Memory Augmented Language Models through Mixture of Word Experts

Cicero Nogueira dos Santos, James Lee-Thorp, Isaac Noble
Chung-Ching Chang, David Uthus
Google Research
cicerons,jamesleethorp,isaacn,ccchang,duthus@google.com

Abstract

Scaling up the number of parameters of language models has proven to be an effective approach to improve performance. For dense models, increasing their size proportionally increases their computational footprint. In this work, we seek to aggressively decouple learning capacity and FLOPs through Mixture-of-Experts (MoE) style models with large knowledge-rich vocabulary based routing functions. Our proposed approach, dubbed Mixture of Word Experts (MoWE), can be seen as a memory augmented model, where a large set of word-specific experts play the role of a sparse memory. We demonstrate that MoWE performs significantly better than the T5 family of models with similar number of FLOPs in a variety of NLP tasks. Moreover, MoWE outperforms traditional MoE models on knowledge intensive tasks and has similar performance to complex memory augmented approaches that often require to invoke custom mechanisms to search the sparse memory.

1 Introduction

Increasing the parameter count of language models has been a primary driver of increased model quality (Raffel et al., 2020; Kaplan et al., 2020; Brown et al., 2020). This is particularly apparent on knowledge intensive tasks, such as TriviaQA (Joshi et al., 2017), where language models with more parameters and learning capacity benefit from soaking up world knowledge from their pretraining data (Chowdhery et al., 2022; Touvron et al., 2023). However, increasing the model size also increases the cost of training and serving the model.

In this work, we build on the Mixture-of-Experts (MoE) paradigm to design a neural net architecture that enjoys the quality benefits from scaling the parameter count but remains FLOPs and latency efficient. Our proposed approach, which we name Mixture-of-Word-Experts (MoWE), follows two design principles: (1) a very large number of experts (tens of thousands instead of 32 to 128 normally used in MoEs layers) that (2) are "word-specific" – that is, they are tied to a large knowledge-rich vocabulary through a fixed routing function. Our design principles are based on the conjecture that using large knowledge-rich vocabularies to perform routing is a principled and effective approach to induce structured sparseness in MoE models (Secs. 2.4 and 4.5.1); which associated to a large number of experts results in memory augmented language model whose *structured* sparse memory is seamlessly integrated to the main model backbone (Secs. 2.2 and 4.3).

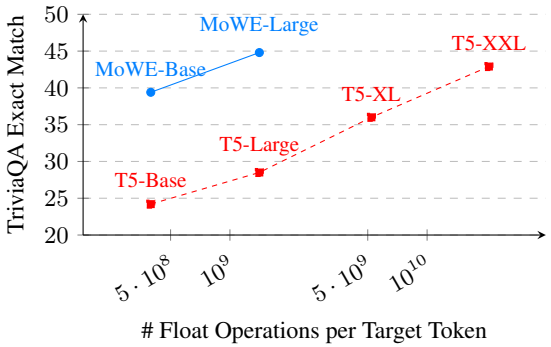

Figure 1: **MoWE vs T5.1.1 on TriviaQA**: MoWE-Base and MoWE-Large perform as well as T5.1.1-XL and T5.1.1-XXL, respectively, while using a significantly smaller number of FLOPs. T5.1.1 results are from Roberts et al. (2020).

We empirically demonstrate that MoWE significantly outperforms T5 models (Raffel et al., 2020) with a comparable number of FLOPs across a variety of NLP tasks. Focusing on knowledge intensive tasks such as TriviaQA (Joshi et al., 2017) and WebQuestions (Berant et al., 2013), we show that a MoWE "Base" sized outperforms T5-XL and a MoWE "Large" outperforms T5-XXL models (see Figure 1), while being at least 4.3x and 6.6x faster to train, respectively. MoWE outperforms vanilla MoE models (Shazeer et al., 2017; Lepikhin et al.,

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 4425–4438

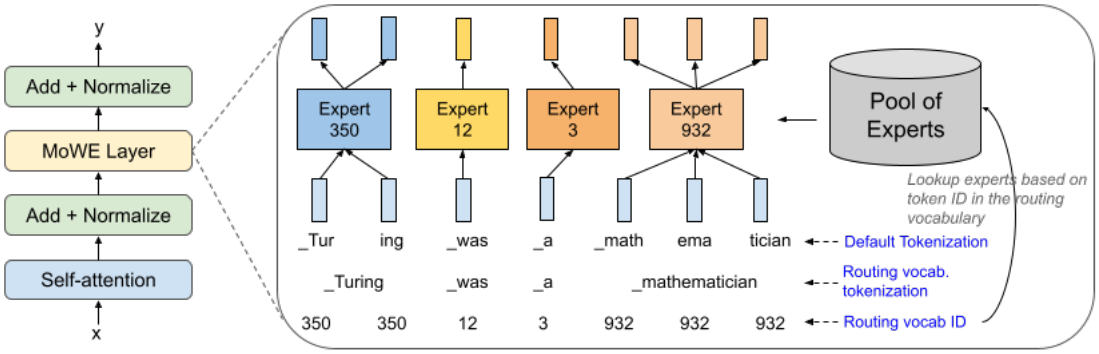

Figure 2: **MoWE Layer**: We replace the FFN layer in a subset of Transformer blocks by a *MoWE Layer*, which is a sparse layer that processes tokens using multiple experts (FFNs). Each input token is processed by a single expert that is selected based on the input token id (at the corresponding sequence position) in the routing vocabulary.

2020; Fedus et al., 2022) on knowledge intensive tasks, while matching performance on NLP task suites such as SuperGLUE (Wang et al., 2019a). Additionally, MoWE also matches or outperforms recently proposed knowledge augmented models (Févry et al., 2020; de Jong et al., 2022), while avoiding invoking any custom mechanism to search the sparse memory.

The main contributions of this work are:

- We propose a novel neural net architecture that effectively combines the efficiency of sparse models with the power of large language models to memorize and retrieve world knowledge; see Table 4 for a downstream peek at how these memories are used.
- We introduce routing functions based on large knowledge-rich vocabularies. Additionally, we propose and validate a new strategy to efficiently train MoE models with: (1) hundreds of thousands of experts and (2) very unbalanced token assignments across experts.
- For knowledge intensive tasks, we present new efficient sparse models that outperform significantly slower dense models that use an order of magnitude more FLOPs.

