Analyzing Commonsense Emergence in Few-shot Knowledge Models

Jeff Da[1] JEFFD@ALLENAI.ORG
Ronan Le Bras[1] RONANLB@ALLENAI.ORG
Ximing Lu[1] XIMINGLU@ALLENAI.ORG
Yejin Choi[1,3] YEJINC@ALLENAI.ORG
Antoine Bosselut[1,2] ANTOINE.BOSSELUT@EPFL.CH
[1] *Allen Institute for AI*
[2] *EPFL*
[3] *University of Washington*

Abstract

Recently, commonsense knowledge models — pretrained language models (LM) fine-tuned on knowledge graph (KG) tuples — showed that considerable amounts of commonsense knowledge can be encoded in the parameters of large language models [Bosselut et al., 2019]. However, as parallel studies show that LMs are poor hypothesizers of declarative commonsense relationships [Petroni et al., 2019] on their own, it remains unclear whether this knowledge is learned during pretraining or from fine-tuning on KG examples.

To investigate this question, we train commonsense knowledge models in few-shot settings to study the emergence of their commonsense representation abilities. Our results show that commonsense knowledge models can rapidly adapt from limited examples, indicating that KG fine-tuning serves to learn an interface to encoded knowledge learned during pretraining. Importantly, our analysis of absolute, angular, and distributional parameter changes during few-shot fine-tuning provides novel insights into *how* this interface is learned.

1. Introduction

Driven by advances in pretrained language models [Liu et al., 2019, Raffel et al., 2019], recent NLP systems have demonstrated considerable improvement [Clark et al., 2020, Lourie et al., 2021] on benchmarks that test the commonsense representation and reasoning abilities of NLP systems [Talmor et al., 2019, Sap et al., 2019b, Bhagavatula et al., 2020, Sakaguchi et al., 2020]. Despite this success, it remains unclear how much commonsense knowledge is actually directly encoded by these models. When prompted out-of-the-box to complete commonsense declarative relationships (as depicted in Figure 1), they exhibit limited ability to map their language modeling abilities to this task [Petroni et al., 2019, Zhou et al., 2020].

However, the zero-shot expression of declarative commonsense knowledge is but one use of language that these models are pretrained to manifest. Infilling evaluations that entangle this ability with general language expression may be narrow tests of true commonsense representation ability. Indeed, commonsense knowledge models [Bosselut et al., 2019] — which finetune pretrained LMs on examples from commonsense knowledge graphs [Speer et al., 2017, Sap et al., 2019a, Jiang et al., 2021] — learn to express declarative commonsense relationships much more effectively. This learning procedure allows these models to recalibrate to the actual task of hypothesizing declarative knowledge. Simultaneously, these systems are

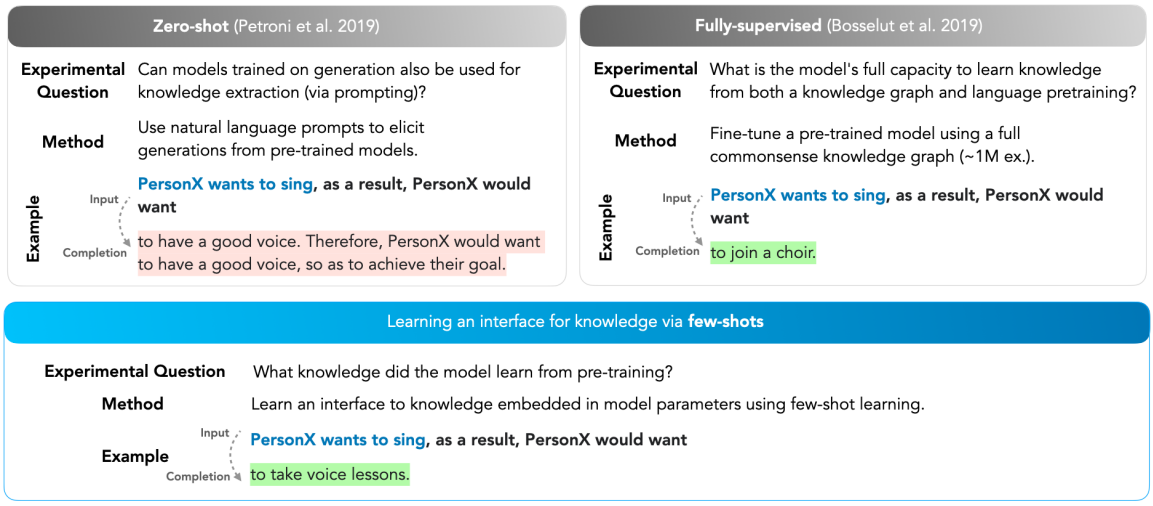

Figure 1: Commonsense knowledge models can be trained to effectively hypothesize commonsense knowledge when trained in few-shots, implying that finetuning serves to *learn* and interface to knowledge encoded during pretraining.

evaluated in challenging train/test settings (where head entities tested on cannot be seen during training), indicating they may learn to transfer implicit representations of declarative knowledge that are learned during pretraining [Hwang et al., 2021].

However, current commonsense knowledge models are trained on a large number of examples from knowledge graphs. Consequently, this comprehensive finetuning obfuscates whether these systems exhibit knowledge that was encoded during language pretraining [Geva et al., 2020, Dai et al., 2021], or whether pretraining merely provides a favorable initialization to learn to generalize from examples in the KG. In this work, we explore this question by training commonsense knowledge models in a few-shot setting. We vary the training budgets of KG examples that they receive, and evaluate their capacity to hypothesize commonsense knowledge after training on each of these budgets. Using this framework, we discover whether the finetuning process causes the model to learn new knowledge from the KG, or whether the model learns an *interface* to existing knowledge it already encodes.

Our results demonstrate that few-shot commonsense knowledge models trained on knowledge graphs in few-shots — up to 10,000× fewer tuples than the full KG — exhibit strong performance relative to zero-shot LMs, and approach the performance of fully-trained knowledge models. Our analysis supports the hypothesis that few-shot finetuning allows the model to adapt knowledge it already encodes implicitly. We find that few-shot training mainly changes parameters in the attention heads of the decoder transformer blocks. The large feed-forward networks that serve as an implicit memory storage for information learned during pretraining [Geva et al., 2020] are minimally updated. Furthermore, we observe that larger knowledge models exhibit most of their parameter change across a more concentrated set of parameters, implying they learn knowledge during pretraining in a less entangled manner due to their capacity. Finally, we find that using natural language prompts to represent relation inputs accelerates commonsense emergence. Our code, few-shot training sets, and models are at https://github.com/allenai/few-shot-comet/.

Head h	Relation r	Generated Tail t (COMeT)	Generated Tail t (GPT-3)
nail	AtLocation	in wall ✓	hammer ✗
video camera	ObjectUse	video recording ✓	record the grooming session ✓
PersonX takes it to the vet	HinderedBy	PersonX doesn't have time ✓	PersonX doesn't know where the vet is. ✓
PersonX gets a call for an interview	xAttr	PersonX is qualified for the job ✓	lucky ✓
PersonX wants to learn how to swim	xAttr	PersonX isn't confident in the water ✓	a good idea ✗
PersonX falls ill	xEffect	PersonX will miss work ✓	is in bed ✓
PersonX sprays by a skunk	xEffect	PersonX will be sick ✓	is stinky ✓
PersonX misses class	xNeed	to have a valid excuse ✓	to have a class ✓
PersonX notices a strange smell	xWant	to investigate the smell ✓	PersonX is distracted by the smell of PersonY's perfume. ✗
PersonX wants to learn how to sing	xWant	to take voice lessons ✓	to learn how to play an instrument ✗

Table 1: Examples of few-shot ($n = 3$) generations produced by COMeT (T5) and GPT-3. COMeT (T5) produces diverse and novel tail hypotheses despite learning from few examples and is able to learn the task interface more easily than GPT-3, which tends to copy the input or generate unnecessary full sentences. Assessments (✓, ✗) are from human evaluators.

2. Background

In this work, we investigate few-shot performance and parameter change of commonsense knowledge models to shed light on the dynamics of commonsense representation learning. Below, we describe background concepts to contextualize this analysis.

Commonsense Knowledge Graphs are structured, relational representations of commonsense knowledge. In our study, we use ATOMIC$^{20}_{20}$ [Hwang et al., 2021],[1] a commonsense KG with 1.33M inferential knowledge tuples about entities and events. It represents a large-scale commonsense repository of textual descriptions that encode social and physical aspects of common human experiences. Across its 1.33M tuples, it captures information about 23 relationship types. Example head entities and relations are shown in Table 1. ATOMIC$^{20}_{20}$ is split into training, development, and test subsets such that no head entities in one set appear in any other. This property allows models trained on this resource to be evaluated on their capacity to generalize commonsense relationships to new entities, events, and situations.

Commonsense Knowledge Models represent facts by learning to encode a commonsense KG [Bosselut et al., 2019]. They are pre-trained as language models and fine-tuned on knowledge graph tuples to learn to hypothesize knowledge relationships through language generation. After training on a large collection of tuples from a KG, they learn the structure and relationships encoded by that KG. Furthermore, because they are seeded with pretrained language models, they learn to generalize the relationships to other entities about which the LM implicitly encodes knowledge [Petroni et al., 2019, Roberts et al., 2020]. Consequently, they can be used to produce precise knowledge on-demand for any entity that can be expressed through language.

3. Experimental Setup

In this section, we outline our experimental setting for training few-shot knowledge models.

Input Each tuple in the ATOMIC$^{20}_{20}$ knowledge graph is composed of a {head h, relation r, tail t} triplet. The head and tail entities in the triplet are natural language words or phrases.

1. Downloadable at https://allenai.org/data/atomic-2020

Relation r	Prompt
ObjectUse	h is used for t
AtLocation	You are likely to find a h in a t
xIntent	Because of h, PersonX wanted t
xWant	After h, PersonX would want t.
xAttr	h is seen as t
isAfter	Something that happens after h is t
oWant	As a result of h, others would want t

Table 2: Examples of language prompts used to represent relations for knowledge models. Prompts significantly speed up transfer in few-shot learning settings.

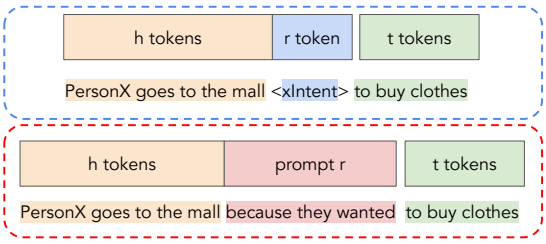

Figure 2: With prompting, KG relations are mapped to natural language templates that more closely resemble inputs the model has learned to process from pretraining.

Commonsense knowledge models are trained by providing the tokens of h and r as inputs to the model and learning to generate the tokens of t. In this work, rather than initializing r with a new token and random embedding [Bosselut et al., 2019], we automatically format the input tuples into natural language prompts to represent the relation [Feldman et al., 2019, Jiang et al., 2020]. Table 2 shows examples of such prompts.

Training We source training tuples from the Atomic$^{20}_{20}$ training set [Hwang et al., 2021]. When constructing few-shot training sets, we set a target number n of examples to randomly sample from the knowledge graph for each relation (*i.e.*, $n = 3 \implies 3$ examples $\times$ 23 relations $= 69$ total training examples). Once a training set of examples is produced, the model is trained on this subset to minimize the negative log-likelihood of the tokens of the tail entity for each tuple. We use the AdaFactor optimizer [Shazeer and Stern, 2018] with a constant learning rate of 0.001, a mini-batch size of 4, and train the model for 3 epochs. Unless stated otherwise (§5.3), we use T5-11B [Raffel et al., 2019] as a seed language model for all experiments. To pick hyperparameters during fine-tuning, we run 5 different sets of hyperparameters and use a validation set of equivalent size to the training set (n examples / relation) to evaluate performance [Perez et al., 2021].

Evaluation We evaluate the knowledge hypothesized by the trained few-shot models using human and automatic evaluations. For the human evaluation (Accept % in Table 3), we use the procedure described in [Hwang et al., 2021]. We ask annotators to label the plausibility generated tuples using a 4-point Likert scale: {*always/often true* (+2), *sometimes/likely true* (+1), *false/untrue* (−1), *nonsensical* (−2)}. We collect 3 annotations per relation, convert each annotation to an acceptability label (*i.e.*, $\{+1, +2\} \rightarrow$ ✓, $\{-1, -2\} \rightarrow$ ✗) and use the majority label as the acceptability judgment. Evaluation agreement is measured using Fleiss's $\kappa = 0.49$ for the acceptability judgment. We also evaluate hypothesized knowledge tuples using automatic metrics: BLEU-1 [Papineni et al., 2002], METEOR [Banerjee and Lavie, 2005], ROUGE-L [Lin, 2004], and CIDEr [Vedantam et al., 2015]. For few-shot experiments, we report average performance across 5 runs with different training sets.

4. Do few-shot knowledge models learn?

We evaluate the general few-shot commonsense interface learning capability of large-scale language models. We train COMeT models as described in §3 and set the number of training

Methodology	Model	BLEU-1	METEOR	ROUGE-L	CIDEr	Accept %
zero-shot	GPT-2 XL	10.1	8.2	9.8	4.7	36.6
	GPT-3	14.3	11.7	13.9	4.7	39.9
few-shot ($n = 3$)	GPT-2 XL (augmentation)	15.1	10.2	13.5	6.5	37.5
	COMET (GPT-2 XL) (finetuning)	18.4	11.1	14.2	7.3	40.1
	GPT-3 (augmentation)	29.9	17.8	25.3	19.0	71.7
	COMET (T5) (finetuning)	24.2	14.4	21.3	18.1	75.7
fully supervised	COMET (GPT-2 XL)	40.7	29.2	48.5	65.3	72.5
	COMET (BART)	46.9	33.0	49.5	65.8	84.5
	COMET (T5)	**48.2**	**34.1**	**50.0**	**66.4**	**86.4**

Table 3: Comparison between various trained knowledge models. Few-shot ($n = 3$) knowledge models transfer well in both the finetuning (COMET) and augmentation (GPT-3) settings.

examples per relation to $n = 3$. We report several zero-shot and few-shot augmentation baselines (*e.g.*, GPT-{2 XL, 3}), which prepend n training examples (for the same relation) to the start of the input sequence as a prompt, and condition the generation on these additional examples. Few-shot augmentation baselines are not fine-tuned, but receive examples using the same prompt formats as the finetuning baselines (Tables 2, 8). Similar to fine-tuning, we report average performance across 5 different example sets used for prompting. As T5 models are not pretrained using a causal language modeling objective, reporting zero-shot evaluation performance for a multi-word generation task would be an unfair comparison, so we only report limited numbers in Table 6 of Appendix A. We also report the results of several fully supervised COMET knowledge models.

Findings We find that both the few-shot augmentation and few-shot learning settings are able to produce high quality commonsense knowledge tuples, indicating that large LMs are also efficient few-shot learners (in addition to showcasing impressive few-shot prompting abilities). Using only $n = 3$ tuples per relation, both GPT-3 and COMET (T5) produce high-quality tuples that are accepted as *plausible* by human evaluators more than 70% of the time — 75.7% for COMET (T5). While this performance falls short of the fully supervised knowledge model performance — 86.4% — it demonstrates that the commonsense representation abilities of language models extends much further than their zero-shot prompting abilities. Furthermore, these abilities can be measured by finetuning on limited examples, supporting the hypothesis that early-stage finetuning serves to learn an interface to knowledge learned during pretraining. In the following section, we support this hypothesis by investigating how parameters of large language models change during finetuning (across different training budgets n §5.2, different model size §5.3, and different prompting strategies §5.4).

5. How do knowledge models learn?

Despite strong empirical results showing that pretrained language models can rapidly adapt to become few-shot language learners [Schick and Schutze, 2021a], little work has focused on *how* language models learn to adapt knowledge learned from pretraining to perform well on downstream end tasks. We explore this problem in the context of few-shot knowledge modeling and investigate how the parameters of few-shot commonsense knowledge models

change during finetuning. We define three measures to investigate parameter change: absolute parameter change, angular parameter change, and distribution of parameter change.

5.1 Measuring parameter change

Notation To more easily discuss measures of parameter change during knowledge model fine-tuning, we review the structure of the transformer blocks in an encoder-decoder transformer LM [Vaswani et al., 2017]. Encoder blocks consist of a self-attention layer followed by a feedforward network. Each self-attention layer contains parameter matrices (referred to in the following sections as q, k, v) that project the query, key, and value inputs before computing a scaled dot-product attention between the key and value, and an output projection o of the attended value vectors. The trainable parameters in the feedforward network consist of two linear projection matrices wi, wo. Each decoder block consists of the same parameter matrices with an additional cross-attention layer (parameter matrices xq, key xk, value xv, and output xo) that attends to the outputs of the encoder blocks.

Absolute Parameter Change To measure a normalized quantity for average change in each parameter matrix, we compute a normalized ℓ_1 distance for each set of parameter matrices in the transformer blocks. Without loss of generality, for each of these parameter matrices, we define Θ^{PT} as the matrix of parameters before finetuning, and Θ^{FT} as the matrix of parameters post-finetuning. The normalized ℓ_1 distance between these two matrices for each parameter matrix type is then:

$$d_{\ell_1} = \frac{1}{mn}\left\|\Theta^{FT} - \Theta^{PT}\right\|_1, \tag{1}$$

where m and n correspond to the row and column dimensionality of the parameter matrix, and $\|\cdot\|_1$ corresponds to the ℓ_1 norm. Intuitively, d_{ℓ_1} measures the degree of absolute per parameter shift between these two sets of parameters.

Angular Parameter Change However, measuring normalized ℓ_1 distance solely captures how the magnitude of the parameter matrix changes, but obfuscates relative drift between parameters in each matrix. For instance, a matrix with high average L1 change may in fact have small relative changes between its elements (i.e., the parameter change may mainly cause a scaling effect on the output vectors). Consequently, we measure the row-wise angular distance between Θ^{PT} and Θ^{FT} to capture the rotational drift of parameters during finetuning:

$$d_{ang} = \frac{1}{m\pi}\sum_{k=1}^{m}\cos^{-1}\left(\frac{\langle\Theta_k^{FT}, \Theta_k^{PT}\rangle}{\left\|\Theta_k^{FT}\right\|_2\left\|\Theta_k^{PT}\right\|_2}\right), \tag{2}$$

where k indexes a common row vector in both parameter matrices, $\langle\cdot\rangle$ corresponds to an inner product, $\|\cdot\|_2$ is the Euclidean (ℓ_2) norm, and $\cos^{-1}$ is the inverse cosine operation. Contrary to d_{ℓ_1}, which represents a per-parameter average distance, d_{ang} is computed between all corresponding rows of the parameter matrices, Θ^{PT} and Θ^{FT}, and averaged per row. We choose to average rows as opposed to columns as each row in the parameter matrix performs a dot product with the input vector to the transformation, and has a correspondence to an individual element in the output vector.

