann, and Richard Socher. Revisiting activation regularization for language rnns. *arXiv preprint arXiv:1708.01009*, 2017.

[14] Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron C. Courville. Systematic generalization: What is required and can it be learned? *CoRR*, abs/1811.12889, 2018.

[15] Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. In *International Conference on Learning Representations*, 2021.

[16] Chihiro Watanabe. Interpreting layered neural networks via hierarchical modular representation. *ArXiv*, abs/1810.01588, 2019.

[17] Shlomi Hod, Stephen Casper, Daniel Filan, Cody Wild, Andrew Critch, and Stuart Russell. Detecting modularity in deep neural networks. *arXiv preprint arXiv:2110.08058*, 2021.

[18] Daniel Filan, Stephen Casper, Shlomi Hod, Cody Wild, Andrew Critch, and Stuart Russell. Clusterability in neural networks. *CoRR*, abs/2103.03386, 2021.

[19] Oyvind Tafjord and Peter Clark. General-purpose question-answering with macaw. *arXiv preprint arXiv:2109.02593*, 2021.

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

[21] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. *ArXiv*, abs/2005.00247, 2021.

[22] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted 1 minimization. *Journal of Fourier analysis and applications*, 14(5):877–905, 2008.

[23] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020.

[24] Mamdouh Farouk. Measuring sentences similarity: a survey. *arXiv preprint arXiv:1910.03940*, 2019.

[25] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *CoRR*, abs/2104.08821, 2021.

[26] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual knowledge in gpt. *arXiv preprint arXiv:2202.05262*, 2022.

[27] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. 2018.

[28] Jesse Vig. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy, July 2019. Association for Computational Linguistics.