2 Mixture-of-Word-Experts

2.1 Mixture-of-Experts (MoE) Background

Transformer-based MoE architectures (Lepikhin et al., 2020; Du et al., 2022; Fedus et al., 2022) are implemented by replacing the dense Feed Forward Network (FFN) layer in a subset of Transformer blocks with a sparse layer of experts. Instead of using a single FFN to process all inputs, the sparse layer employs a set of FFNs (the experts). Each token representation is processed by a single (top-1) or a subset (top-k) of experts. The promise in MoE models is to vastly increase the number of parameters in the network without significantly increasing the amount of computation.

Common MoE implementations replace every other FFN layer of the Transformer architecture by a sparse layer that contains between 32 and 128 experts (Lepikhin et al., 2020; Du et al., 2022; Fedus et al., 2022). Tokens are assigned to particular experts by a *routing function* that is learned jointly with the rest of the parameters of the network. Because of the nature of the one-hot assignments of tokens to experts, training the routing function is tricky and typically performed indirectly by rescaling expert outputs by the assignment probability (the "router confidence") that a given token should be assigned to a particular expert.

2.2 Mixture-of-Word-Experts (MoWE) Architecture

Similar to MoE models, MoWE is a Transformer-based architecture (Vaswani et al., 2017) where the FFN layer of a subset of Transformer blocks is replaced by a *MoWE Layer*, which is a sparse layer that processes tokens using a pool of experts (FFNs). In a MoWE layer, a token representation at position i is processed by a single expert that is selected based on the id, in the *routing vocabulary*, of the corresponding *input* sequence token at position i. Figure 2 illustrates a MoWE layer.

Routing decisions are driven by a large auxiliary vocabulary. There are two tokenizations of the input: (1) the *default tokenization* which is the regular one that defines the input tokens and their embeddings; and (2) the *routing tokenization*,

which is performed using a large auxiliary *routing vocabulary* (introduced in Section 2.4). The token ids resulting from the routing tokenization are called *routing ids*. In a MoWE layer, routing consists of mapping routing ids to experts ids through a hash function. In the extreme case where each word in the routing vocabulary has its own expert, the routing id corresponds directly to the expert id, as illustrated in Figure 2.

Importance of a large pool of experts. A MoWE layer uses tens or hundreds of thousands of experts, which are normally smaller (smaller MLP dimension) than the regular, dense FFN layer. The goal of using a large number of experts is to encourage specialization. With an extremely large number of experts, each word in the routing vocabulary is assigned to its own expert. However, we found that it is more efficient (both in terms of memory and training signal) to have fewer experts than vocabulary entries and share some experts across multiple routing ids. Nevertheless, a token with a given id is always routed to the same expert.

Recent work suggests that Transformers act as key-value memories (Geva et al., 2021; Dai et al., 2022; Zhang et al., 2022), and that factual knowledge seems to be stored in the FFNs (Dai et al., 2022; Meng et al., 2022). We conjecture that the large routing vocabulary and associated large number of experts further encourage the MoWE layer to function as a sparse memory. We find that using complete words instead of word pieces (see Section 2.4) to perform routing is a strong inductive bias that makes it easier for the experts to specialize on specific words. For example, the expert for the word "Turing" will be activated only when that word appears in the input, and therefore will be specialized on content that co-occur with that word. By using word-specific *key-value memories* (word experts), our hope is that MoWE can make it easier for the model to store and retrieve information about those words.

2.3 Overcoming the Challenges of using Tens of Thousands of Experts

Most large scale MoE models are implemented using the single program, multiple data (SPMD) parallelism strategy; see, for example, Lepikhin et al. (2020). Data and experts are cosharded across devices. Data that is originally on device x but is assigned, by the routing function, to an expert on device y must be transferred between devices through all-to-all communications. Under the single program paradigm on modern accelerators, experts send and receive the same amount of data and perform that same amount of computation (same array shapes on each device). Effectively implementing MoWE using vanilla SPMD poses some key challenges: (1) The sheer number of experts brings an unpractical overhead in terms of all-to-all communication. (2) Word frequency follows a Zipfian-like distribution. This unbalanced nature of vocabulary-driven routing requires different word experts to process orders of magnitude more tokens than others. We propose a new strategy that overcomes these challenges and allows an efficient implementation of the MoWE layer. Our method contains three main ingredients:

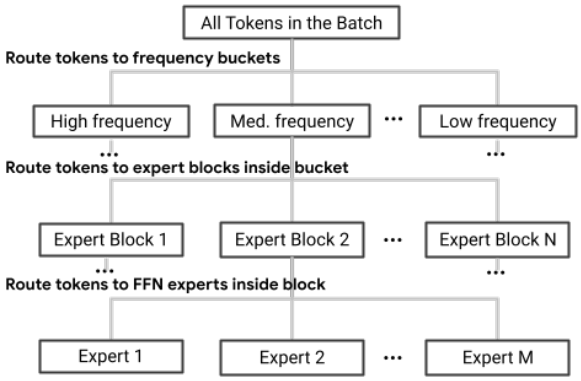

Figure 3: **Hierarchical Routing.** Tokens are first routed to buckets that handle routing ids of similar frequency. Inside each bucket, experts are grouped in blocks, and each token is routed to the block that contains its assigned expert. Inside the block, each token is routed to and processed by an actual expert.

Expert Blocks. We group experts into blocks that are sharded across devices. All-to-all communication is only performed between blocks instead of between experts. Provided we keep the number of expert blocks small enough, we can increase the number of experts without increasing all-to-all communication costs. For example, if we use 128 blocks with 256 experts each, we end up with 32768 experts. We are able to use expert blocks because the fixed routing function pre-defines which block, and which expert inside the block, will process a given token.

Frequency Bucketing. To overcome the unbalanced word frequency distribution, we compute the frequency of words in a sample of 2B tokens from our pretraining data and then split the routing vocabulary into k buckets, where the words in each bucket have approximately the same frequency. Each bucket is then handled by a separate

set of expert blocks. Conceptually, the k MoWE layers are executed in parallel. With this approach, experts in different buckets can have different sizes, different architectures and can support different token capacities (process a diff. number of tokens)[1].