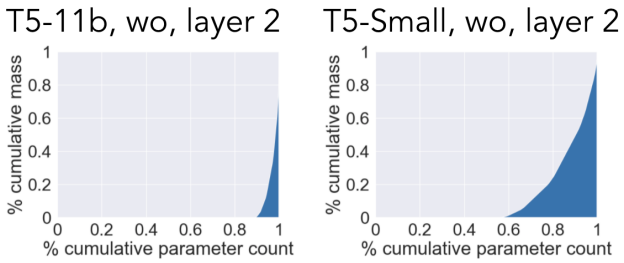

Figure 3: Example visualization of area under curve for FFN parameter matrices, $n = 30$

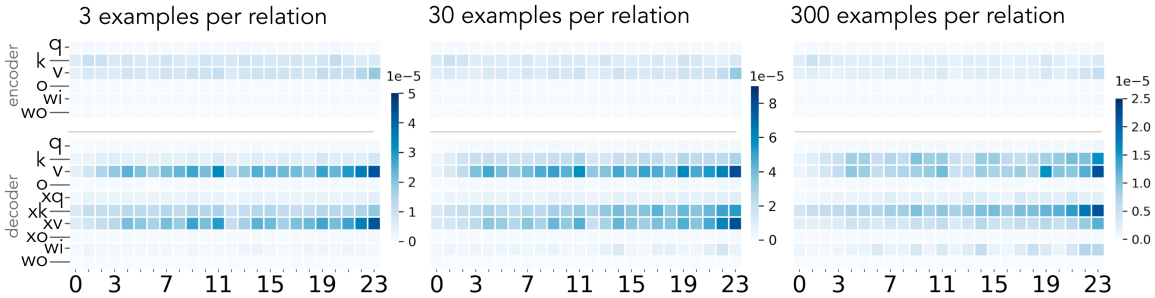

Figure 4: Normalized ℓ_1 distance between Θ^{FT} and Θ^{PT} for each parameter matrix at each layer of the encoder and decoder of the COMET (T5-Large) model. We report the change for training budgets of $n = 3$, 30, and 300.

Distributional Parameter Change In additional to angular distance, we also report the distribution of parameter changes for a given matrix. More concretely, parameter changes may be concentrated across varying numbers of dimensions. We compute the change in each dimension by calculating $W = |\Theta^{PT} - \Theta^{FT}|$, sorting parameters by their absolute change, and plotting % cumulative parameter number against % cumulative mass as a function of a given parameter change $w_i \in W$:

$$\left(\frac{\texttt{count}(w \in W, w \leq w_i)}{\texttt{count}(w \in W)}, \frac{\sum_{w \in W, w \leq w_i} w}{\sum_{w \in W} w} \right) \tag{3}$$

Figure 3 visualizes this relationship. Intuitively, a matrix with a small AUC has a skewed relative distribution of changes, signaling that a select number of dimensions are responsible for a large amount of the learning during fine-tuning. Likewise, a matrix with a large AUC learns a relatively even parameter change distribution.

5.2 How does few-shot learning change the parameters of the pretrained LM?

We first investigate how different example budgets affect parameter change. We train a few-shot COMET (T5) model across different example budgets ($n \in \{3, 30, 300\}$). Table 4 shows that performance increases monotonically as we grow the example budget. To explore the differences between models, we investigate the parameter changes between each fine-tuned model in Fig. 4. We include additional heatmaps for angular change in the Appendix.

Findings In Figure 4, we observe that the ℓ_1 distance between Θ^{FT} and Θ^{PT} increases as we increase the training budget in the decoder. Interestingly, we note that most parameter changes occur in the decoder of the LM (and more strongly in its later layers), rather than the encoder, implying that much of the parameter shift may be due to learning how to *express* commonsense knowledge in a declarative form. Furthermore, when comparing the changes across parameter matrix types, the key and value matrices of the attention layers change more while the feedforward network parameters (*wi*, *wo*) remain relatively unchanged. These findings suggest the model learns how to process the structure of declarative commonsense knowledge expressions, but does not need to modify its stored representations of this knowledge learned during pretraining, hinting at a possible explanation for the success of adapters for limited-parameter finetuning of NLP models [Houlsby et al., 2019].

# **Ex**	COMET **Model**	**BLEU-1**	**METEOR**	**ROUGE-L**	**CIDEr**
3	T5-Small	13.4 ± 3.0	8.8 ± 3.1	11.5 ± 1.1	6.8 ± 1.0
	T5-Large	23.5 ± 1.3	13.6 ± 1.1	19.1 ± 1.2	12.6 ± 2.1
	T5-11B	**24.2** ± 1.7	**14.4** ± 1.9	**21.3** ± 3.7	**18.1** ± 7.9
30	T5-Small	26.2 ± 2.0	15.5 ± 0.4	20.8 ± 0.6	11.9 ± 0.3
	T5-Large	29.6 ± 0.6	16.6 ± 0.4	23.0 ± 0.5	14.3 ± 0.6
	T5-11B	**31.9** ± 0.3	**18.7** ± 0.5	**26.0** ± 0.6	**19.8** ± 1.2
300	T5-Small	37.0 ± 1.0	23.5 ± 0.4	35.9 ± 0.9	36.8 ± 2.2
	T5-Large	**39.1** ± 0.8	26.3 ± 0.4	**40.4** ± 0.9	46.6 ± 2.0
	T5-11B	**39.1** ± 0.9	**26.7** ± 0.9	40.1 ± 1.5	**48.5** ± 2.8

Table 4: Effect of model size for few-shot commmonsense knowledge modeling performance.

5.3 How does model size affect knowledge model learning?

We train a few-shot COMET (T5) model (with $n = 30$) across different pretrained language model sizes, i.e. Small, Large, and 11B, with 60M, 770M, and 11B parameters, respectively.

Findings In Table 4, we observe that the importance of model size decreases as the training budget increases, reinforcing that model scale is critically important in few-shot learning settings [Brown et al., 2020]. When examining the parameter change heatmaps in Figure 5, we note the ℓ_1 distances between finetuned and original parameters increase as model size increases. As these distances are normalized by the number of parameters, this observation implies that when trained on the same examples, the smaller model experiences smaller magnitude gradient updates over time. This pattern indicates more destructive inference between parameter updates in the smaller models, an observation supported by the larger angular parameter changes in the encoders of the smaller models.

Interestingly, several parameter matrices display high angular change but low ℓ_1 change (e.g., in early *wo* layers of T5-11B), suggesting that a matrix with low average absolute parameter change may still contain critical dimensions which are used to learn the task. To investigate these differences, we look at the AUC heatmaps (Fig. 5, *bottom row*), and note that as the feedforward networks get larger with increasing model size, the relative number of parameters that change to encode new information drops drastically in both the decoder and encoder. Furthermore, the lower AUCs for the encoder parameter changes indicate that the encoder layers generally experience more concentrated changes as the model size increases.

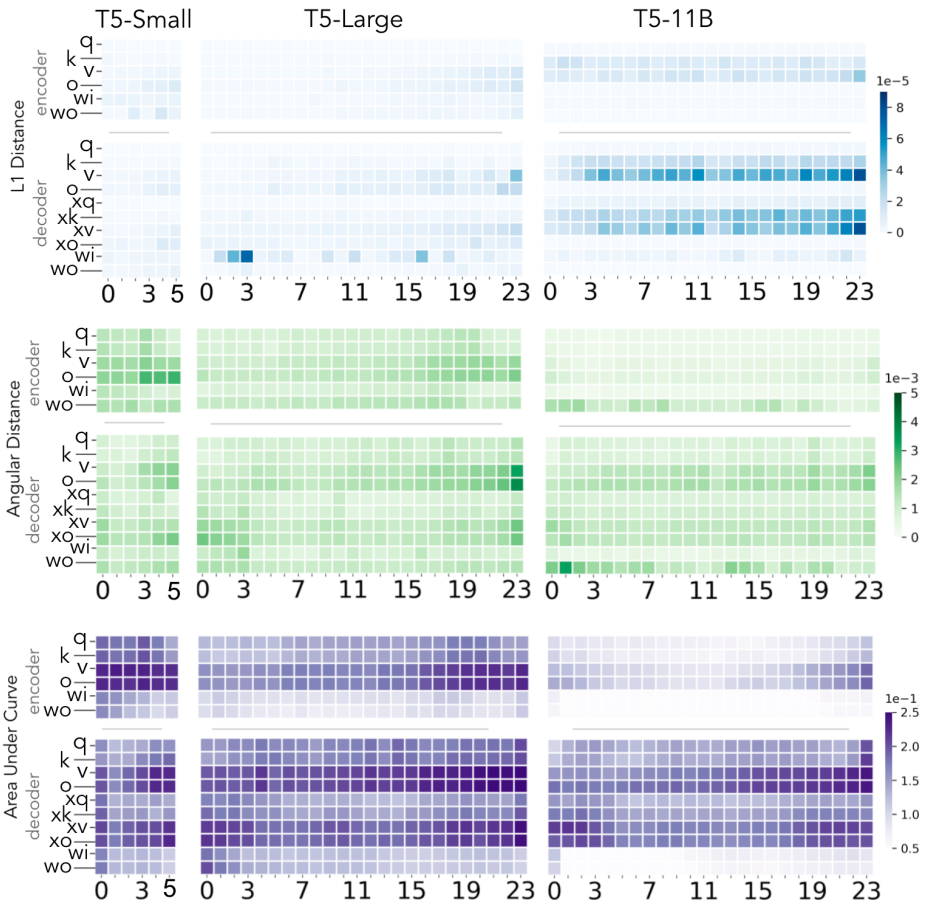

Figure 5: Parameter change measures for different knowledge model sizes.

These two patterns suggest that the increased capacity of larger language models allow them to encode knowledge in a less distributed manner, making their knowledge easier to access.

5.4 How do prompts influence knowledge model learning?

Recent work has shown that prompts can help models elicit knowledge from pretrained language representations [Feldman et al., 2019, Shin et al., 2020]. However, eliciting knowledge through zero-shot prompting has drawbacks. For example, the output is sensitive to subtle variations in the construction of the language prompt [Jiang et al., 2020]. Here, we explore whether prompts can accelerate few-shot learning by initializing natural language prompts for each relation (Table 2) training on these expressions of relations rather than initializing new relation token embeddings for each relation as in Bosselut et al. [2019]. As a control for whether the model understands the semantics of specific relation prompts, or merely benefits from arbitrary expressions of relations using language, we also train a model where we shuffle prompts among the relations (*i.e.* shuffling the **Prompt** column of Table 2).

Findings In Table 5, we see that knowledge models can efficiently learn from fewer examples when relations are represented using correct natural language prompts (∼10×

# Ex	Input	BLEU-1	METEOR	ROUGE-L	CIDEr
3	Prompts	**24.2** ± 1.7	**14.4** ± 1.9	**21.3** ± 3.7	**18.1** ± 7.9
	Embedding	13.9 ± 1.3	11.4 ± 1.2	13.5 ± 0.9	7.4 ± 0.8
30	Prompts	**31.9** ± 0.3	**18.7** ± 0.5	**26.0** ± 0.6	**19.8** ± 1.2
	Embedding	18.1 ± 0.8	13.5 ± 1.1	16.7 ± 1.1	9.7 ± 1.7
	Shuffled	15.6 ± 0.8	11.3 ± 0.5	14.2 ± 0.5	8.3 ± 0.8

Table 5: Prompts accelerate few-shot commonsense interface learning.

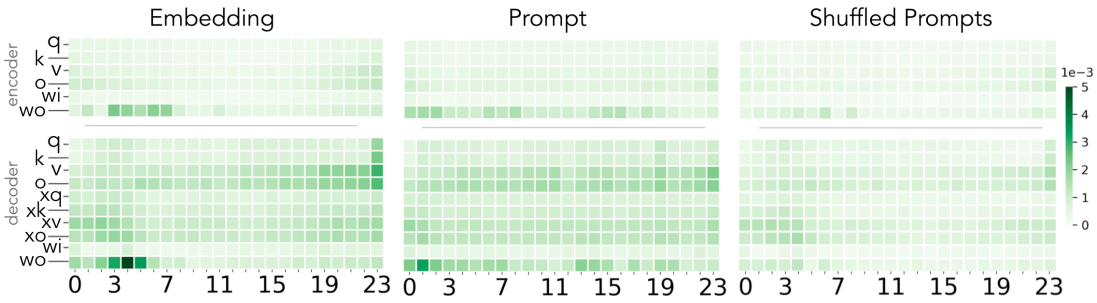

Figure 6: Average angular distance between original and finetuned parameters for prompts, shuffled prompts, and initialized relation embeddings. Each model is trained with $n = 30$.

fewer examples), an observation made by contemporaneous works on other tasks [Scao and Rush, 2021]. Interestingly, however, we find that the absolute and angular distance heatmaps for prompt and embedding relation inputs are similar (Fig. 6; ℓ_1 heatmap in Appendix), implying that most of the parameter change during fine-tuning is not linked to the way the relation is represented. One notable difference is that models with embedding relation inputs have a significant angular change in layer 4 of the decoder *wo* matrix. Models that use prompts see a larger angular change in layer 2 of the decoder *wo* matrix. This discrepancy suggests that both models must learn to adapt their parameters to the new input format, but do so by adapting the same parameters at different layers. We note that there is no such concentrated angular change for the **Shuffled** setting, perhaps because the model must first *unlearn* language relationships between head entities and prompts, requiring more uniform parameter updates.

6. Related Work

Commonsense Knowledge Models Our work uses commonsense knowledge models, first proposed by Bosselut et al. [2019], to learn from commonsense knowledge graphs. Hwang et al. [2021] also trained commonsense models on ATOMIC$^{20}_{20}$, but focused on fully-supervised learning. Other works have developed commonsense knowledge models that are grounded to visual scenes [Park et al., 2020, Da et al., 2021], requiring multimodal commonsense inference generation. Recent works extend commonsense knowledge models beyond generating single-hop inferences and generate multi-branch [Bosselut et al., 2021] and multi-hop [Wang et al., 2020b] inferential structures. Commonsense knowledge base completion is also a closely related task to commonsense inference generation [Li et al., 2016, Saito et al., 2018]. Recent

works on this task combine language and graph structure representations for improved generalization [Malaviya et al., 2020, Wang et al., 2020a]. Our work differs from these prior studies by investigating commonsense knowledge models in few-shot settings, and analyzing the emergence of their capabilities as a function of parameter change.

Few-shot Learning with Prompting In recent years, the term few-shot has taken two meanings: the classical definition of training on limited examples [Fei-Fei et al., 2006, Fink, 2005], and a new *in-context* definition where models are given examples as a prepended augmentation of their context and process these examples as input to recognize the structure of a task [Brown et al., 2020]. Various contemporaneous works related to ours have studied the use of natural language and artificial prompts in language models [Schick and Schutze, 2021a, Gao et al., 2020, Tam et al., 2021, Schick and Schutze, 2021b] as accelerators of few-shot learning. Scao and Rush [2021] further empirically study the data efficiency advantage gained from using prompting to represent a task. Our work differs from these studies in that we do not use prompts to map a classification task to masked language modeling, but instead use prompts as priming text for a generation task. Most similar to our approach is perhaps Schick and Schütze [2020], which uses prompts for few-shot summarization, but their work does not focus on how few-shot training affects the learned representations of their model.

7. Conclusion

In this work, we propose few-shot commonsense knowledge modeling, where models are finetuned on a limited number of examples to evaluate how efficiently they can adapt their pretrained representations to the task of hypothesizing commonsense knowledge tuples. Our results demonstrate that large language models require few examples to adapt their learned representations to the task and our analysis explores how their parameters change to enable this rapid emergence of commonsense representation ability.

Acknowledgements

The authors would like to thank the anonymous reviewers for their feedback, and the Amazon Mechanical Turk community for help with annotation. The authors thank Vered Shwartz and Chris Manning for feedback on early drafts. We also gratefully acknowledge the support of DARPA under No. N660011924033 (MCS), JD.com, and the Allen Institute for AI. TPU machines for conducting experiments were provided by Google.

References

S. Banerjee and A. Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *IEEvaluation@ACL*, 2005.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, S. Yih, and Yejin Choi. Abductive commonsense reasoning. In *ICLR*, 2020.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, A. Çelikyilmaz, and Yejin Choi. Comet: Commonsense transformers for automatic knowledge graph construction. In *ACL*, 2019.

Antoine Bosselut, Ronan Le Bras, and Yejin Choi. Dynamic neuro-symbolic knowledge graph construction for zero-shot commonsense question answering. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

T. Brown, B. Mann, Nick Ryder, Melanie Subbiah, J. Kaplan, P. Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, G. Krüger, Tom Henighan, R. Child, Aditya Ramesh, D. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, E. Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, J. Clark, Christopher Berner, Sam McCandlish, A. Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. In *IJCAI*, 2020.

Jeff Da, M. Forbes, Rowan Zellers, Anthony Zheng, Jena D. Hwang, Antoine Bosselut, and Yejin Choi. Edited media understanding: Reasoning about implications of manipulated images. In *ACL*, 2021.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, and Furu Wei. Knowledge neurons in pretrained transformers. *ArXiv*, abs/2104.08696, 2021.

Li Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:594–611, 2006.

J. Feldman, Joe Davison, and Alexander M. Rush. Commonsense knowledge mining from pretrained models. In *EMNLP*, 2019.

Michael Fink. Object classification from a single example utilizing class relevance metrics. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005. URL https://proceedings.neurips.cc/paper/2004/file/ef1e491a766ce3127556063d49bc2f98-Paper.pdf.

Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. *ArXiv*, abs/2012.15723, 2020.

Mor Geva, R. Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *ArXiv*, abs/2012.14913, 2020.

N. Houlsby, A. Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and S. Gelly. Parameter-efficient transfer learning for nlp. In *ICML*, 2019.

Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. Comet-atomic 2020: On symbolic and neural commonsense knowledge graphs. *AAAI*, 2021.