Hierarchical Routing. Given a batch of tokens, the first step is to route them to frequency buckets. Next, inside each bucket, each token is routed to the expert block that contains its assigned expert. Finally, inside the block, each token is routed to and processed by an actual expert. Since routing decisions are based purely on (static) routing ids, token-to-expert assignments are known beforehand and the full path through the hierarchical routing tree becomes trivial. Fig. 3 illustrates this process.

Our proposed strategy allowed us to pretrain MoWE-Base models with up to 1 million (small) experts using 16 v3 TPUs. We did not observe any training instability (e.g. gradient blowup) that are often reported in the pretraining of regular MoE models (Zoph et al., 2022); we suspect is a helpful artifact of our fixed routing scheme.

2.4 Knowledge-Rich Routing Vocabulary

A straightforward strategy to build a large routing vocabulary consists in using the pretraining dataset to train a large vocabulary SentencePiece tokenizer (Kudo and Richardson, 2018). However, initial experiments indicated that this method is suboptimal as many words in the vocabulary turn out to be uninformative – many are just variations of the form of other words. To build a *knowledge-rich routing vocabulary* that contains more *informative* tokens, we derive the vocabulary from a knowledge rich dataset as follows:

(1) Start with the set of all entity and relation names that appears in a Wikidata dump.

(2) Lowercase and split each name using white space and a regex to remove punctuation.

(3) Order tokens based on their frequency in the C4 dataset (Raffel et al., 2020) (version 2.2.0), which is our pretraining dataset.

(4) Select the top 1M tokens to form our routing vocabulary.

This strategy increases the likelihood that the majority of entries in the vocabulary are (single word) names – i.e., terms that we want to store knowledge about. For example, tokenization with a T5.1.1 32K vocabulary breaks down the word "`mathematician`" into 5 tokens ("`math`", "`e`","`m`","`a`", "`tician`"), while our 1M routing vocabulary keeps it as a single token; see also Figure 2. Ideally, the two tokenizations should be aligned as in the figure, but the only hard constraint is that each token from the default tokenization (which defines the input sequence) needs to have a routing id. Appendix D shows more samples of the top words in the routing vocabulary.

Finally, to allow (a) efficient lookup of routing ids and (b) the use of the MoWE layer in auto-regressive scenarios where normally only the initial part of the word is known, we approximate the *routing tokenization* using a hash operation. More specifically, we use the following steps:

- **Offline:** (1) we extend the auxiliary vocabulary by concatenating the default T5 32K vocabulary to it. (2) we tokenize each entry in the auxiliary vocabulary using the default tokenizer and build a hash table where the key is the sequence of (default) token ids and the value is the routing id (a sequential number).

- **Online:** given a tokenized input sequence s composed of n token ids $\{t_1, t_2, ..., t_n\}$, we create the routing id of token t_i by first looking up in the hash-table all sub-sequences $\{t_{i-k}, ..., t_i\}$ for $k \in [0, 8]$, and adopt the routing id of the largest sub-sequence.

3 Experimental Setup

3.1 Tasks and Datasets

We present results on a wide range of NLP tasks. That said, as our main goal is to assess the performance of MoWE on knowledge intensive tasks, we focus our analysis on closed-book question answering tasks: TriviaQA (Joshi et al., 2017), WebQuestions (Berant et al., 2013) and Natural Questions (Kwiatkowski et al., 2019). As in Roberts et al. (2020), our model has no access to external knowledge/text during finetuning and inference. Similar to Lee et al. (2019); Roberts et al. (2020), we perform evaluation by holding out 10% of the training set as a validation set; models are finetuned on the remaining 90% of the data. We also check the performance of MoWE for the claim verification task using the FEVER dataset (Thorne et al., 2018). Finally, to compare our results with regular MoE Transformer models (Lepikhin et al., 2020;

[1] Gale et al. (2023) offers a potential way to avoid some of this bucket, although there likely remains similar effective lower bounds to array and bucket shapes to ensure efficiency.

Model	TriviaQA	WebQuestions	Natural Questions	FEVER	SuperGLUE	Train Time Ratio to T5.1.1-Base
T5.1.1-Base	24.2	28.2	25.7	61.3	77.2	1.0
MoWE-Base	39.4	35.7	29.6	66.3	83.5	2.0
T5.1.1-Large	28.2	29.5	27.3	63.0	85.1	3.1
MoWE-Large	**44.8**	**38.8**	31.9	**68.5**	87.4	4.0
T5.1.1-XL	36.0	32.4	29.5	65.9	88.5	8.6
T5.1.1-XXL	42.9	35.6	**32.8**	67.5	**89.9**	26.4

Table 1: Comparison of MoWE and T5.1.1 models on five different language processing tasks. We use exact match for TriviaQA, WebQuestions and Natural Questions. We use accuracy for FEVER and a blended average of accuracy and F1 scores for the SuperGLUE suite as in (Raffel et al., 2020). T5.1.1. results for TriviaQA, WebQuestions and Natural Questions are from (Roberts et al., 2020). For each model, we also report the training time relative to T5.1.1-Base.; estimated by running each model with a batch size of 256 and input (output) sequence length of 512 (62) on 64 v3 TPUs – the smallest slice that could fit T5-XXL with 256 examples. Note that this likely underestimates the speed of the smaller models, which would enjoy better utilization on fewer devices.

Zoph et al., 2022), we apply MoWE to SuperGLUE benchmark (Wang et al., 2019b). We pretrain all models on the C4 dataset (Raffel et al., 2020).

3.2 MoWE Setup and Hyperparameters.

Following popular (Fedus et al., 2022) and state-of-the-art (Zoph et al., 2022) Transformer-based encoder-decoder MoE models, we use T5.1.1 as the backbone of our MoWE models.

Our main results are from an architecture with four MoWE-layers – two in the encoder and two in the decoder, and each MoWE layer contains 32K experts. We use four MoWE layers as they offer good accuracy without sacrificing computational performance due to routing overhead (see Appendix B.0.1). We place MoWE layers near the middle of the encoder (decoder) to ensure that: (1) the MoWE layers receive a representation of the token that is already somewhat contextualized; (2) after the MoWE layer, there are still multiple Transformer Blocks that can benefit from the output of that layer. Parameters are shared across all MoWE layers with the following goal: (1) it makes the MoWE layer even more similar to a memory that is accessed at different points of the network; (2) we can keep the overall number of sparse parameters relatively low without the need to decrease the total and the size of experts. Additionally, empirical results indicated that sharing parameters across the MoWE layers leads to better performance. The routing vocabulary has 2^{20} ($\sim$1M) entries and was constructed as described in Section 2.4. MoWE-Base and MoWE-Large models have 31B and 45.5B parameters, respectively. See Appendix A for more details.

Pretraining is performed using the same span masking approach used in T5 (Raffel et al., 2020). Following T5 models, our main results use MoWE models pretrained for roughly 1 trillion tokens – 1M steps, with batch size 2048 and input sequence length of 512 tokens; the target sequence length is 114. We use the same pretraining hyperparameters of T5.1.1, and use 64 TPUs v3 for pretraining.

During finetuning for downstream tasks, we freeze all MoWE experts to avoid both overfitting and catastrophic forgetting of knowledge acquired during pretraining (See Appendix B.0.2 for ablations). This is an important distinction to MoE models, which finetune the experts for the downstream tasks. The main hyperparameter that we tune during finetuning is the learning rate. We only use cross-entropy loss; no additional auxiliary losses are used.

4 Experimental Results and Discussion

4.1 Comparison with T5.1.1

In Table 1, we summarize MoWE results on 5 different NLP tasks and alongside T5.1.1 models. MoWE-Base and MoWE-Large outperform T5.1.1-Base and T5.1.1-Large, respectively, on all five tasks. There is a significant gain in performance for knowledge intensive tasks – in particular for TriviaQA, WebQuestions and FEVER. On TriviaQA, MoWE-Base outperforms T5.1.1-Base by 15.2 points in exact match, which corresponds to a 62.8% improvement. On the same dataset, MoWE-Large outperforms T5.1.1-Large by about 16.6 points. Remarkably, MoWE-Base outperforms T5.1.1-XL on all knowledge intensive tasks, while achieving a 4.3x relative training speedup. Similarly, on the same tasks, MoWE-Large outper-

Model	# of params	# of sparse layers	# of expts per layer	Avg. expert MLP dim.	Params shared?	TQA	WQ	NQ	SG
MoE-Top2-Base	2B	12	32	2048	No	26.5	27.7	25.8	80.2
MoWE-Base$_{Light}$	2B	2	8K	141	Yes	**30.4**	**30.8**	**27.1**	**81.0**
MoE-Top2-Base	29.2B	12	512	2048	No	36.2	31.6	28.5	**83.5**
MoWE-Base	31B	4	32K	577	Yes	**39.4**	**35.7**	**29.6**	**83.5**
ST-MoE-L	4.1B	24	32	2816	No	33.8	33.2	29.5	**86.9**
MoWE-Large$_{Light}$	4.1B	2	8K	197	Yes	34.5	34.9	28.9	86.5
MoWE-Large	45B	4	32K	618	Yes	**41.5**	**39.1**	**30.9**	86.3

Table 2: Comparison of MoWE with regular MoE models on TriviaQA (TQA), WebQuestions (WQ), Natural Questions (NQ) and SuperGLUE (SG). MoE-Top2 models are based on the canonical GShard Top-2 MoE Transformer (Lepikhin et al., 2020). ST-MoE-L results are from (Zoph et al., 2022).

forms or has competitive results to T5.1.1-XXL, while achieving a 6.6x relative training speedup.

Despite not being optimized for inference speed, our current implementation of MoWE already offers significant advantages compared to similar-performing models in knowledge intensive tasks. MoWE-Base is approximately 2.7x faster than T5-XL during inference, while MoWE-Large is 6.1x faster than T5-XXL.

4.2 Comparison with Regular MoEs

Table 2 compares MoWE models with two MoE variants: the state-of-the-art ST-MoE-Large (Zoph et al., 2022) and the canonical GShard Top-2 MoE Transformer (Lepikhin et al., 2020). Models in the top (bottom) part of the table use T5-Base (T5-Large) as the backbone, hence they have # FLOPS similar to T5-Base (T5-Large). While the *-Base models were trained for 1M steps, the *-Large models were trained for 500K steps only to match the pretraining steps of ST-MoE-L (Zoph et al., 2022). Table 2 also highlights some architectural differences between MoWE and regular MoEs. The latter uses a larger number of sparse layers, each with a small number of experts and there is no parameter sharing. In both MoE-Top2 and ST-MoE, every other layer is a sparse layer. In MoWE, as experts are tied to the routing vocabulary and we want to encourage expert specialization, we use a large number of experts whose params are shared across a small number of sparse layers. Sharing expert parameters across the MoWE layers allows the use of a large number of experts without exploding the total number of parameters.

In Table 2, we include results for *light* versions of MoWE Base and Large that contains 2B and 4.1B parameters, respectively. MoWE-*$_{Light}$ uses 8K (smaller) experts instead of 32K; see Appendix A.1 for details. Note that this is not the ideal setup for MoWE because each expert is shared by a large number of token ids and the average expert size is also smaller. Nevertheless, MoWE-Base$_{Light}$ outperforms MoE-Top2 (2B) in all four tasks, and MoWE-Large$_{Light}$ outperforms or achieves similar performance to ST-MoE-L in all tasks. Compared to the MoE baselines in Table 2, MoWE-Base and MoWE-Large perform significantly better on the knowledge intensive tasks, and achieve comparable performance on SuperGLUE.

4.3 The MoWE Layer is a Sparse Memory

We perform an experiment to assess to what extent a MoWE model relies on the MoWE layer to perform the TriviaQA task. In particular, we are interested in measuring the impact of not processing (*skipping*) relevant words in the MoWE layer when the model is generating the answer. We finetuned a MoWE model on TriviaQA and included the condition that tokens with routing ids >32K are skipped by the MoWE layer during finetuning and inference. This is equivalent to use the skip connection to generate the output for those tokens in the MoWE layer. We set the threshold to 32K because the first 32K routing ids roughly correspond to frequent and less knowledge-driven tokens that resulted from concatenating the default vocabulary to the auxiliary one (see Section 2.4).

Selectively Skips some Question Tokens in the MoWE Layer?	TriviaQA EM
No	**35.1**
Yes	25.6

Table 3: Effect on TriviaQA EM of skipping (not processing) tokens with routing id >32K in MoWE layer.