Liwei Jiang, Antoine Bosselut, Chandra Bhagavatula, and Yejin Choi. "i'm not mad": Commonsense implications of negation and contradiction. *ArXiv*, abs/2104.06511, 2021.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *Transactions of the Association for Computational Linguistics (TACL)*, July 2020. URL https://arxiv.org/abs/1911.12543.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. Commonsense knowledge base completion. In *ACL*, volume 1, pages 1445–1455, 2016.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *ACL*, 2004.

Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

Nicholas Lourie, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Unicorn on rainbow: A universal commonsense reasoning model on a new multitask benchmark. In *AAAI*, 2021.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. Commonsense knowledge base completion with structural and semantic context. In *AAAI*, 2020.

Kishore Papineni, S. Roukos, T. Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002.

Jae Sung Park, Chandra Bhagavatula, R. Mottaghi, Ali Farhadi, and Yejin Choi. Visualcomet: Reasoning about the dynamic context of a still image. In *ECCV*, 2020.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models. *ArXiv*, abs/2105.11447, 2021.

F. Petroni, Tim Rocktäschel, Patrick Lewis, A. Bakhtin, Y. Wu, Alexander H. Miller, and S. Riedel. Language models as knowledge bases? *ArXiv*, abs/1909.01066, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, W. Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, abs/1910.10683, 2019.

Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.437. URL https://www.aclweb.org/anthology/2020.emnlp-main.437.

Itsumi Saito, Kyosuke Nishida, Hisako Asano, and Junji Tomita. Commonsense knowledge base completion and generation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 141–150, 2018.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *AAAI*, 2020.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. Atomic: An atlas of machine commonsense for if-then reasoning. In *AAAI*, 2019a.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. 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 4463–4473, Hong Kong, China, November 2019b. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. URL https://www.aclweb.org/anthology/D19-1454.

Teven Le Scao and Alexander M. Rush. How many data points is a prompt worth? In *NAACL*, 2021.

Timo Schick and H. Schutze. It's not just size that matters: Small language models are also few-shot learners. In *NAACL*, 2021a.

Timo Schick and H. Schutze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *EACL*, 2021b.

Timo Schick and Hinrich Schütze. Few-shot text generation with pattern-exploiting training. *ArXiv*, abs/2012.11926, 2020.

Noam M. Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. *ArXiv*, abs/1804.04235, 2018.

Taylor Shin, Yasaman Razeghi, IV RobertLLogan, Eric Wallace, and S. Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *EMNLP*, 2020.

Robyn Speer, J. Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. *ArXiv*, abs/1612.03975, 2017.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://www.aclweb.org/anthology/N19-1421.

Derek Tam, R. R. Menon, M. Bansal, Shashank Srivastava, and Colin Raffel. Improving and simplifying pattern exploiting training. *ArXiv*, abs/2103.11955, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, L. Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Ramakrishna Vedantam, C. L. Zitnick, and D. Parikh. Cider: Consensus-based image description evaluation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4566–4575, 2015.

Bin Wang, Guangtao Wang, J. Huang, Jiaxuan You, J. Leskovec, and C. J. Kuo. Inductive learning on commonsense knowledge graph completion. *ArXiv*, abs/2009.09263, 2020a.

Peifeng Wang, Nanyun Peng, Filip Ilievski, Pedro Szekely, and Xiang Ren. Connecting the dots: A knowledgeable path generator for commonsense question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4129–4140, Online, November 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.369. URL https://www.aclweb.org/anthology/2020.findings-emnlp.369.

Xuhui Zhou, Yue Zhang, Leyang Cui, and Dandan Huang. Evaluating commonsense in pre-trained language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9733–9740, Apr. 2020. doi: 10.1609/aaai.v34i05.6523. URL https://ojs.aaai.org/index.php/AAAI/article/view/6523.

Appendix A. Accuracy in zero-shot MLM setting

While little work has explored few-shot knowledge completion, recent works have investigated performance of zero-shot knowledge graphs [Petroni et al., 2019, Feldman et al., 2019]. Thus, we investigate the ability of T5-11B to complete commonsense knowledge in a zero-shot setting.

Model	BLEU-1	METEOR	ROUGE-L	CIDEr
T5 - Zero-shot	6.7	7.8	7.3	7.3
T5 - Few-shot ($n = 3$)	31.9	18.7	26.0	19.8
T5 - Fully-supervised	48.2	34.1	50.0	66.4

Table 6: Zero-shot performance of T5-11B

We use prompts to leverage the masking objective of the language model pretraining. Since the mask only predicts several tokens at a time, for relations with longer length tail entities, we allow the model to predict up to 7 mask tokens in succession, or until the model predicts an empty string for the mask. We suggest that this is still only a workaround, and masked models are poor predictors of longer length tail entities, as indicated by our results above.

Appendix B. Additional Results on the Effect of Relation Input Format

In Table 7, we provide further experimental results on the effect of relation input format on few-shot performance. Namely, we extend the results from Table 5 with the case $n = 300$. These results confirm that knowledge models can efficiently learn from fewer examples when relations are represented using natural language prompts. We also show the results from using paraphrases of the main prompts for training (original and paraphrased prompts can be found in Tables 8 and 10, respectively). We find that the paraphrased prompts do cause a slight drop in performance, but that this drop is generally close to the margin of error, indicating that while prompt formulation is an important consideration [Jiang et al., 2020], fine-tuning on the prompts does make the model less sensitive to prompt variations.

# Ex	Input	BLEU-1	METEOR	ROUGE-L	CIDEr
3	Prompts	**24.2** ± 1.7	**14.4** ± 1.9	**21.3** ± 3.7	**18.1** ± 7.9
	Embedding	13.9 ± 1.3	11.4 ± 1.2	13.5 ± 0.9	7.4 ± 0.8
30	Prompts	**31.9** ± 0.3	**18.7** ± 0.5	**26.0** ± 0.6	**19.8** ± 1.2
	Embedding	18.1 ± 0.8	13.5 ± 1.1	16.7 ± 1.1	9.7 ± 1.7
	Shuffled Prompts	15.6 ± 0.8	11.3 ± 0.5	14.2 ± 0.5	8.3 ± 0.8
	Paraphrased	30.5 ± 1.3	18.3 ± 0.5	24.5 ± 0.8	18.4 ± 1.4
300	Prompts	**39.1** ± 0.9	**26.7** ± 0.9	**40.1** ± 1.5	**48.5** ± 2.8
	Embedding	25.2 ± 0.8	18.5 ± 0.9	26.4 ± 1.9	26.8 ± 4.3

Table 7: Effect of relation input format on few-shot performance. Prompts accelerate few-shot commonsense interface learning. We show mean performance over 5 random splits of training examples, and standard deviation (±) between splits.

Relation	Template
ObjectUse	{} is used for
AtLocation	You are likely to find {} in
MadeUpOf	{} is made up of
HasProperty	{} is
CapableOf	{} can
Desires	{} wants
NotDesires	{} does not want
isAfter	Something that happens after {} is
HasSubEvent	Something you might do while {} is
isBefore	Something that happens before {} is
HinderedBy	{} is hindered by
Causes	Sometimes {} causes
xReason	{}. The reason for PersonX doing this is
isFilledBy	{} can be filled by
xNeed	But before {}, PersonX needed
xAttr	{} is seen as
xEffect	As a result of {}, PersonX will
xReact	As a result of {}, PersonX feels
xWant	After {}, PersonX would want
xIntent	Because of {}, PersonX wanted
oEffect	as a result of {}, others will
oReact	as a result of {}, others would feel
oWant	as a result of {}, others would want

Table 8: Prompts used for relations in ATOMIC2020.

Something you might do while {} is ___
Something you might do while design software is determine deliverables
Something you might do while scuba dive is take off scuba gear
Something you might do while play ball is put on mitt

Table 9: Example of augmentation experiments, following [Brown et al., 2020]. {} indicates the location of the head, and the language model is asked to complete the tail by finishing the sentence.

Relation	**Template**
ObjectUse	a {} can be used for
AtLocation	You could find {} in the location
MadeUpOf	{} is made up of
HasProperty	{} will have
CapableOf	{} is capable of
Desires	a {} desires
NotDesires	a {} does not desire
isAfter	Before {},
HasSubEvent	You might do {} while doing
isBefore	After {},
HinderedBy	{}. This is hindered by
Causes	Sometimes {} causes
xReason	{}. PersonX did this because
isFilledBy	{} is filled
xNeed	Before {}, PersonX needs to
xAttr	{}. An attribute of PersonX is
xEffect	The effect of {} PersonX will be
xReact	As a result of {}. PersonX will be
xWant	After {}, PersonX will want to
xIntent	For {}, PersonX did this to
oEffect	An effect of {} on others will be
oReact	As a result of {}, other feel
oWant	After {}, others will want to

Table 10: Paraphrased version of the prompts used (for paraphrased experiment in the Appendix.)

Appendix C. Additional Experiments on Parameter Change Measures

In addition, we provide extensive results of the ℓ_1 and angular distances, as well as the distributional parameter change metric (AUC), for both the encoder and decoder of various model sizes (Small, Large, and 11B) and different example budget ($n \in \{3, 30, 300\}$) (Figures 7-36). When computing AUC diagrams, we round each weight change to the nearest 10^{-5}.

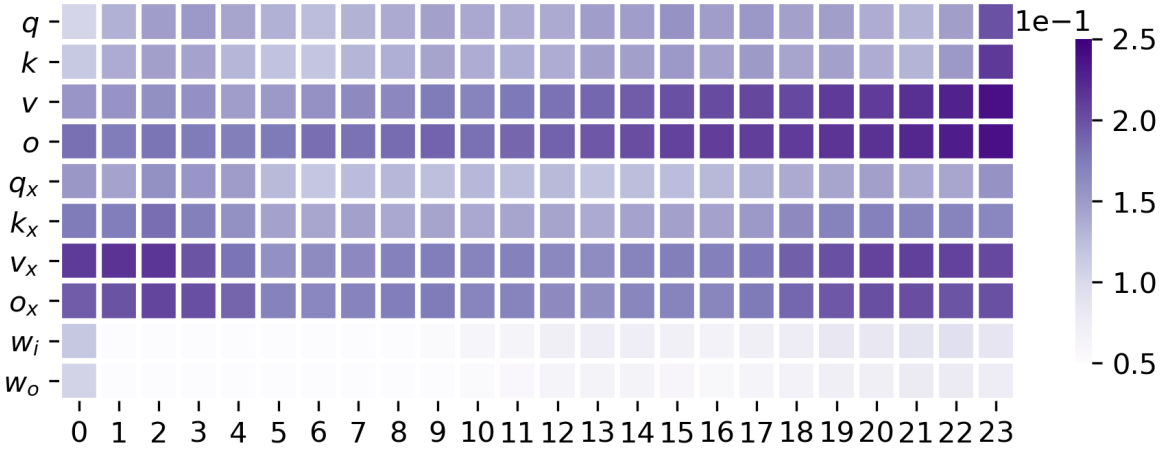

Figure 7: Area Under Curve, Decoder, T5-11B, $n = 30$

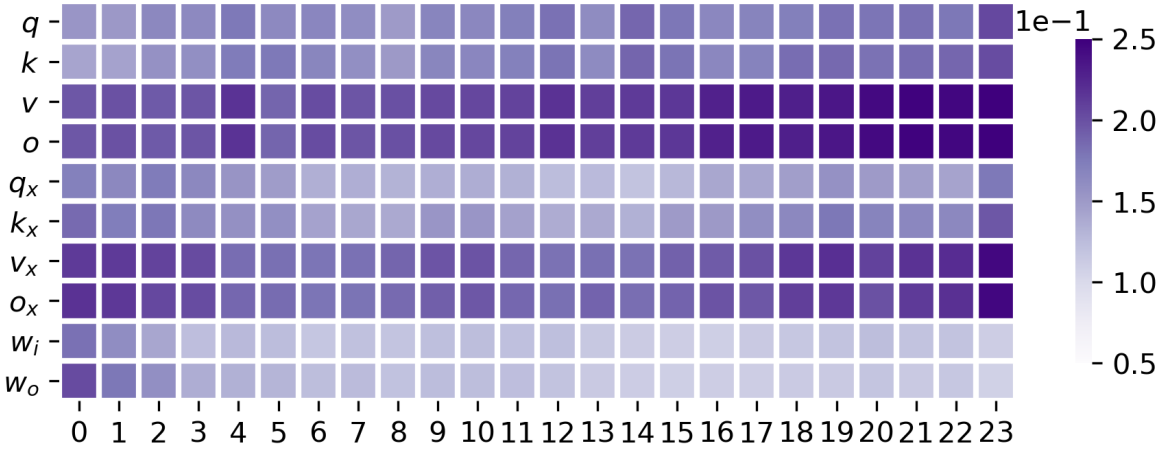

Figure 8: Area Under Curve, Decoder, T5-Large, $n = 30$

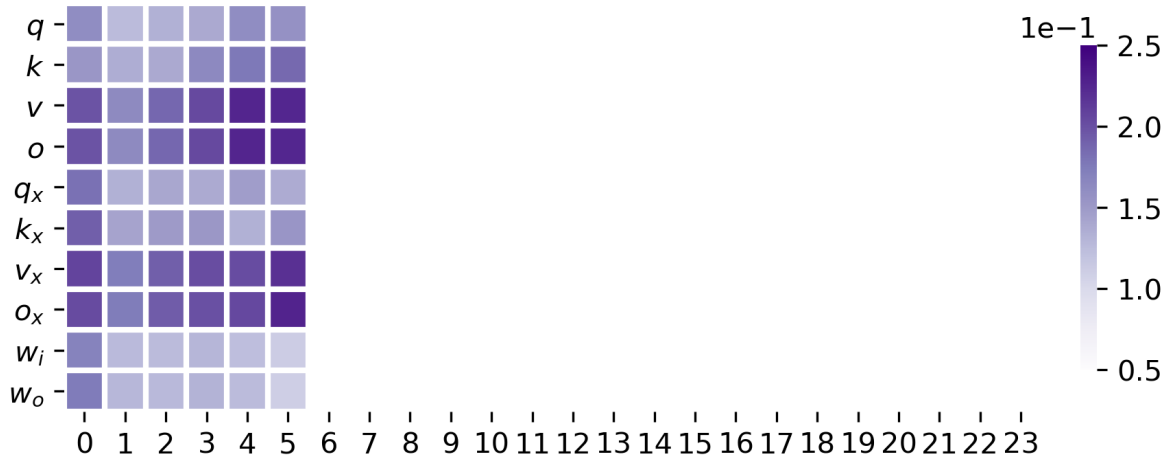

Figure 9: Area Under Curve, Decoder, T5-Small, $n = 30$

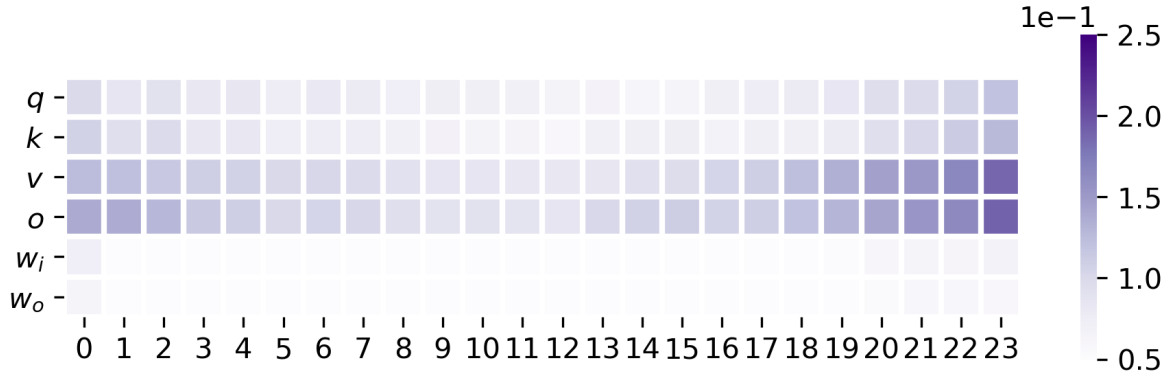

Figure 10: Area Under Curve, Encoder, T5-11B, $n = 30$

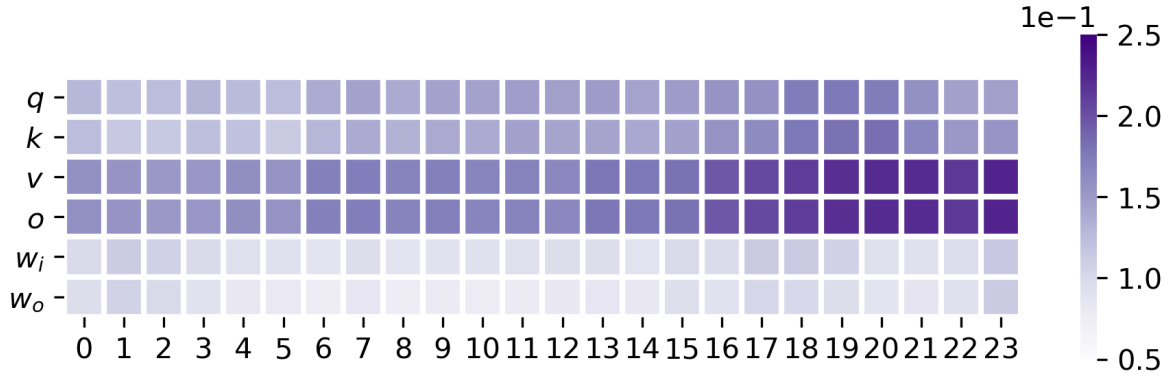

Figure 11: Area Under Curve, Encoder, T5-Large, $n = 30$

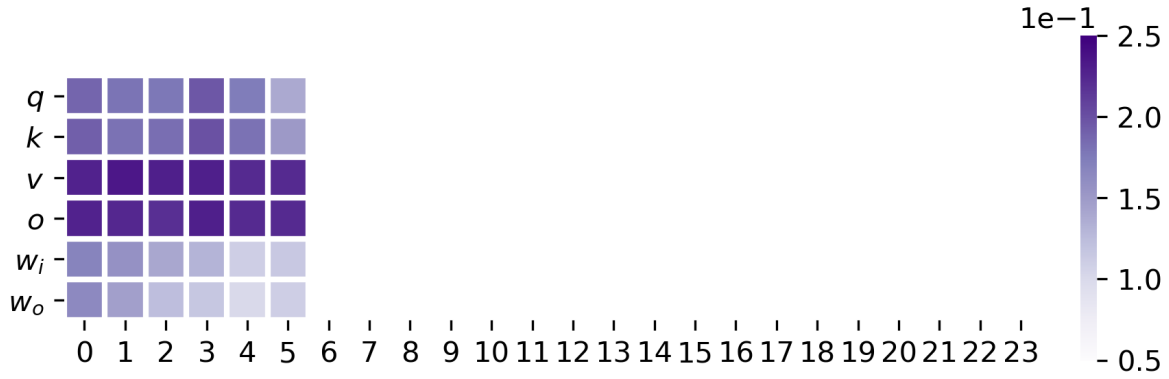

Figure 12: Area Under Curve, Encoder, T5-Small, $n = 30$

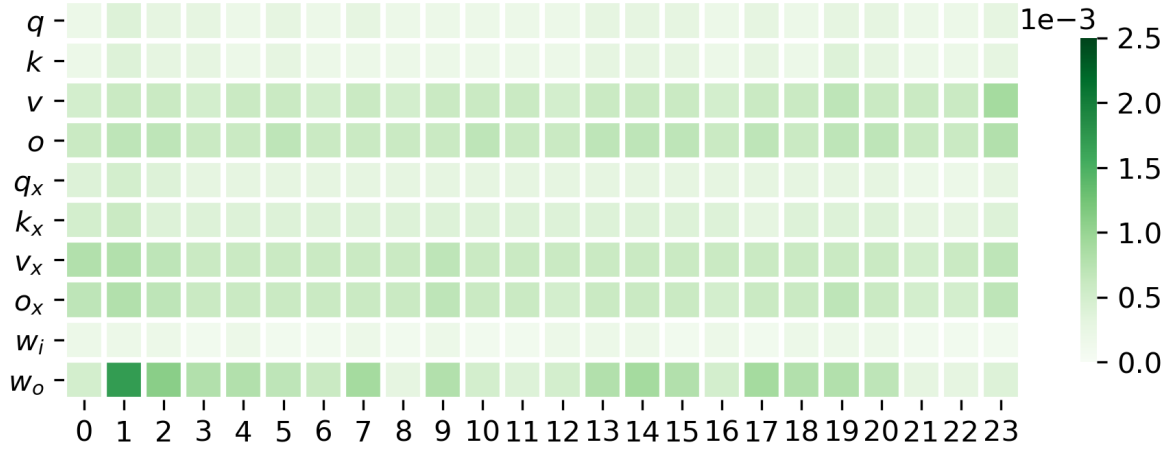

Figure 13: Angular change, Decoder, T5-11B, $n = 3$

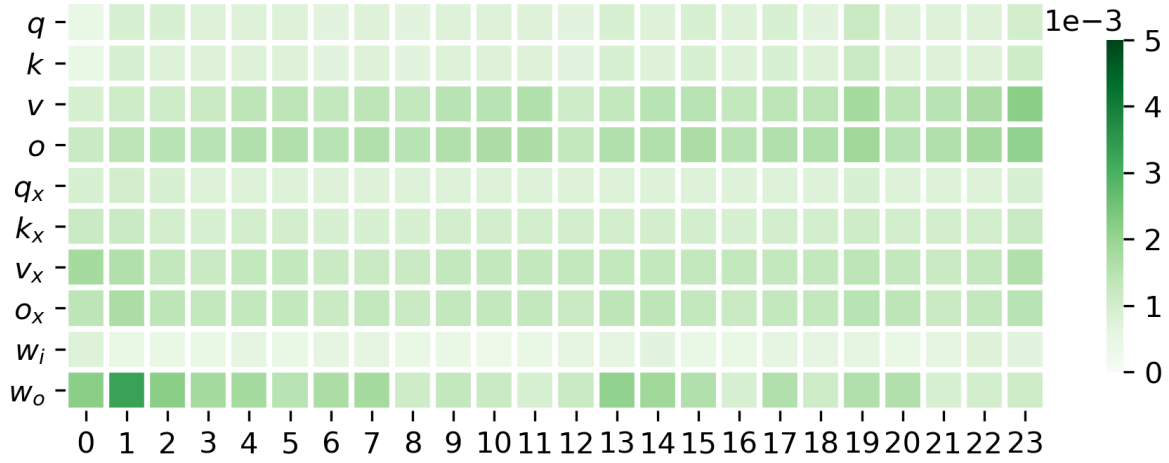

Figure 14: Angular change, Decoder, T5-11B, $n = 30$

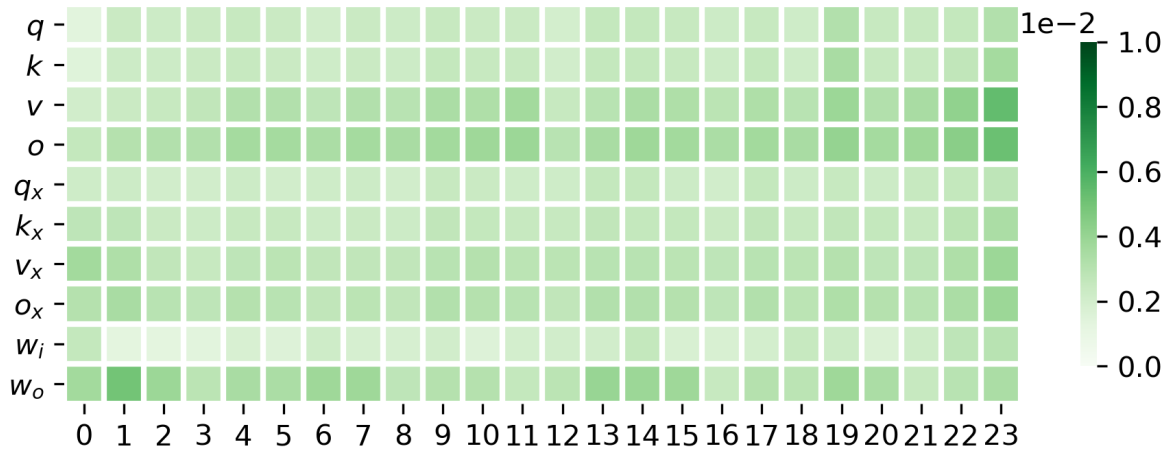

Figure 15: Angular change, Decoder, T5-11B, $n = 300$

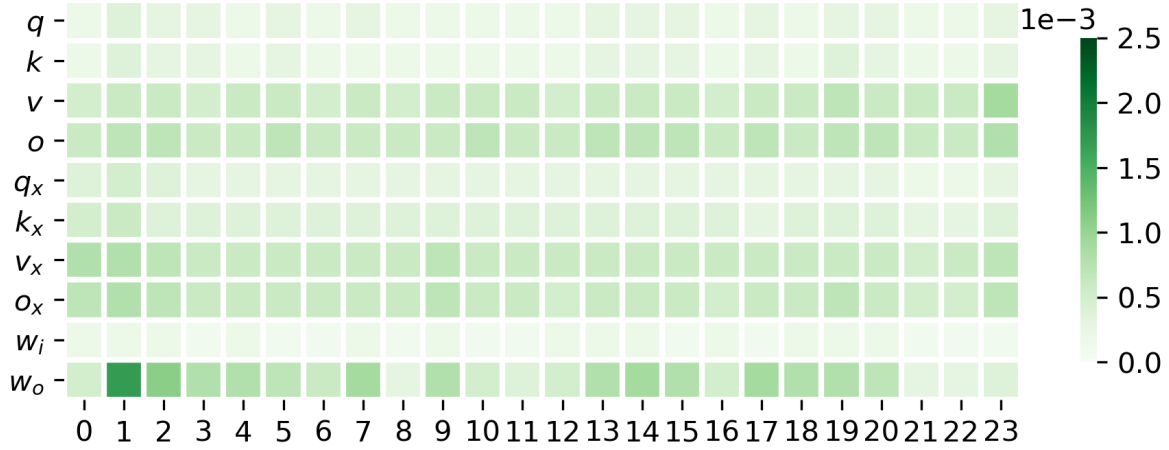

Figure 16: Angular change, Encoder, T5-11B, $n = 3$

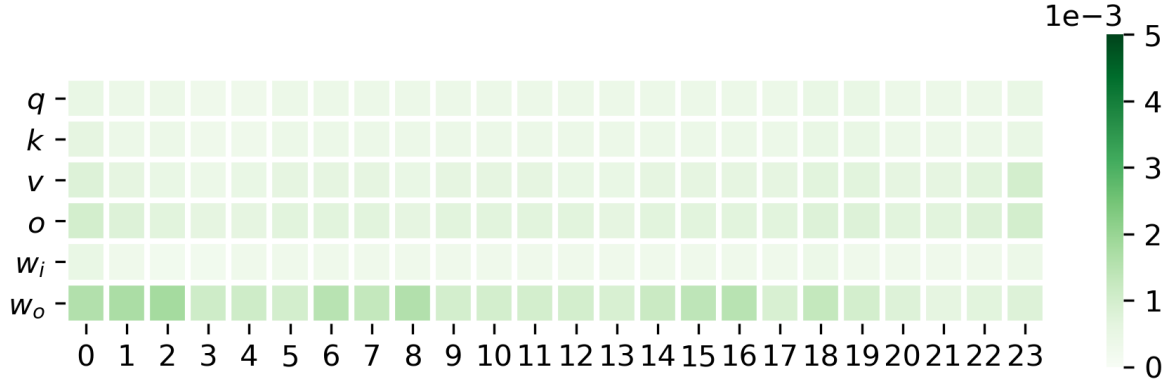

Figure 17: Angular change, Encoder, T5-11B, $n = 30$

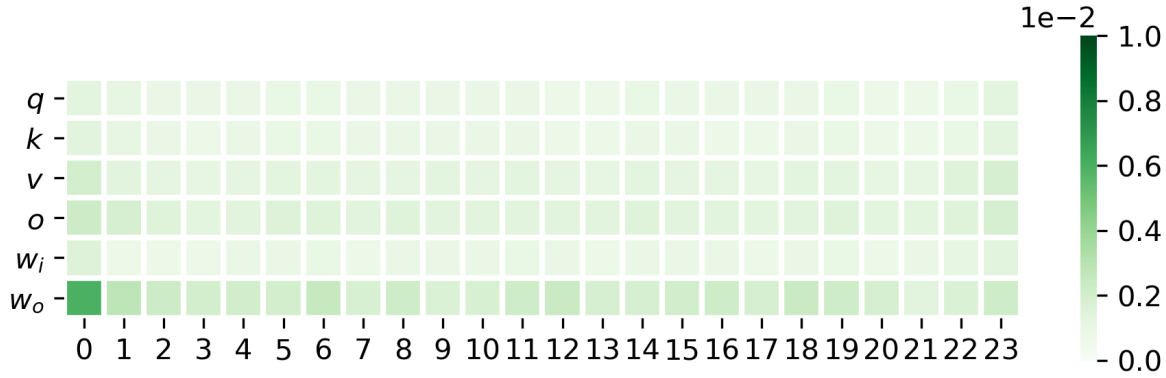

Figure 18: Angular change, Encoder, T5-11B, $n = 300$

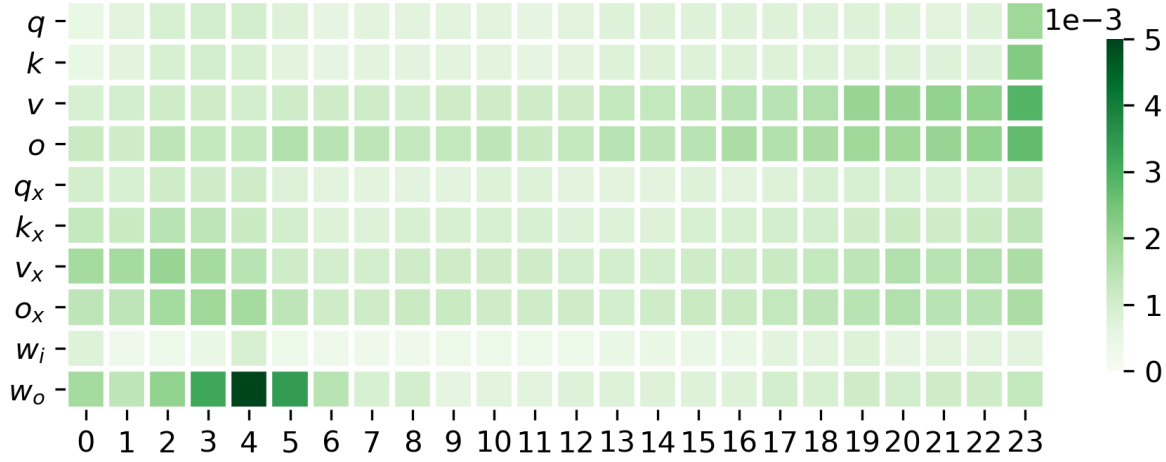

Figure 19: Angular change, Decoder, T5-11B (relation embedding), $n = 30$

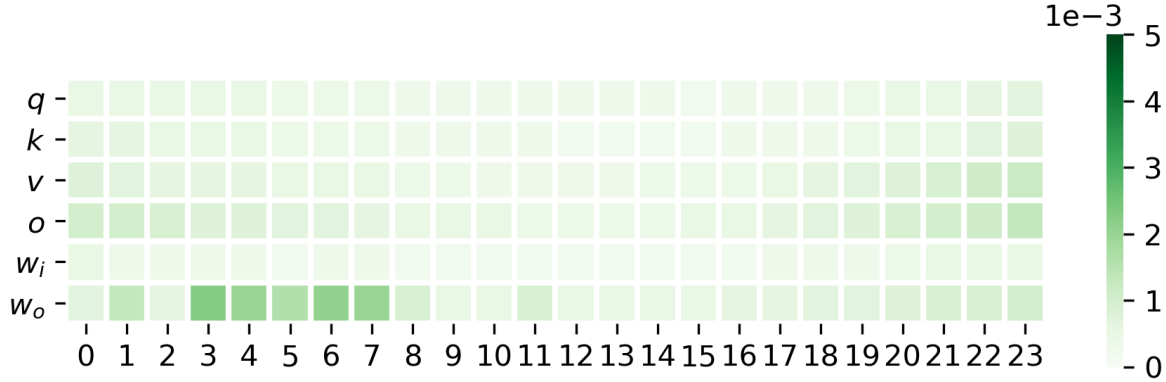

Figure 20: Angular change, Encoder, T5-11B (relation embedding), $n = 30$

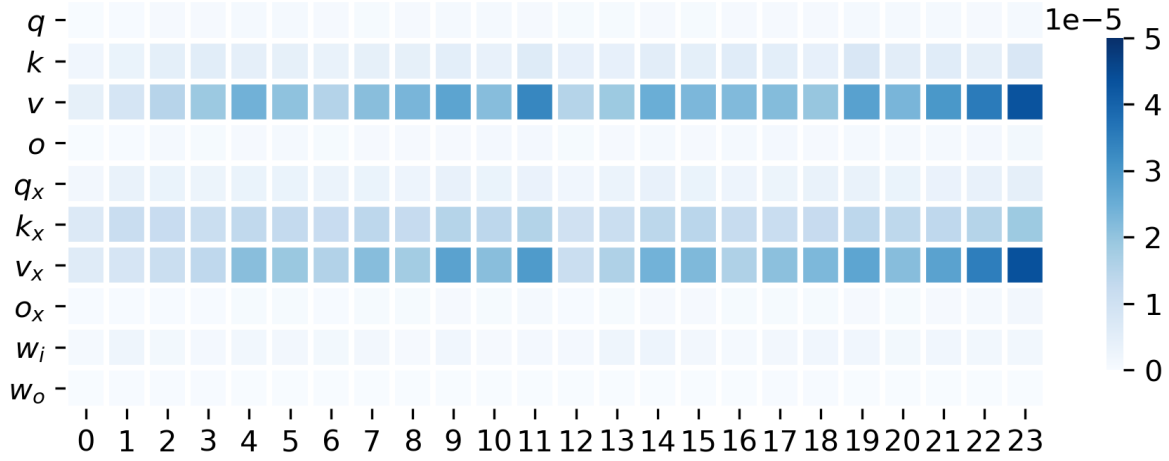

Figure 21: L1 change, Decoder, T5-11B, $n = 3$

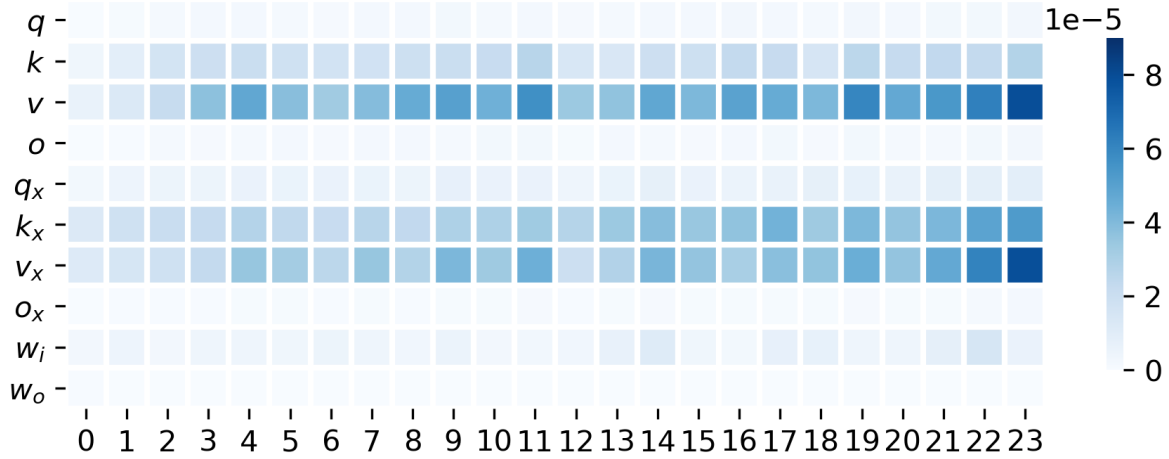

Figure 22: L1 change, Decoder, T5-11B, $n = 30$

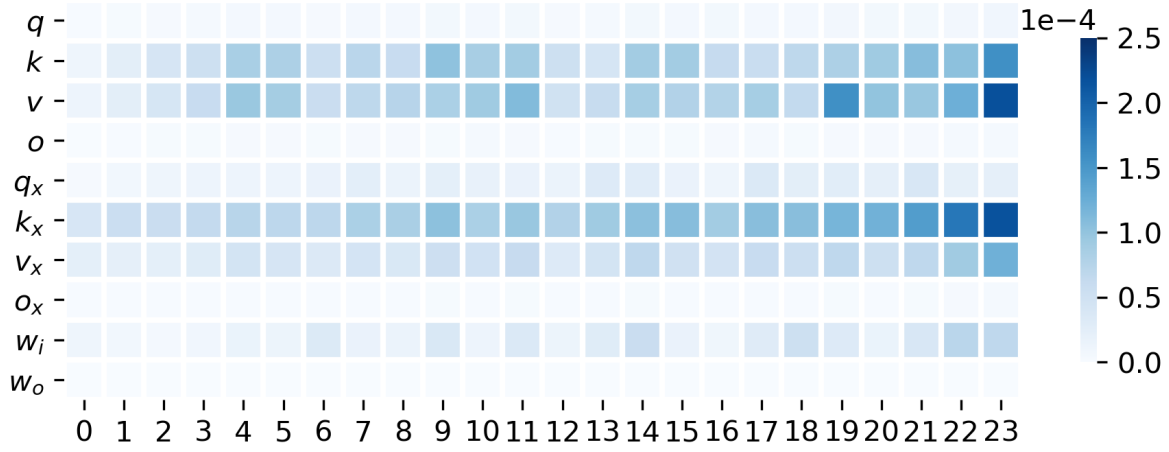

Figure 23: L1 change, Decoder, T5-11B, $n = 300$

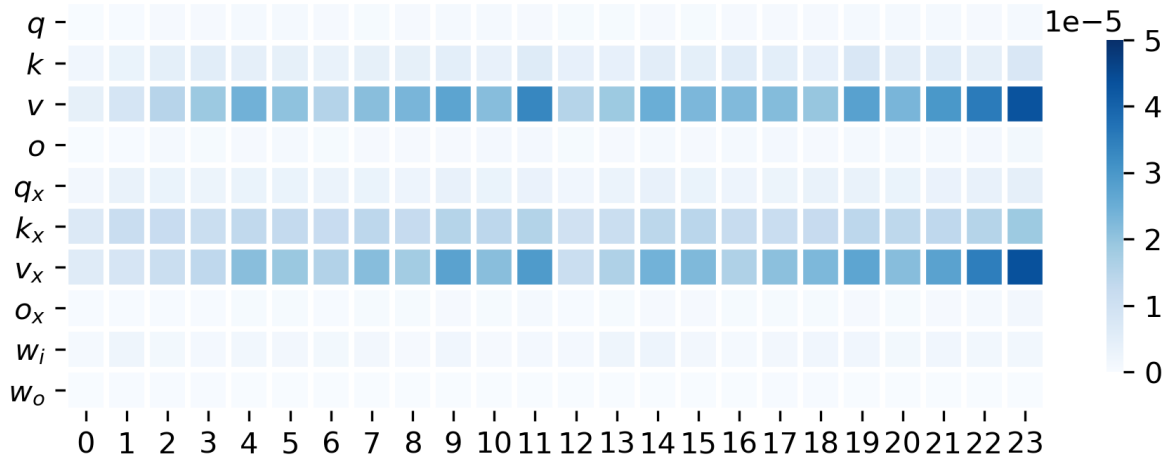

Figure 24: L1 change, Encoder, T5-11B, $n = 3$

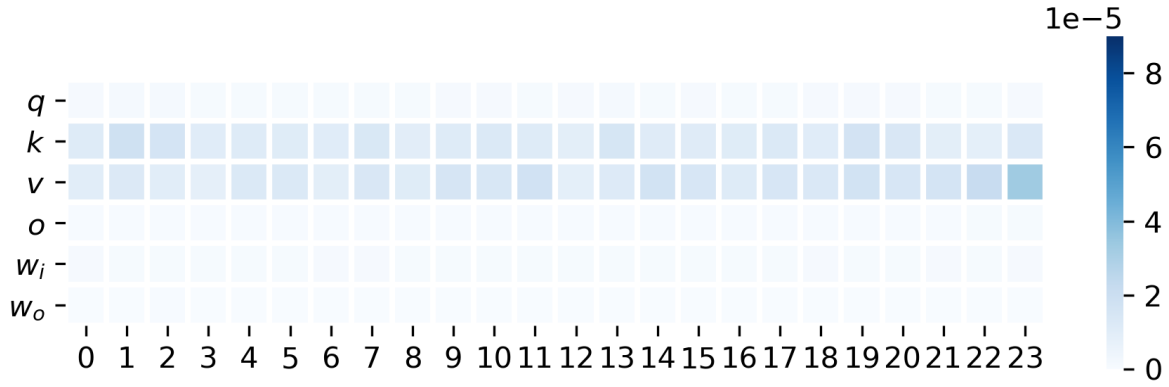

Figure 25: L1 change, Encoder, T5-11B, $n = 30$

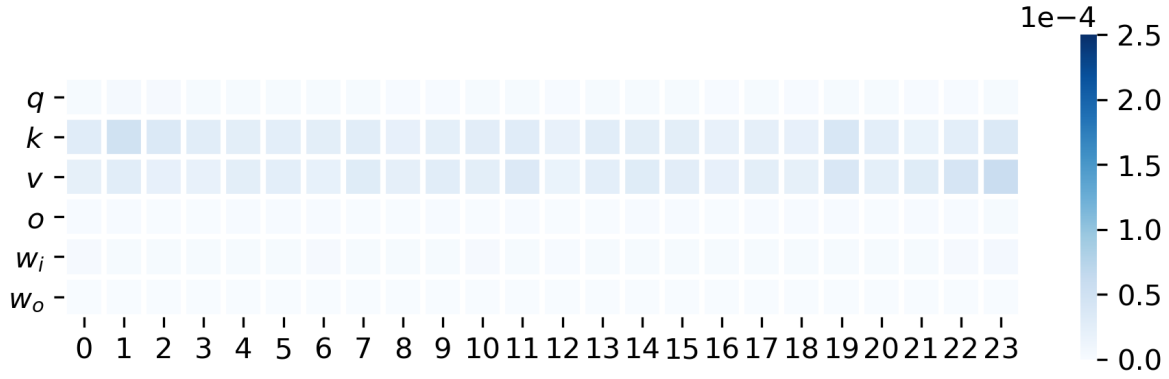

Figure 26: L1 change, Encoder, T5-11B, $n = 300$

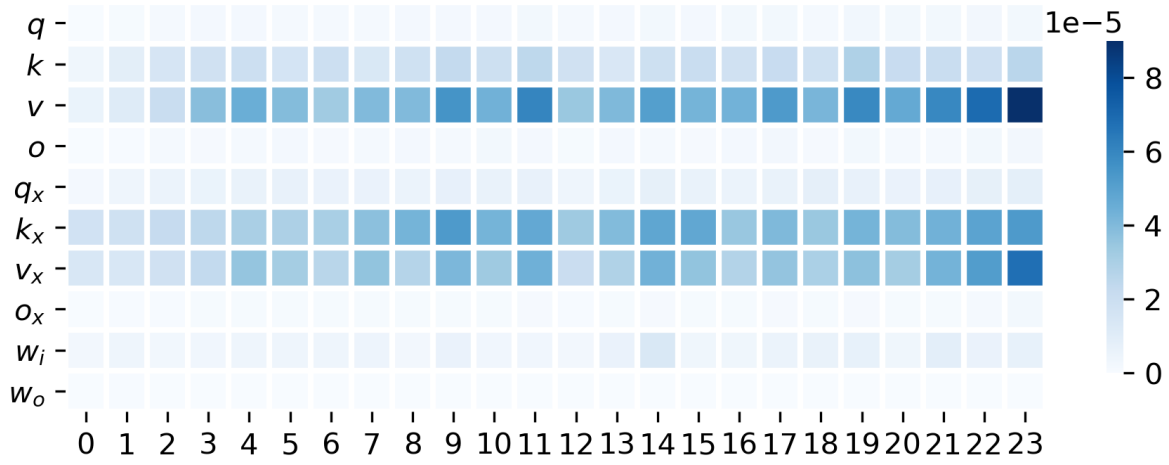

Figure 27: L1 change, Decoder, T5-11B (relation embedding), $n = 30$

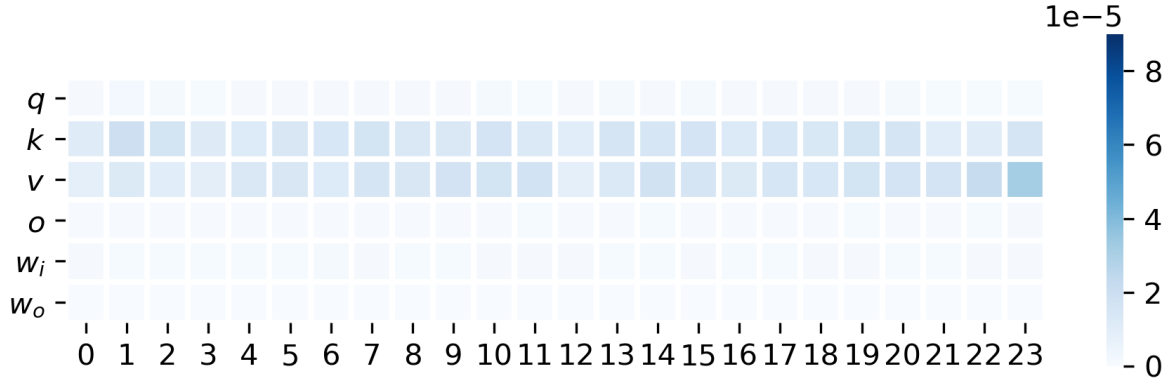

Figure 28: L1 change, Encoder, T5-11B (relation embedding), $n = 30$

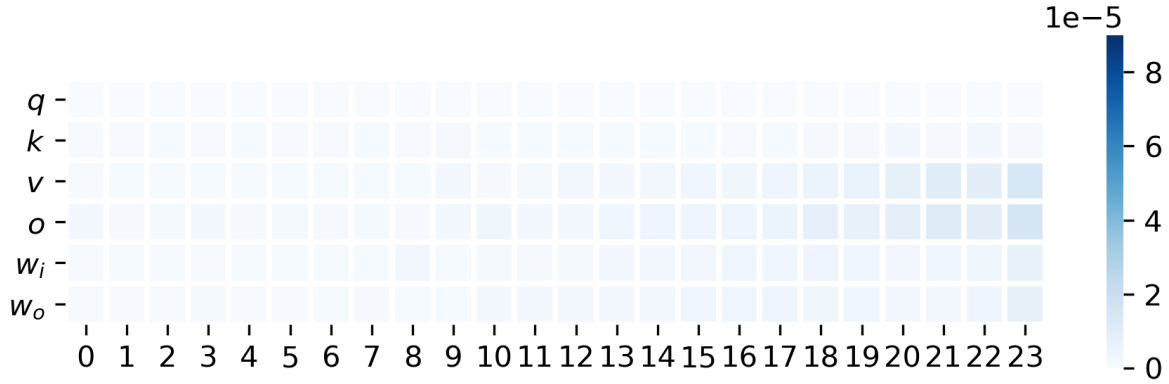

Figure 29: L1 change, Encoder, T5-Large, $n = 30$

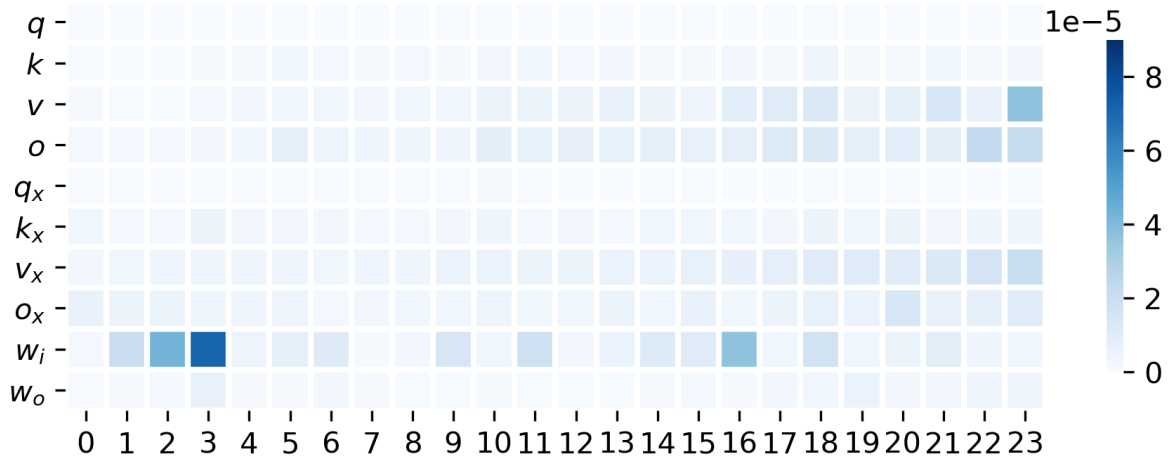

Figure 30: L1 change, Decoder, T5-Large, $n = 30$

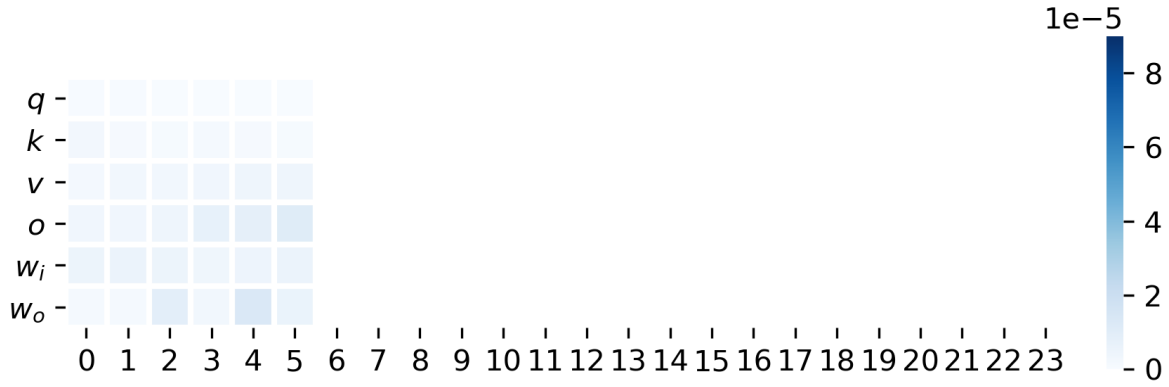

Figure 31: L1 change, Encoder, T5-Small, $n = 30$

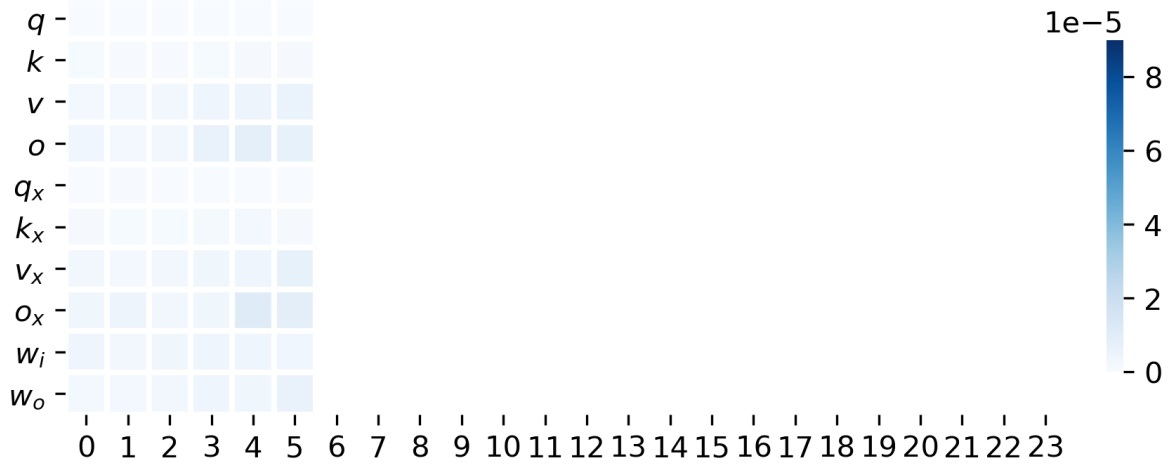

Figure 32: L1 change, Decoder, T5-Small, $n = 30$

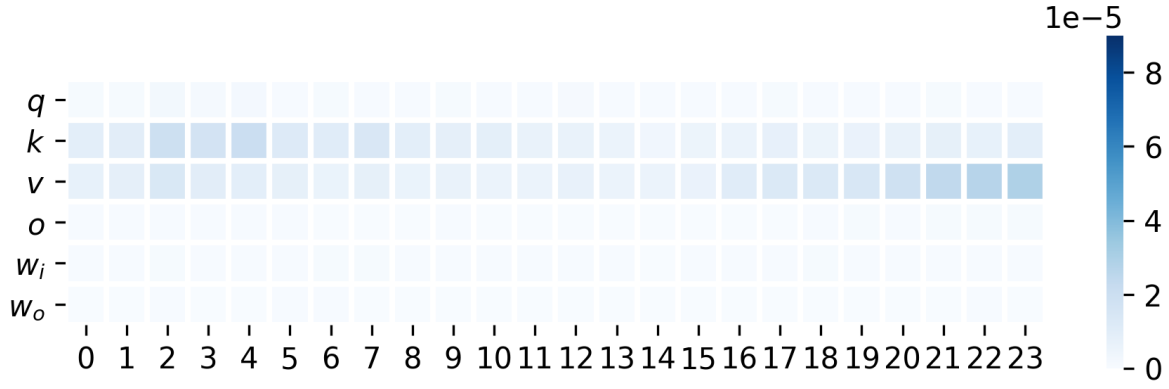

Figure 33: L1 change, Encoder, T5-11B, Shuffled Prompts, $n = 30$

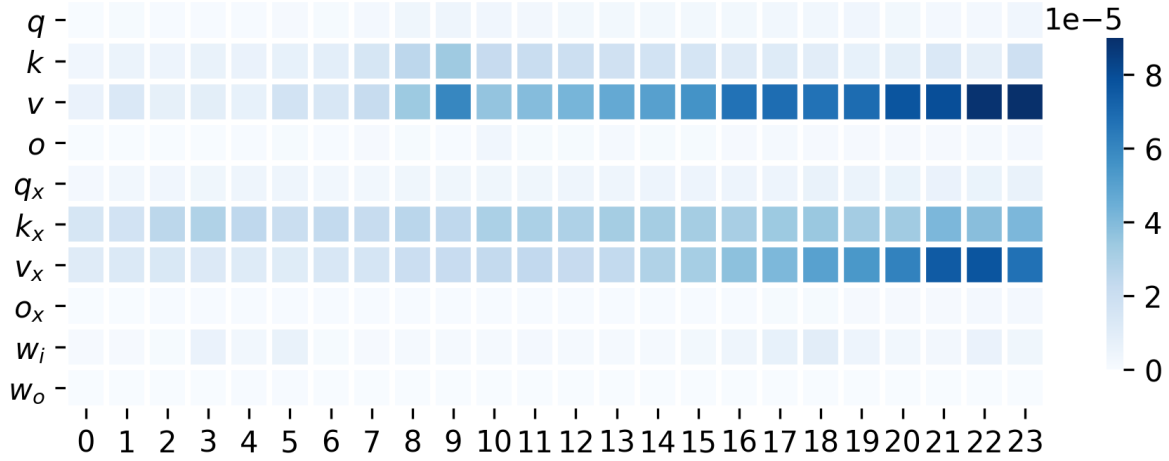