In Table 3, we show that there is a significant drop of 9 points in EM when tokens with routing id >32K are skipped in the MoWE Layer. This

Question	Skips highlighted words in the MoWE layer when generating the answer? No	Yes
What is the name of **Adele**'s first album?	19	Addiction
Who followed William **Taft** as US President?	Woodrow Wilson	James Garfield
Quinsy affects which part of the human body?	Tonsils	Feet
What country will host the **2022** FIFA World Cup competition?	Qatar	Brazil
What is **Neptune**'s main satellite?	Triton	Uranus
What was the first name of Italian **statesman** and writer **Machiavelli**?	Niccolo	Francois
Almeria, **Merlot**, and **Waltham** Cross are which fruit?	Grapes	Apple

Table 4: Example TriviaQA questions and their respective answers from two configurations of a pretrained MoWE-Base model depending on whether we skip (not process) the highlighted words in the MoWE layer. The answer generated by the model can change completely (from correct to incorrect in these cases) by simply not processing a single relevant word in the MoWe layer. In this experiment, the MoWE model has a single MoWE layer that is located in the encoder and contains 32K experts.

result indicates that MoWE models rely heavily on the experts of words that are in our knowledge-rich vocabulary. In Table 4, we show selected examples of questions and their respective answers for two setups: skipping or not highlighted tokens. It is remarkable that not using a single MoWE expert makes the model answer the question in a completely different way. For the MoWE model used in this experiment, a single expert represents only 0.33% of the estimated total number of activated parameters. Note that, because the MoWE layer is frozen during finetuning, all the knowledge that is being leveraged in the downstream task comes from the pretraining corpus. These results suggest that (at least part of) the pretraining world knowledge needed to answer some questions is stored in the deactivated experts.

4.4 Comparison with Memory Augmented models

In this section we compare the performance of MoWE with recently proposed memory augmented models: Entities as Experts (EaE) (Févry et al., 2020) and Transformer Over Mention Encodings (TOME) (de Jong et al., 2022) on two knowledge intensive tasks. These models were pretrained on Wikipedia data using entity aware losses, and their memory component focus primarily on that domain. To make MoWE models a little more specialized on Wikipedia domain, which is known to benefit tasks such as TriviaQA, we followed (Roberts et al., 2020) and used the Salient Span Masking (SSM) data from (Guu et al., 2020) to perform an additional number of 40K pretraining steps.

We summarize the experimental results in Table 5[2]. MoWE-Base model outperform EaE on both datasets. MoWE-Large model outperforms both baselines on FEVER and has similar or competitive performance to TOME models on TriviaQA.

Model	TQA	FEVER
EaE	43.2	66.1 / 63.6
TOME 1	50.8	70.5 / 67.8
TOME 2	**54.6**	**71.1** / 68.1
MoWE-Base + SSM	44.9	69.1 / 66.9
MoWE-Large + SSM	**50.2**	70.5 / **68.7**

Table 5: Comparison of MoWE with EaE and TOME. Results for both models are from (de Jong et al., 2022). Results for TQA are dev, while FEVER is dev/test. TOME 1 uses two mem. layers and TOME 2 uses two.

EaE and TOME models are arguably more customized solutions to these tasks. For example, EaE and TOME tackle TriviaQA as an entity linking task, where a closed set of 1M Wikipedia entities is used for ranking. In contrast, MoWE performs open-ended answer generation, which is more flexible but also more challenging. Additionally, both EaE and TOME use specialized training procedures, including adding additional loss functions and entity or noun phrase chunking, and require k-nn tools to search relevant embeddings in their memory. In MoWE models, the "sparse memory" is integrated into the model backbone and accessed seamlessly as any other model parameter. As a consequence, MoWE can be trained in a similar fashion to a T5 model with no external tools/models.

4.5 Ablation Study

In this section we present ablation experiments on different architectural choices of MoWE. Addi-

[2]For TriviaQA, we report results for the validation set only because the server used to score the test set is no longer active.

tional ablations can be found in Appendix B.

4.5.1 Effectiveness of Knowledge-Driven Routing Vocabularies

In this section, we show evidence to support our conjecture that routing with large knowledge-rich vocabularies leads to better performance by varying the size of the routing vocabulary. For the experiments in this section we use a *baseline* MoWE model configuration with a fixed T51.1-Base backbone with 32K experts, yielding 15.5B sparse parameters. For vocabularies smaller than 1M, we use the top-K words (by frequency in C4 dataset) from our 1M routing vocabulary described in Section 2.4. We report results mainly on the TriviaQA and Natural Questions datasets and we use F1 metric instead of exact match because it is slightly less noisy and highlights the trends more clearly.

Figure 4 shows that results progressively improve as we increase the routing vocabulary. These improvements are more pronounced when training for longer; see Figure 5. As we increase the size of the routing vocabulary, we increase the lexical-based inductive bias injected in the model via the routing function. For TriviaQA, there is an improvement of ∼2 points in F1 when using routing vocabularies with size above 262K. See Appendix B for additional ablation experiments on the number of experts used.

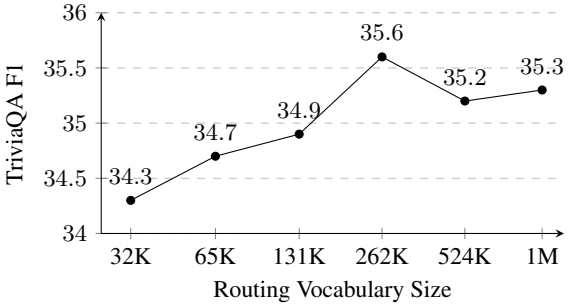

Figure 4: Performance on TriviaQA with different routing vocabulary sizes. These models are pretrained for 200K training steps.

4.5.2 Effect of Number of Experts

We present two additional experiments on how the number of experts affect MoWE performance. First, we check the impact of varying the number of experts between 16K, 32K and 64K while keeping fixed the routing vocabulary to 1M size and the model size to 15.5B. In all experiments in this section, we pretrain the models for 200K steps. In Fig. 6 we see that 32K experts seems to be a sweet spot in terms of number of experts for MoWE. Using a larger number of smaller experts is preferable because is is more memory efficient and also speeds up our lookup table implementation of Expert Blocks in frequency bucket 4.