Figure 34: L1 change, Decoder, T5-11B, Shuffled Prompts, $n = 30$

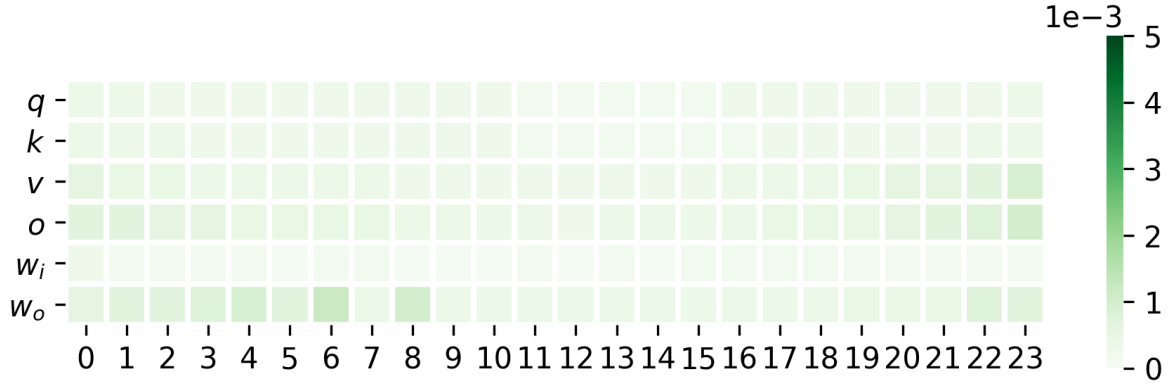

Figure 35: Angular change, Encoder, T5-11B, Shuffled Prompts, $n = 30$

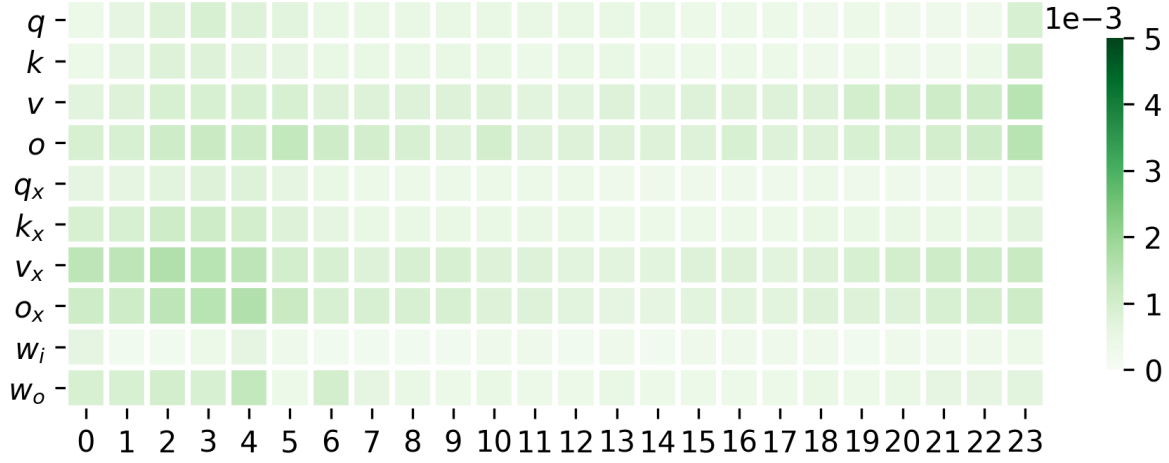

Figure 36: Angular change, Decoder, T5-11B, Shuffled Prompts, $n = 30$

Appendix A. Accuracy in zero-shot MLM setting

While little work has explored few-shot knowledge completion, recent works have investigated performance of zero-shot knowledge graphs (**??**). Thus, we investigate the ability of T5-11B to complete commonsense knowledge in a zero-shot setting.

Model	BLEU-1	METEOR	ROUGE-L	CIDEr
T5 - Zero-shot	6.7	7.8	7.3	7.3
T5 - Few-shot ($n = 3$)	31.9	18.7	26.0	19.8
T5 - Fully-supervised	48.2	34.1	50.0	66.4

Table 1: Zero-shot performance of T5-11B

We use prompts to leverage the masking objective of the language model pretraining. Since the mask only predicts several tokens at a time, for relations with longer length tail entities, we allow the model to predict up to 7 mask tokens in succession, or until the model predicts an empty string for the mask. We suggest that this is still only a workaround, and masked models are poor predictors of longer length tail entities, as indicated by our results above.

Appendix B. Additional Results on the Effect of Relation Input Format

In Table 2, we provide further experimental results on the effect of relation input format on few-shot performance. Namely, we extend the results from Table **??** with the case $n = 300$. These results confirm that knowledge models can efficiently learn from fewer examples when relations are represented using natural language prompts. We also show the results from using paraphrases of the main prompts for training (original and paraphrased prompts can be found in Tables 3 and 5, respectively). We find that the paraphrased prompts do cause a slight drop in performance, but that this drop is generally close to the margin of error, indicating that while prompt formulation is an important consideration (**?**), fine-tuning on the prompts does make the model less sensitive to prompt variations.

# Ex	Input	BLEU-1	METEOR	ROUGE-L	CIDEr
3	Prompts	**24.2** ± 1.7	**14.4** ± 1.9	**21.3** ± 3.7	**18.1** ± 7.9
	Embedding	13.9 ± 1.3	11.4 ± 1.2	13.5 ± 0.9	7.4 ± 0.8
30	Prompts	**31.9** ± 0.3	**18.7** ± 0.5	**26.0** ± 0.6	**19.8** ± 1.2
	Embedding	18.1 ± 0.8	13.5 ± 1.1	16.7 ± 1.1	9.7 ± 1.7
	Shuffled Prompts	15.6 ± 0.8	11.3 ± 0.5	14.2 ± 0.5	8.3 ± 0.8
	Paraphrased	30.5 ± 1.3	18.3 ± 0.5	24.5 ± 0.8	18.4 ± 1.4
300	Prompts	**39.1** ± 0.9	**26.7** ± 0.9	**40.1** ± 1.5	**48.5** ± 2.8
	Embedding	25.2 ± 0.8	18.5 ± 0.9	26.4 ± 1.9	26.8 ± 4.3

Table 2: Effect of relation input format on few-shot performance. Prompts accelerate few-shot commonsense interface learning. We show mean performance over 5 random splits of training examples, and standard deviation (±) between splits.

Relation	Template
ObjectUse	{} is used for
AtLocation	You are likely to find {} in
MadeUpOf	{} is made up of
HasProperty	{} is
CapableOf	{} can
Desires	{} wants
NotDesires	{} does not want
isAfter	Something that happens after {} is
HasSubEvent	Something you might do while {} is
isBefore	Something that happens before {} is
HinderedBy	{} is hindered by
Causes	Sometimes {} causes
xReason	{}. The reason for PersonX doing this is
isFilledBy	{} can be filled by
xNeed	But before {}, PersonX needed
xAttr	{} is seen as
xEffect	As a result of {}, PersonX will
xReact	As a result of {}, PersonX feels
xWant	After {}, PersonX would want
xIntent	Because of {}, PersonX wanted
oEffect	as a result of {}, others will
oReact	as a result of {}, others would feel
oWant	as a result of {}, others would want

Table 3: Prompts used for relations in ATOMIC2020.

Something you might do while {} is ___
Something you might do while design software is determine deliverables
Something you might do while scuba dive is take off scuba gear
Something you might do while play ball is put on mitt

Table 4: Example of augmentation experiments, following (**?**). {} indicates the location of the head, and the language model is asked to complete the tail by finishing the sentence.

Relation	Template
ObjectUse	a {} can be used for
AtLocation	You could find {} in the location
MadeUpOf	{} is made up of
HasProperty	{} will have
CapableOf	{} is capable of
Desires	a {} desires
NotDesires	a {} does not desire
isAfter	Before {},
HasSubEvent	You might do {} while doing
isBefore	After {},
HinderedBy	{}. This is hindered by
Causes	Sometimes {} causes
xReason	{}. PersonX did this because
isFilledBy	{} is filled
xNeed	Before {}, PersonX needs to
xAttr	{}. An attribute of PersonX is
xEffect	The effect of {} PersonX will be
xReact	As a result of {}. PersonX will be
xWant	After {}, PersonX will want to
xIntent	For {}, PersonX did this to
oEffect	An effect of {} on others will be
oReact	As a result of {}, other feel
oWant	After {}, others will want to

Table 5: Paraphrased version of the prompts used (for paraphrased experiment in the Appendix.)

Appendix C. Additional Experiments on Parameter Change Measures

In addition, we provide extensive results of the ℓ_1 and angular distances, as well as the distributional parameter change metric (AUC), for both the encoder and decoder of various model sizes (Small, Large, and 11B) and different example budget ($n \in \{3, 30, 300\}$) (Figures 1-30). When computing AUC diagrams, we round each weight change to the nearest 10^{-5}.