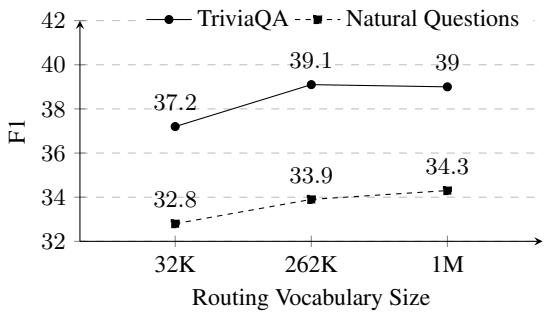

Figure 5: Performance on TriviaQA and Natural Questions with different routing vocabulary sizes. These models are pretrained for 1M training steps (longer than Figure 4).

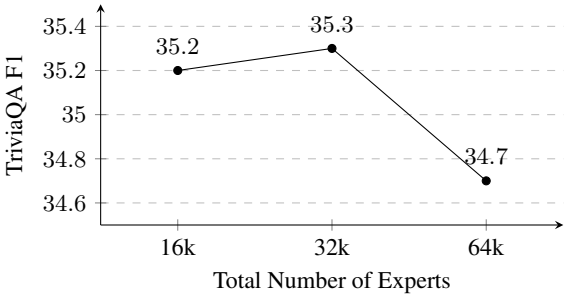

Figure 6: Performance on TriviaQA of different MoWE-baseline models where we fix the routing vocabulary to 1M size and vary the number of experts.

Next, we evaluate MoWE performance when we increase the number of experts to match the size of large routing vocabularies. We keep the total number of sparse parameters fixed by decreasing the size of the experts in each experiment. Therefore, when using 1M experts, the MLP dim of each expert is 8, while the MLP proj dimension when using 32K experts is 256. To the best of our knowledge, this is the first time that a Transformer-based MoE model is trained with up to a million experts. In Fig. 7 we show results for increasing MoWE-baseline for up to 1M experts. We see a progressive degradation in performance when matching the number of experts to the size of the vocabulary. We believe this is mainly due to two factors: (1) the number of training updates that each experse receive becomes increasingly sparse; (2) the size of the experts are decreased.

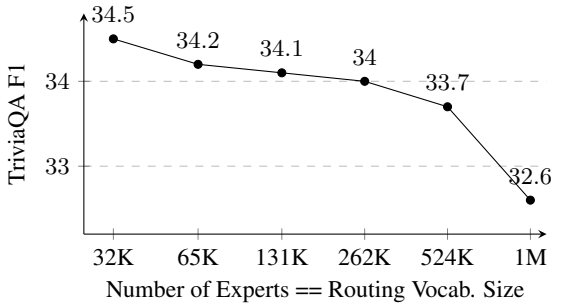

Figure 7: Performance on TriviaQA of different MoWE-baseline models where the number of experts match the routing vocabulary.

5 Related Work

Sparsely-activated Mixture-of-Experts (MoE) models (Shazeer et al., 2017) increase parameter count with sublinear increases in computation cost (FLOPs) by sparsely activating modules ("experts"). Recently, Transformer-based MoE models have achieved state-of-the-art performance and efficiency wins in language (Lepikhin et al., 2020; Fedus et al., 2022; Du et al., 2022; Artetxe et al., 2021; Zoph et al., 2022), vision (Riquelme et al., 2021) and multimodal (Mustafa et al., 2022).

In contrast to the aforementioned MoE models, MoWE uses tens of thousands of experts; Du et al. (2022), for example, found diminishing performance in their MoE models beyond roughly 64 or 128 experts. To support more experts, MoWE uses a fixed routing scheme, unlike vanilla models which all rely on learned top-k routing mechanisms to assign tokens $\rightarrow$ experts, or Zhou et al. (2022) who use learned top-k expert $\rightarrow$ token assignments. The MoWE routing function assigns tokens to individual experts based on their token id in an auxiliary vocabulary. This is reminiscent of Hash Layers (Roller et al., 2021), which assigns tokens to experts based on a fixed hash bucketing, with the difference that many different token ids, based on the *embedding vocabulary*, are bucketed together and assigned to individual experts. As a further consequence of the increased number of experts, we freeze the MoWE experts during finetuning to avoid both overfitting and catastrophic forgetting of knowledge acquired during pretraining.

In standard SPMD MoE implementations, experts have fixed capacity buffers and can therefore only process a fixed fraction of the input tokens, so most top-k routing models invoke an auxiliary load balancing loss (Shazeer et al., 2017) to encourage even distribution of tokens across experts. Because routing is fixed, MoWE expert capacity buffers can be sized according to expected token frequency. Recent work, such as Gale et al. (2023) relaxes expert buffer constraints with variable expert buffer "blocks".

MoWE models bridge the gap between MoE models and **Memory augmented models**, such as Mention Memory (de Jong et al., 2022), FILM (Verga et al., 2021), Entities as Experts (Févry et al., 2020) and Knowledge Prompts (dos Santos et al., 2022), which call a memory bank when processing inputs. Memory models have proven effective in knowledge intensive tasks but can have few drawbacks: (1) They typically require a specialized training procedure, that differ from dense models, in order to effectively learn to use the "external" memory. (2) Training data is normally very domain specific (most cases focus on Wikipedia) and, as a result, each models can only be applied to tasks that benefit from that data.

On the other hand, MoWE is simple to train – no additional losses and no need to learn to search the memory. It seamlessly integrates with the model as there is no need to perform search using a nearest neighbor style tool during inference or training; the predefined routing avoids this search altogether. MoWE models can be trained on generic pretraining data (C4 in our case). The link between memory augmented and MoWE models, is that the entities are encoded into the model when identified with particular experts. However, unlike memory models, the experts/entities are small neural networks rather than embeddings.

6 Conclusions

We presented MoWE, a novel neural net architecture effectively bridging the efficiency of sparse MoE models with the knowledge retrieval capabilities of memory-augmented models. By leveraging a large, knowledge-rich vocabulary for routing and employing tens of thousands of word-specific experts, MoWE demonstrates significant performance gains on knowledge-intensive tasks. Our findings highlight the effectiveness of lexical-driven routing and the potential of word experts as a form of sparse, integrated memory within language models, opening doors for future research. Future investigations could analyze the properties of word experts, explore integration within decoder-only models, and extend MoWE to other modalities and languages.