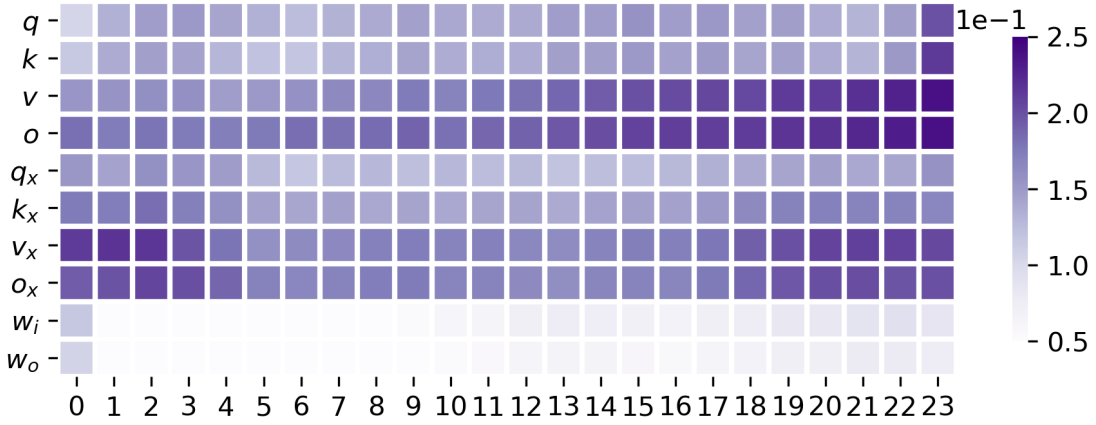

Figure 1: Area Under Curve, Decoder, T5-11B, $n = 30$

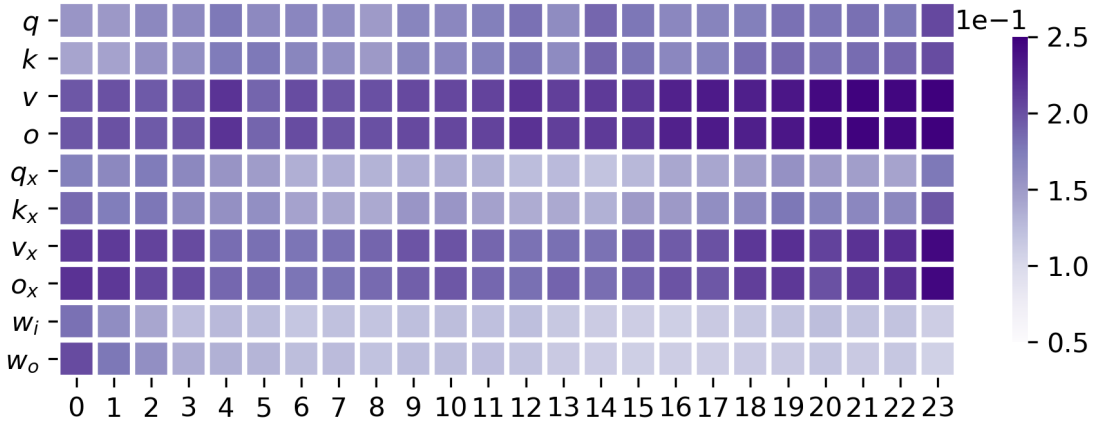

Figure 2: Area Under Curve, Decoder, T5-Large, $n = 30$

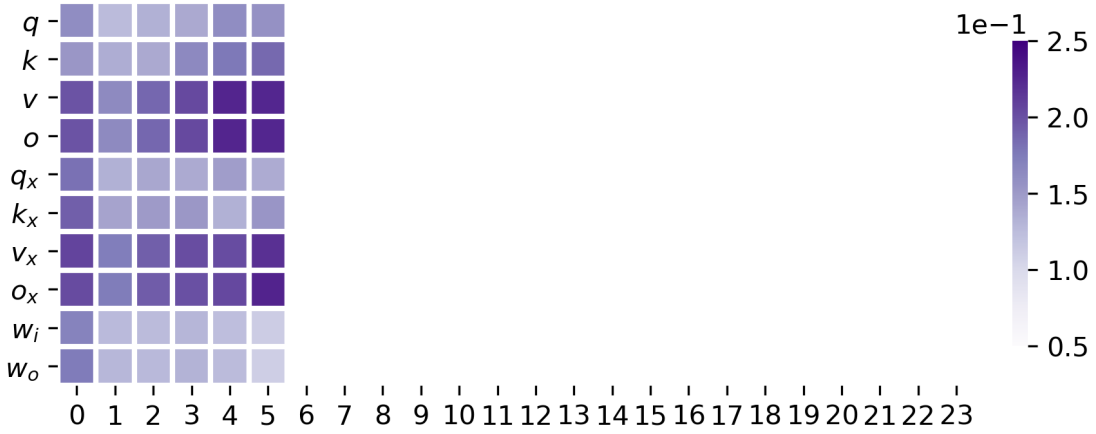

Figure 3: Area Under Curve, Decoder, T5-Small, $n = 30$

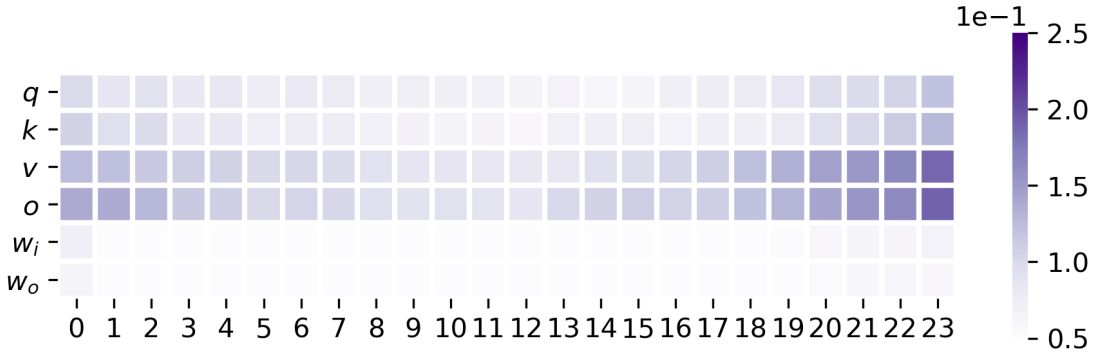

Figure 4: Area Under Curve, Encoder, T5-11B, $n = 30$

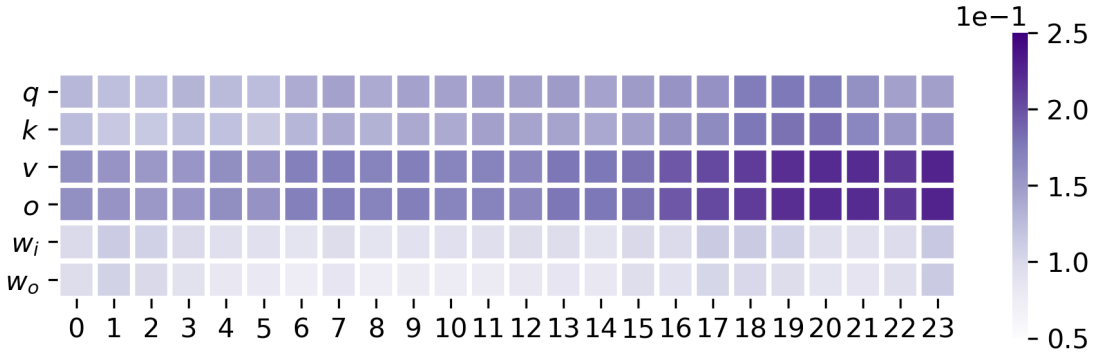

Figure 5: Area Under Curve, Encoder, T5-Large, $n = 30$

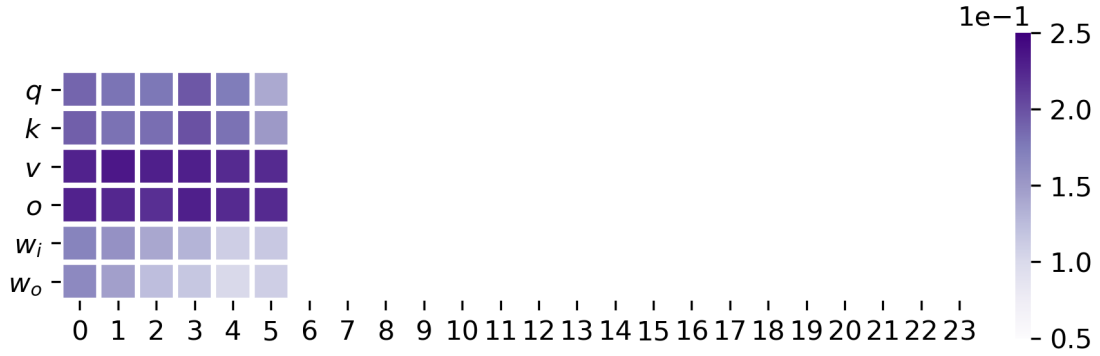

Figure 6: Area Under Curve, Encoder, T5-Small, $n = 30$

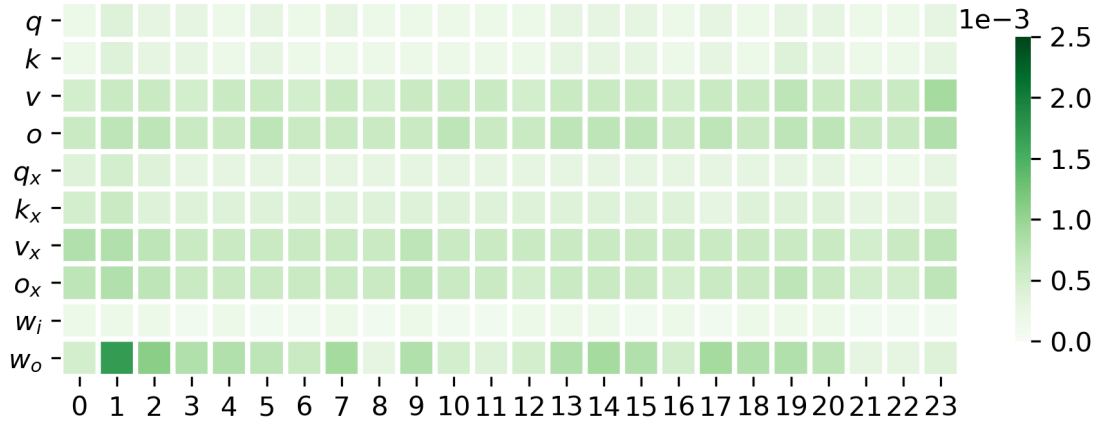

Figure 7: Angular change, Decoder, T5-11B, $n = 3$

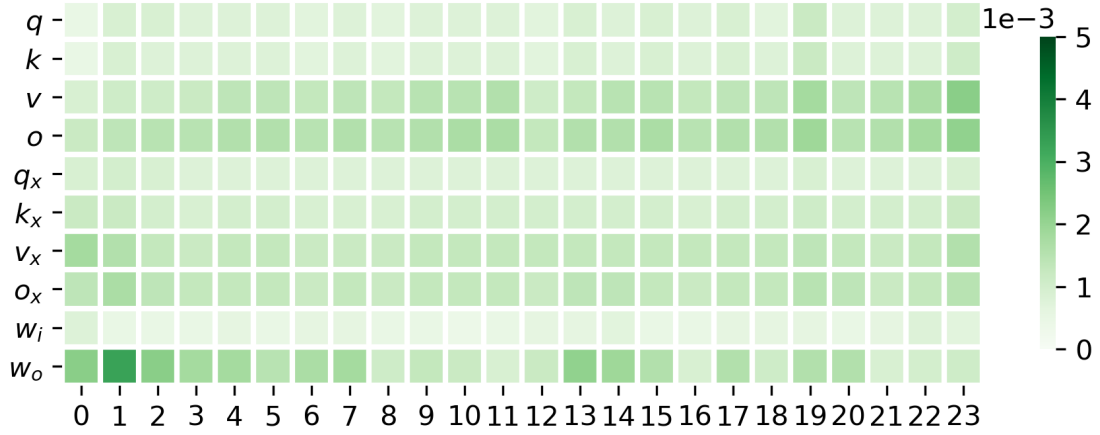

Figure 8: Angular change, Decoder, T5-11B, $n = 30$

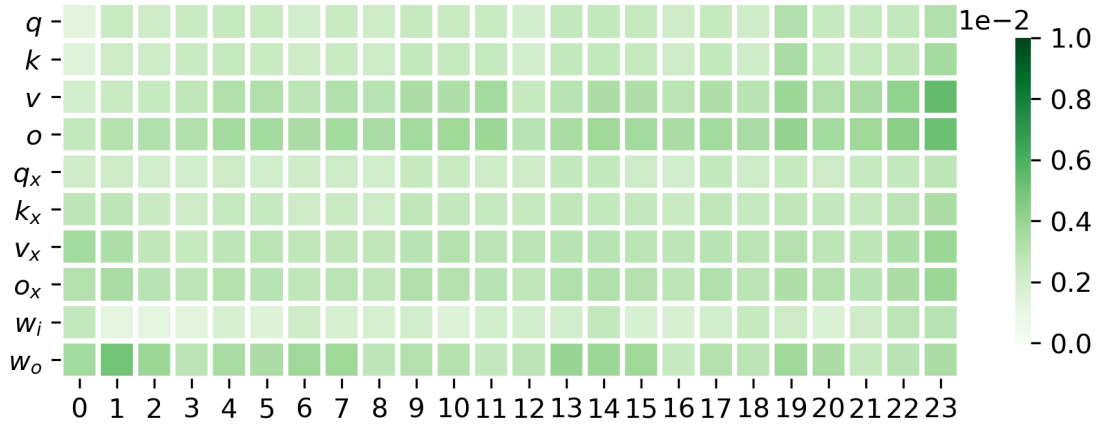

Figure 9: Angular change, Decoder, T5-11B, $n = 300$

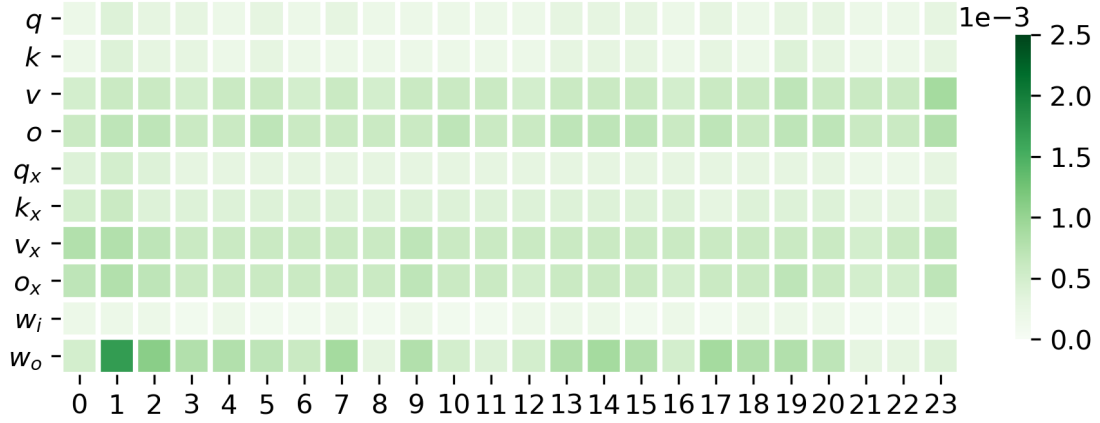

Figure 10: Angular change, Encoder, T5-11B, $n = 3$

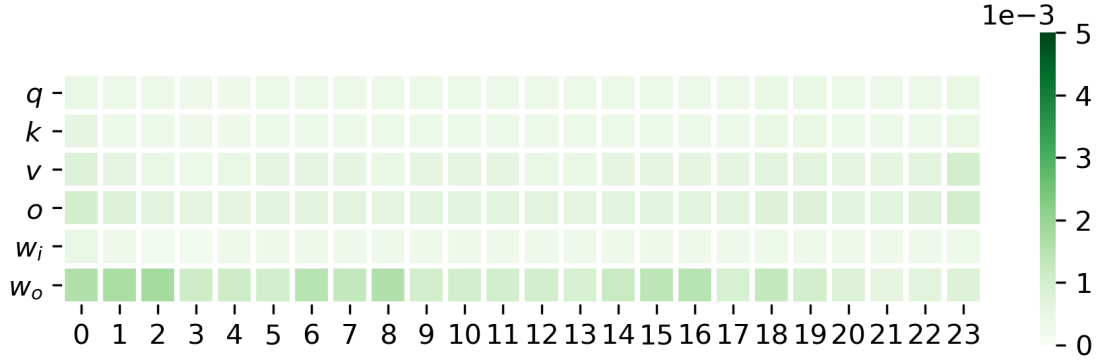

Figure 11: Angular change, Encoder, T5-11B, $n = 30$

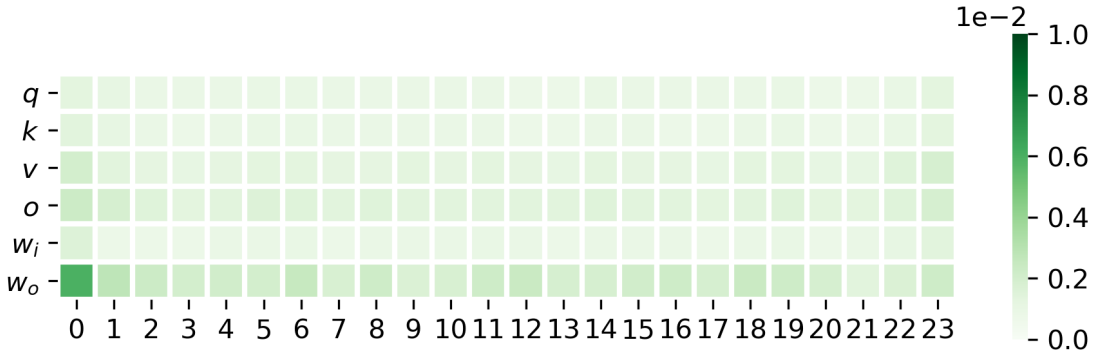

Figure 12: Angular change, Encoder, T5-11B, $n = 300$

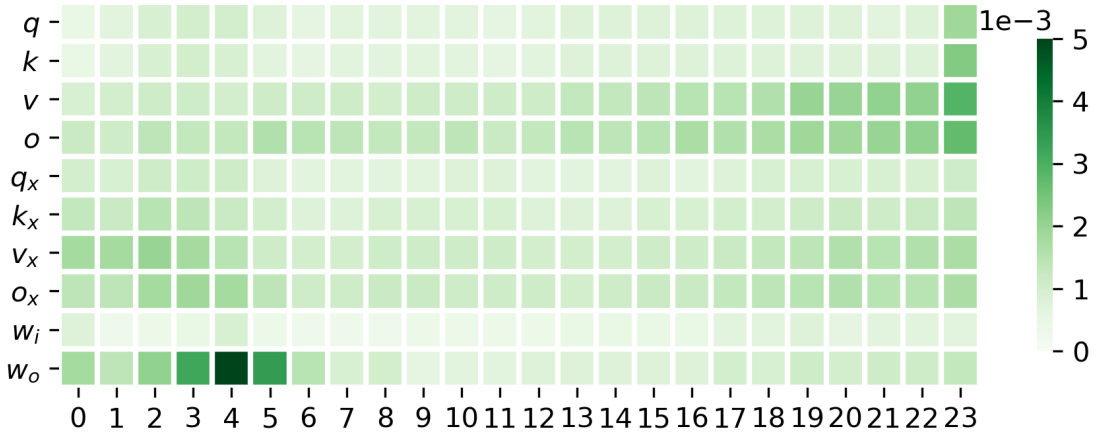

Figure 13: Angular change, Decoder, T5-11B (relation embedding), $n = 30$

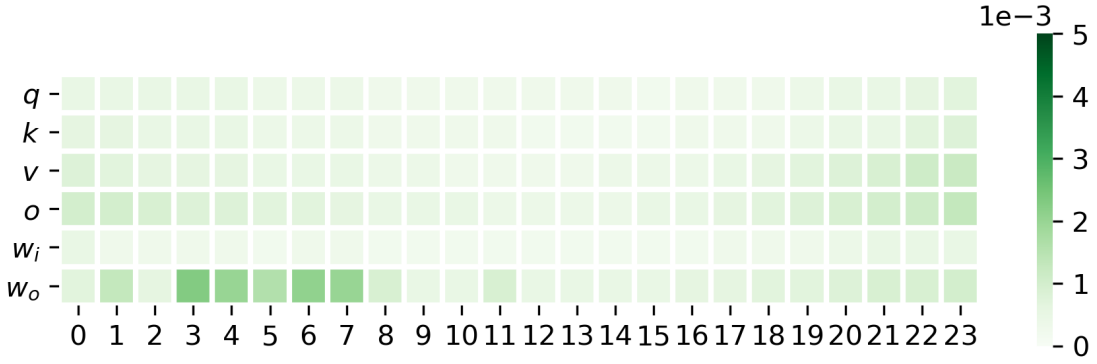

Figure 14: Angular change, Encoder, T5-11B (relation embedding), $n = 30$

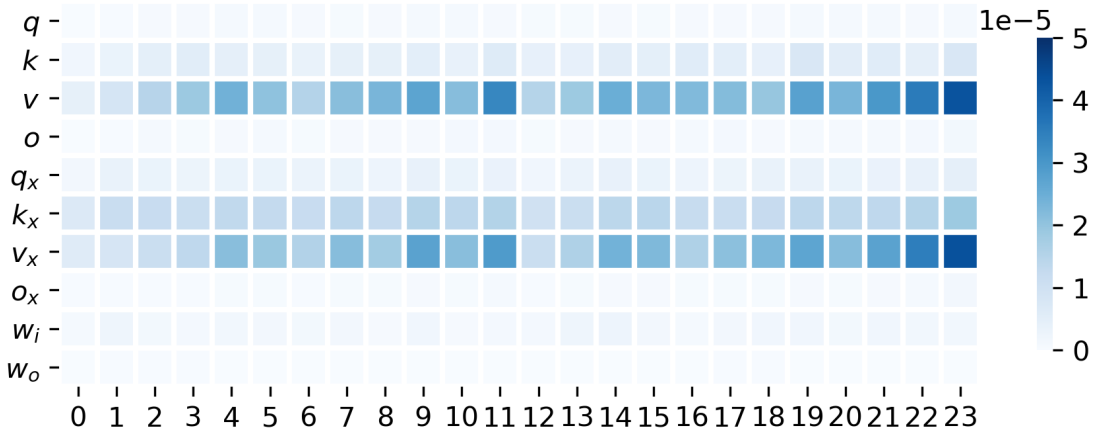

Figure 15: L1 change, Decoder, T5-11B, $n = 3$

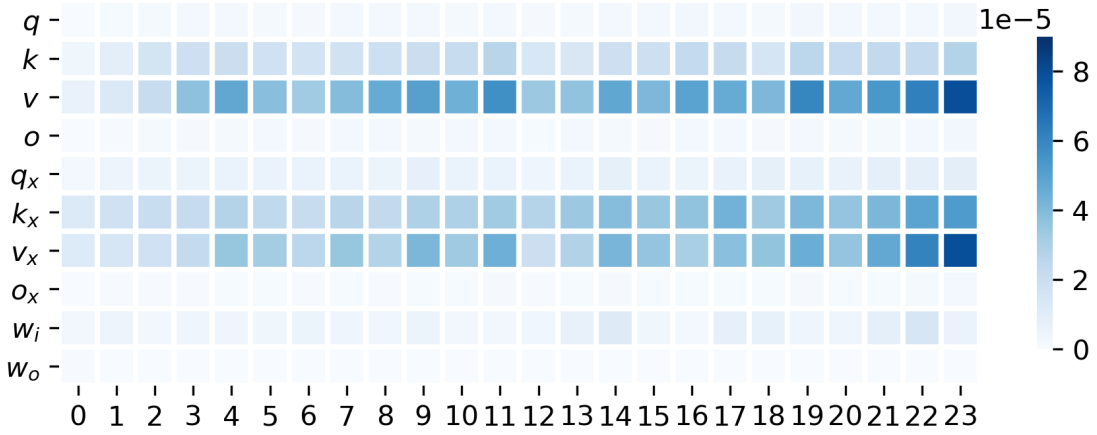

Figure 16: L1 change, Decoder, T5-11B, $n = 30$

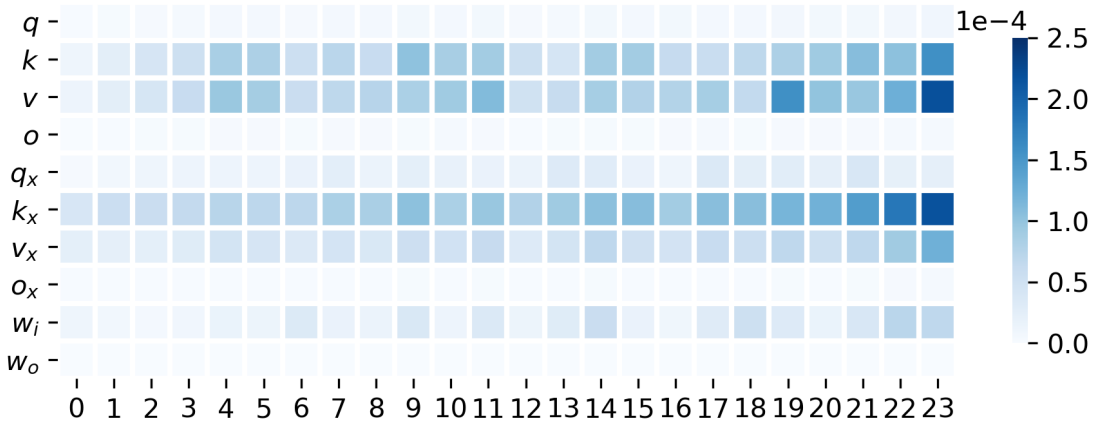

Figure 17: L1 change, Decoder, T5-11B, $n = 300$

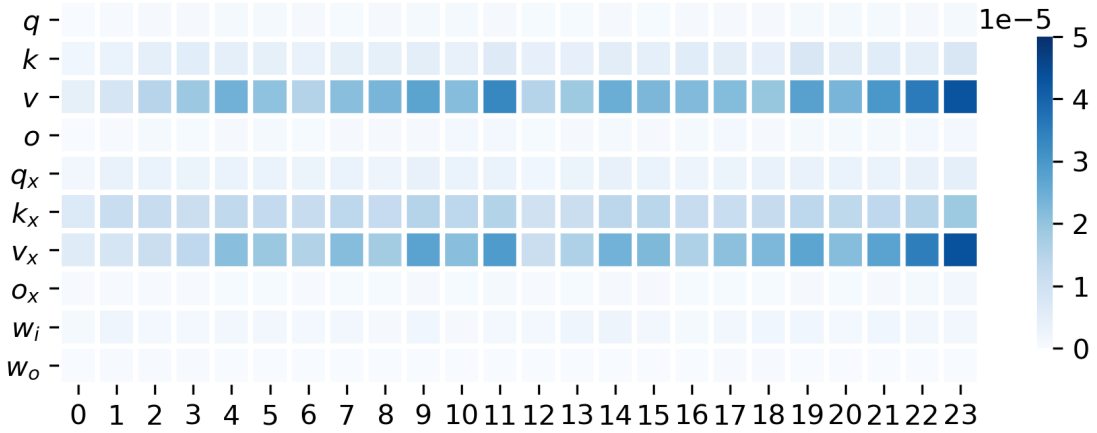

Figure 18: L1 change, Encoder, T5-11B, $n = 3$

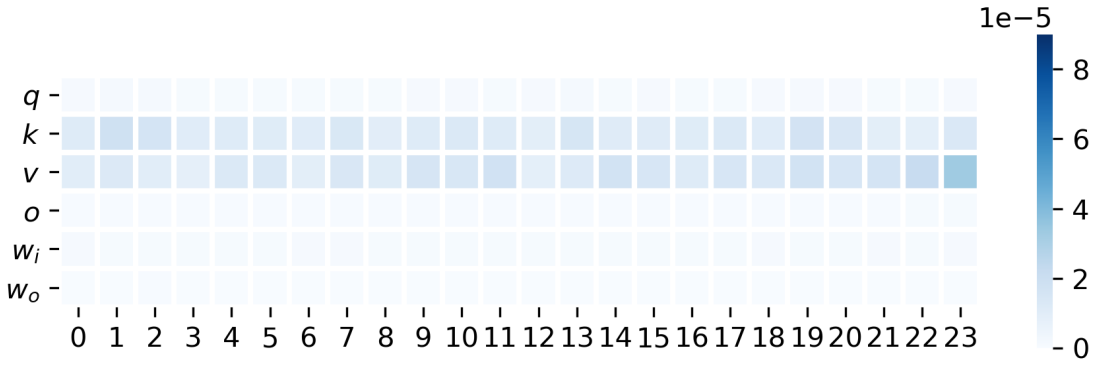

Figure 19: L1 change, Encoder, T5-11B, $n = 30$

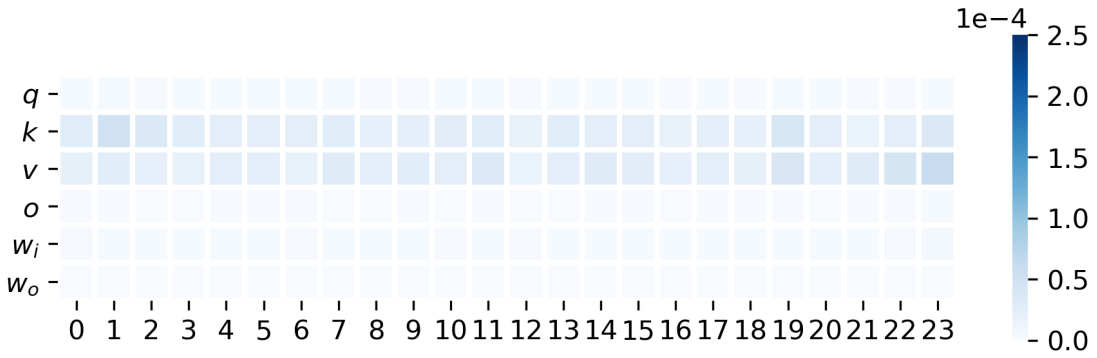

Figure 20: L1 change, Encoder, T5-11B, $n = 300$

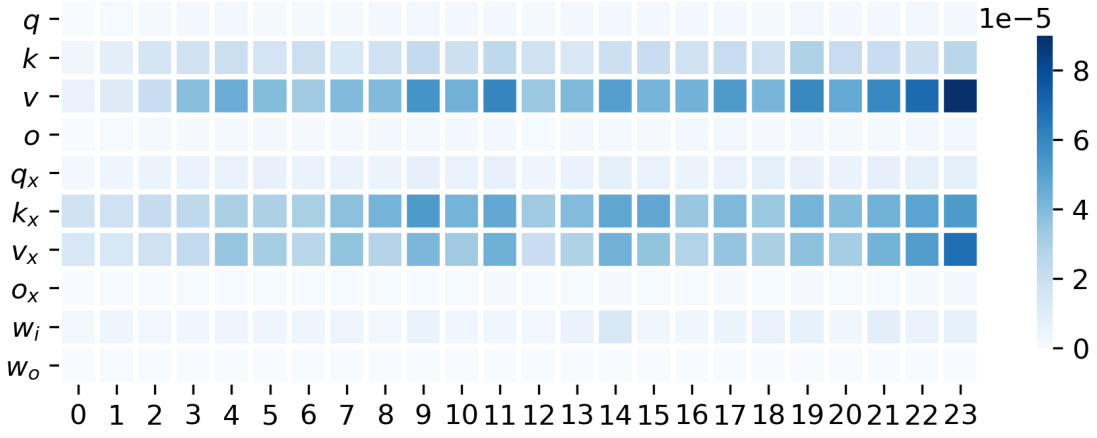

Figure 21: L1 change, Decoder, T5-11B (relation embedding), $n = 30$

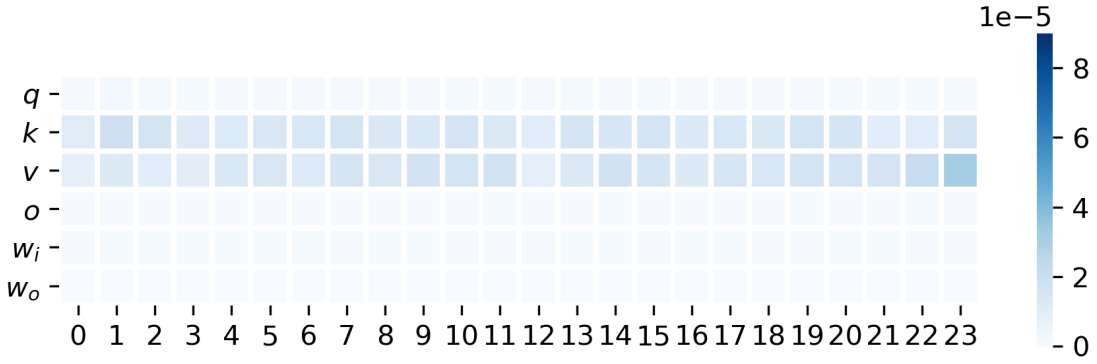

Figure 22: L1 change, Encoder, T5-11B (relation embedding), $n = 30$

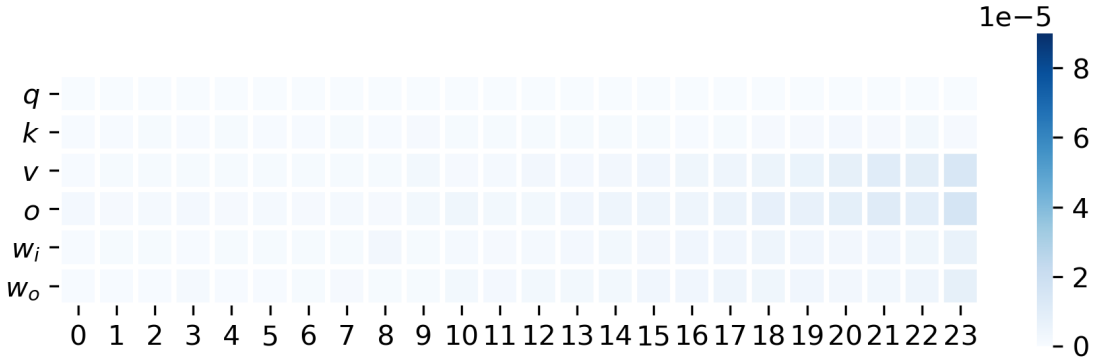

Figure 23: L1 change, Encoder, T5-Large, $n = 30$

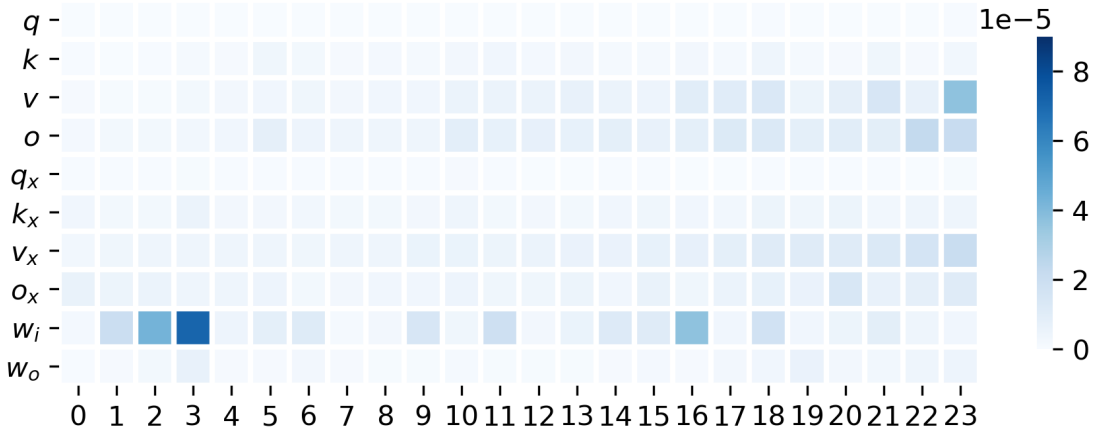

Figure 24: L1 change, Decoder, T5-Large, $n = 30$

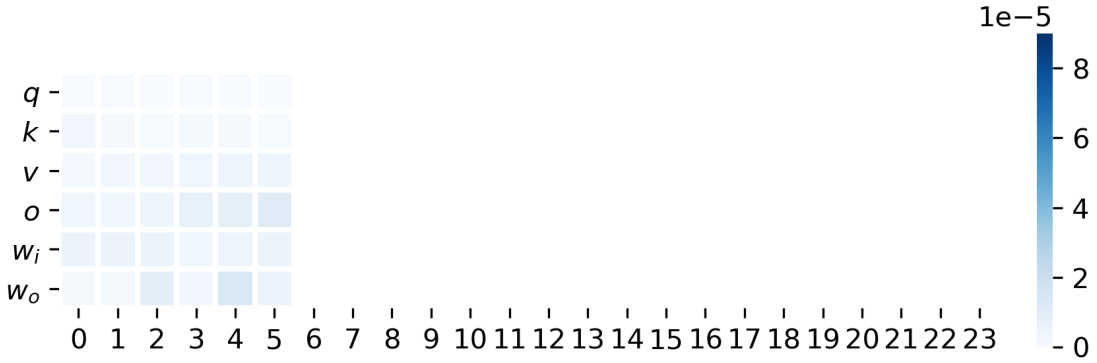

Figure 25: L1 change, Encoder, T5-Small, $n = 30$

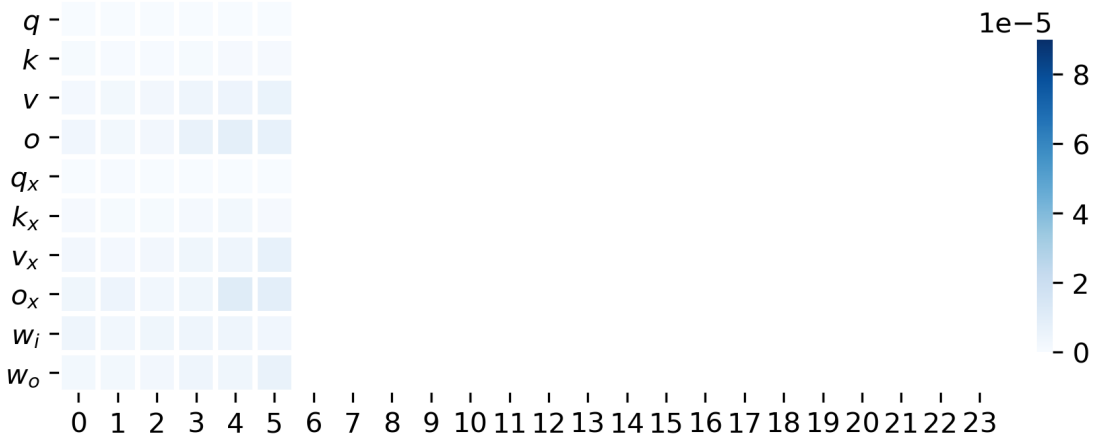

Figure 26: L1 change, Decoder, T5-Small, $n = 30$

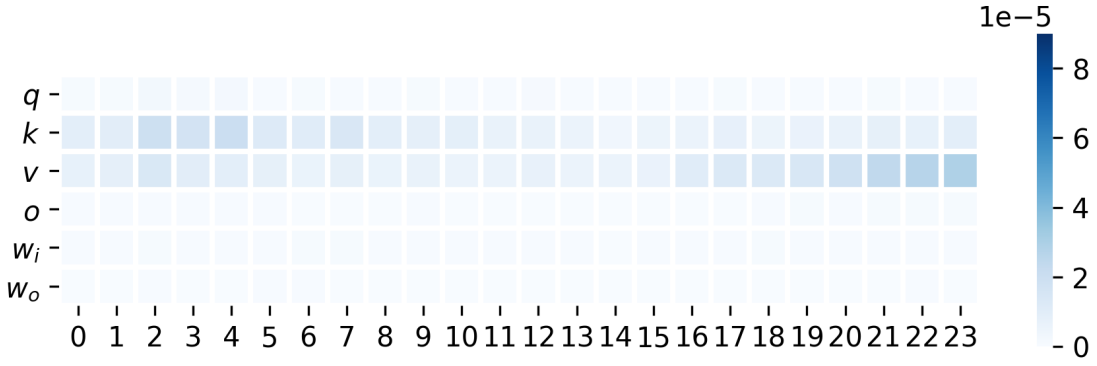

Figure 27: L1 change, Encoder, T5-11B, Shuffled Prompts, $n = 30$

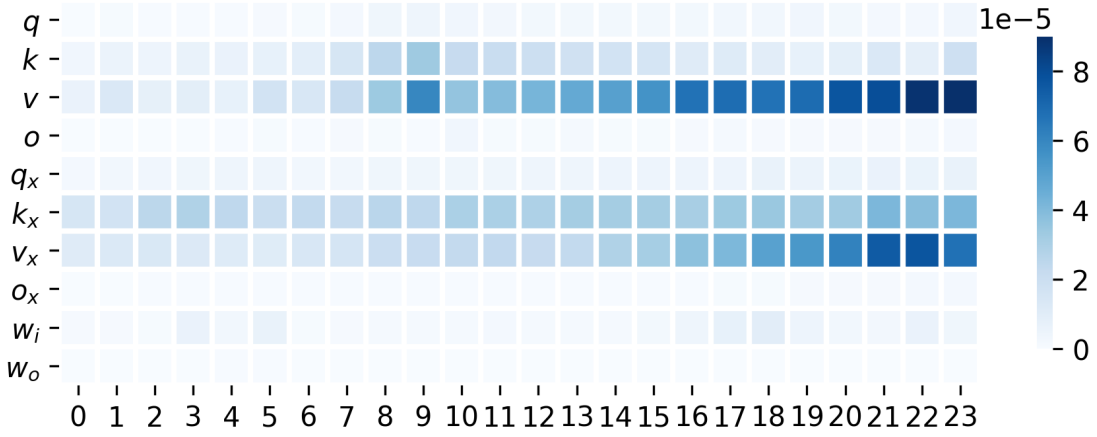

Figure 28: L1 change, Decoder, T5-11B, Shuffled Prompts, $n = 30$

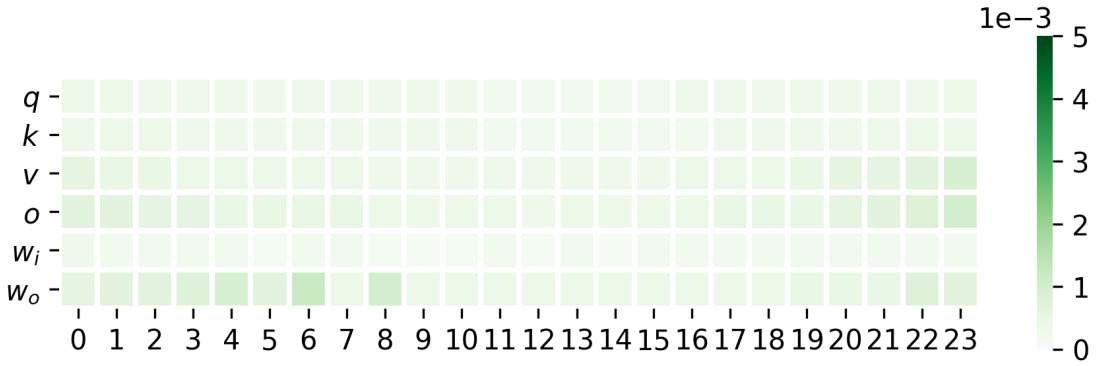

Figure 29: Angular change, Encoder, T5-11B, Shuffled Prompts, $n = 30$

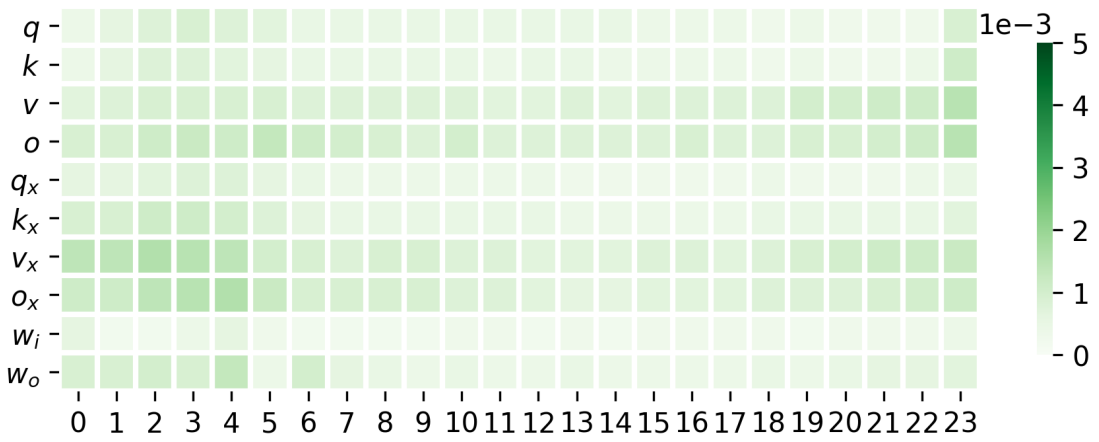

Figure 30: Angular change, Decoder, T5-11B, Shuffled Prompts, $n = 30$