7 Limitations

Due to the limited computational budget allocated for this project, we were not able to test MoWE models beyond the Large size (880M dense parameters; 45.5B total parameters). Nevertheless, we believe that the trend in terms of improvement from MoWE-large to MoWE-XL should be consistent with the improvements with see from T5-Large to T5-XL.

We have not explored potential properties that might emerge in the word experts. For example, are experts of related words somewhat similar? Can we use a matrix similarity metric to find relevant nearest neighbors of a given expert? We believe this is an interesting research direction and leave it to future work.

We have not explored the use of MoWE-layers in decoder-only models. We focused on T5-based models for ease of comparison with popular (base/large) MoE models (Lepikhin et al., 2020; Zoph et al., 2022).

Our experiments are limited to NLP tasks, and English datasets. Given the successful trend of Transformers in vision (Dosovitskiy et al., 2020) and other modalities, and also across multiple languages (e.g. (Xue et al., 2020)), we believe our results will extend across language and other modalities.

References

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3554–3565, Online. Association for Computational Linguistics.

Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, et al. 2021. Efficient large scale language modeling with mixtures of experts. *arXiv preprint arXiv:2112.10684*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. JAX: composable transformations of Python+NumPy programs.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

Michiel de Jong, Yury Zemlyanskiy, Nicholas FitzGerald, Fei Sha, and William W. Cohen. 2022. Mention memory: incorporating textual knowledge into transformers through entity mention attention. In *International Conference on Learning Representations*.

Cicero Nogueira dos Santos, Zhe Dong, Daniel Cer, John Nham, Siamak Shakeri, Jianmo Ni, and Yun hsuan Sung. 2022. Knowledge prompts: Injecting world knowledge into language models through soft prompts.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias

Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity.

Thibault Févry, Livio Baldini Soares, Nicholas FitzGerald, Eunsol Choi, and Tom Kwiatkowski. 2020. Entities as experts: Sparse memory access with entity supervision. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4937–4951, Online. Association for Computational Linguistics.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. 2023. Megablocks: Efficient sparse training with mixture-of-experts. *Proceedings of Machine Learning and Systems*, 5.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*.

Basil Mustafa, Carlos Riquelme, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. 2022. Multimodal contrastive learning with limoe: the language-image mixture of experts. *arXiv preprint arXiv:2206.02770*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. 2021. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595.

Adam Roberts, Hyung Won Chung, Anselm Levskaya, Gaurav Mishra, James Bradbury, Daniel Andor, Sharan Narang, Brian Lester, Colin Gaffney, Afroz Mohiuddin, et al. 2022. Scaling up models and data with t5x and seqio. *arXiv preprint arXiv:2203.17189*, 13.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. 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. Association for Computational Linguistics.

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason E Weston. 2021. Hash layers for large sparse models. In *Advances in Neural Information Processing Systems*.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Pat Verga, Haitian Sun, Livio Baldini Soares, and William Cohen. 2021. Adaptable and interpretable neural MemoryOver symbolic knowledge. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3678–3691, Online. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. *SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems*. Curran Associates Inc., Red Hook, NY, USA.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. Byt5: Towards a token-free future with pre-trained byte-to-byte models.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. MoEfication: Transformer feed-forward layers are mixtures of experts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 877–890, Dublin, Ireland. Association for Computational Linguistics.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. 2022. Mixture-of-experts with expert choice routing.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. St-moe: Designing stable and transferable sparse expert models.

A MoWE Setup

Our main experiments on MoWE-Base and MoWE-Large use an architecture with four MoWE-layers in total. There are two in the encoder and two in the decoder, and parameters are shared across all MoWE layers. Those layers are placed at Transformer blocks 5 and 10 in the Base model, and at blocks 9 and 17 in the Large model. We placed MoWE layers towards the middle of the encoder (decoder) because: (1) they receive a representation of the token that is already somewhat contextualized; (2) after the MoWE layer, there are still multiple Transformer Blocks that can benefit from the output of that layer. Unless otherwise informed, each MoWE layer contains 32K experts and the routing vocabulary has ~1M entries. The routing vocabulary used in our experiments was derived from the Wikidata dump previously used by Agarwal et al. (2021).

Our current implementation of MoWE was coded in Jax (Bradbury et al., 2018) on top of the T5X (Roberts et al., 2022) framework. [3]

Some additional configurations are provided in the following sections.

A.1 Configuration of Frequency Buckets, Expert Blocks and Experts

We split the vocabulary into four different frequency buckets. Token frequency was computed using a sample from our pretraining dataset. The MoWE layer does not process the top 16 most frequent tokens in the routing vocabulary, i.e. those tokens ids are never routed to an expert. These tokens are punctuation marks and other non-content words and we estimate they can represent up to

[3] https://github.com/google-research/t5x

Configuration	**Bucket 1**	**Bucket 2**	**Bucket 3**	**Bucket 4**
Num. of routing ids covered	128	880	1024	$\sim 2^{20}$
Num. of Expert Blocks	128	128	128	128
Num. of Experts per Block	1	7	8	235
Total Num. of Experts	128	896	1024	30080
Expert MLP dimension (Base)	2048	2048	1024	512
Expert MLP dimension (Large)	2816	2816	1536	512

Table 6: Configuration of experts and blocks in the four frequency buckets used in MoWE-Base (31B) and MoWE-Large (45B). A routing vocabulary of 1M token ids is considered.

Configuration	**Bucket 1**	**Bucket 2**	**Bucket 3**	**Bucket 4**
Num. of routing ids covered	128	880	1024	$\sim 2^{20}$
Num. of Expert Blocks	64	64	64	64
Num. of Experts per Block	1	1	1	128
Total Num. of Experts	64	64	64	8192
Expert MLP dimension (Base)	2048	2048	2048	96
Expert MLP dimension (Large)	2816	2816	2816	136

Table 7: Configuration of experts and blocks in the four frequency buckets used in the MoWE-Base$_{Light}$ and MoWE-Large$_{Light}$ that we refer in Section 4.2.

28% of the tokens in a batch. This speeds up the training time and does not hurt downstream performance, as these tokens are not content words. The configuration of the four frequency buckets is described in Table 6. Using this configuration, we get a model with ~31B parameters in the case of the Base model and ~45.5B sparse parameters in the case of the Large model. The difference in the number of parameters is due to the use of different MLP projection dimensions (see Table 6) and the token embedding size, which is 768 in Base and 1024 in Large.

Notice in Table 6 that for buckets 1 to 3 we use one expert per token. In this configuration, in bucket 4 the experts are shared for multiple tokens. This bucket contains mainly low frequency tokens, which are the majority in the vocabulary. Additionally, due to the large number of experts in this bucket, the Expert Blocks are implemented as lookup tables. Although we believe the current configuration is not optimal and can be improved, it already produces efficient models.

In Table 7, we detail the configuration of the four frequency buckets and respective expert number and sizes for the MoWE-Base$_{Light}$ and MoWE-Large$_{Light}$ that we refer in Section 4.2.

A.2 Additional Hyperparameters

For pretraining MoWE models, we used the default T5X hyperparamters for T5.1.1. Unless otherwise mentioned, pretraining is performed for roughly 1 trillion tokens – 1M steps, with batch size 2048 and input sequence length of 512 tokens; the target sequence length is 114.

For downstream task we normally use batch sizes of 256 or 512. For most datasets, a learning rate of 2e-4 and dropout rate of 0.05 gave the best results. The main exception is SuperGLUE and Fever datasets, which work better with LRs between 1e-3 and 5e-4.

B Additional Ablation Experiments

In this section we present additional ablation experiments on different architectural choices of MoWE. In all experiments, we pretrain the models for 200K steps.

B.0.1 Impact of the number of MoWE Layers

In Table 8, we show the impact of using a different number of MoWE-Layers in encoder and decoder. All models were trained for 200K steps. We can see in Table 8 that going from one to two layers in the encoder gives a significant gain in EM (31.0 -> 31.6). However, going from 2 to 3 layers does not give improvements on EM. Adding MoWE layers to the decoder improves the performance, specially when using 2 layers in the encoder.

In Table 9 we show results on using different expert sizes in each of the four bucket sizes. A single MoWE layer is used, and it is located in the encoder. We start with a configuration where the experts in Bucket 1 has experts with MLP dimension 512, and sequentially half the value for the

# MoWE Layers Encoder	Decoder	EM	F1
1	0	31.0	36.3
2	0	31.6	36.9
3	0	31.5	37.1
1	1	31.4	36.7
2	1	32.4	37.5
2	2	33.1	38.4

Table 8: Impact of the number of MoWE layers on TriviaQA for Base model. Expert parameters are shared across MoWE layers.

next consecutive bucket. This results in a model with 3.9B sparse params, whose performance on TriviaQA is presented in the firs row of Table 9. The the following rows, we consecutively double the size of the expert in each bucket, which doubles the total number of sparse parameters. There is a consistent improvement of 1 point in EM when doubling the model size. We believe the increase would be larger if we pretrained the model for 1M steps instead of 200K steps.

# sparse params	MLP Dim. of Experts in each Frequency Bucket				EM	F1
	B1	B2	B3	B4		
3.9B	512	256	128	64	28.5	33.7
7.8B	1024	512	256	128	29.6	34.8
15.5B	2048	1024	512	256	30.0	35.3
31.0B	2048	2048	1024	512	31.0	36.3

Table 9: Impact on TriviaQA EM and F1 of using different expert sizes in the four different buckets.

B.0.2 Freezing vs Unfreezing Experts During Finetuning

MoWE-Base on TriviaQA gets EM of 37.7 when freezing the experts during finetuning. When we allow the update of experts during finetuning, EM drops by 5 points to 33.5.

C Metrics and Baseline Setup

We use the following metrics in our experiments: for TriviaQA, WebQuestions and Natural Question we mostly report results in terms of Exact Match (EM), except for some ablation experiments, where we report results in terms of F1. For Fever dataset, we report the accuracy in both validation and test sets. For SuperGLUE, following previous works (Raffel et al., 2020; Xue et al., 2022), we finetune MoWE models on a mixture of all tasks in the benchmark, select the best result per task and present the average validation set scores over all tasks.

We use the MoE-Top2 implementation from T5X framework in our comparative experiments. Dense and sparse layers are interleaved, which results in a total of 12 sparse layers: 6 in the encoder and 6 in the decoder. We use Top-2 routing and most hyperparameters are default, except for expert dropout (0.3) and learning rate during finetuning, which we set to 5e-4 for QA tasks and Fever. For SuperGLUE, we follow the recomendation from ST-MoE paper and used a larger learning rate (1e-3) and small batch size (256), except for the model with 512 experts, for which we used batch size of 512. https://github.com/google-research/t5x/

D Example of Entries from Knowledge Rich Vocabulary

Top 50 word, by frequency in C4, in the routing vocabulary: *'isn', 'aren', '...', '3d', '1st', 'whilst', 'copyright', 'creates', '2nd', 'tells', 'adds', 'wet', '3rd', '·', 'likes', 'filling', 'yours', 'ˆ', 'accordance', '4th', 'amongst', 'sees', '20th', 'mp3', '5th', 'woods', '19th', 'tx', 'toy', 'solely', 'thinks', '21st', 'sits', 'asks', '10th', 'receives', 'worlds', '6th', 'singles', 'blues', 'tops', 'inn', 'lean', 'mills', '7th', 'ranges', 'bears', 'newer', '8th', 'node'*.

In the top 50 words by frequency, we still see many words that are variations of common words, like *"sees"*. However, the quality of the vocabulary improves significantly later in the rank. For instance, this are the top 50 after position 6000 of 1M: *'consignment', 'billboards', 'primal', 'discrepancy', 'callback', 'freeware', 'horticulture', 'jb', 's8', 'aspirants', 'commemorative', 'brisk', 'arched', 'pondering', 'fluff', 'diwali', 'landline', 'wilder', 'apocalyptic', 'patchwork', 'airs', 'stagnant', '412', 'watery', 'hospitalization', 'mccoy', 'serbian', 'paprika', 'headsets', 'deserts', 'pulley', 'orthopaedic', 'disparity', 'egyptians', 'painfully', 'kenyan', 'bale', 'condemnation', 'deportation', 'incline', 'perfumes', 'undergraduates', 'favoured', 'pvp', 'bbb', 'lyons', 'fremont', 'eurozone', 'afl', 'monogram'*.

More work can definitely be done to improve the routing vocabulary, but we wanted to keep it simple for our experiments.