We thank the reviewers for their thorough and insightful reviews. We first respond to a common question from Reviewers 3 and 5 regarding the information stored in the memory / memory access patterns, then address each review in turn:

Memory access (Reviewer 3, Reviewer 5). We tried to investigate how the memory is used, and whether we can find interesting memory access patterns. For a given memory value index, we looked at the n-grams that were the most responsible for accessing this memory index. We found that for some memory indices, the associated n-grams were nicely related to a very specific topic. However, for other memory indices (in particular, the most frequently accessed ones), the pattern of the associated n-grams was not as clear. We need to investigate more to understand how exactly the model uses the memory.

Reviewer 2

- For the uniformity of queries, we experimented with the Kozachenko Leonenoko estimator as a loss term to favor high-entropy distribution, but we observed that it made little difference in practice. We hypothesize that a uniform distribution is a difficult target for a distribution that approximately follows a Zipf law. We thank the reviewer for suggesting Rae et al., we will incorporate it into our related work section. They propose to reallocate rarely used memory slots, to improve memory coverage. In contrast, our product key set enables to have a very good coverage of the memory and avoid this issue.
- Chen et al. compare the theoretical aspects of the Transformer and the RNN. In RNNs they argue that the memory is the hidden state, that is context-dependent. In our work, the memory consists of a set of (context-independent) parameters. The memory is of course accessed in an input specific way, but it is static unlike in RNNs.
- The keys are determined by the product set, and the values are an embedding table. Parameters of both are learned jointly with the rest of the network. The queries are computed as in a regular transformer; we will update the paper to make it clearer.
- Our multi-head mechanism is akin to the multi-head used in the attention layer, but it is used in the memory layer. We will fix the terminology to clarify this point.

Reviewer 3

- Our experiments are by design very large-scale and all take a large amount of time and computational power to converge (several days on 32 GPUs). Running each of them several times would be extremely expensive. However, our observation is that these experiments are overall very stable. For some particular model configurations, we actually ran several experiments with different random initializations and found that the variance was overall extremely small across runs (with differences typically smaller than 0.1 perplexity).
- We performed similar experiments (with / without memory) with the training objective presented in BERT (masked language modeling). Our findings were very similar to the ones presented in the paper (with regular language modeling): a model with 12 layers and a memory outperforms a model with 24 layers without memory, both for models of dimensionality 1024 and 1600.
- We opted for one task in order to retain the extensive empirical analysis and rigorous methodology. In our upcoming work, we are now exploring this layer for computer vision applications.

Reviewer 5

- The described layer is indeed not fully differentiable because there is a discontinuity when there is a switch between the k-th and (k+1)-th nearest neighbor, and "Ensuring that the k-th attention weight is zero" as suggested is a way to address this issue. This is something we actually tried, we had a hyper-parameter to remove the k-th weight to all weights (so that each weight remains positive, and the k-th one is zero) and added an extra L1-normalization step on the updated weights (to ensure they still sum to 1). However, in practice we found that the k-th weight was always very small anyway (the smallest value of a softmax over typically 32 k-NN scores), and we did not observe any difference of results by fixing this score to exactly zero.
- Our comment regarding the priority list was indeed unclear, we will clarify it in the paper.
- In Figure 1, we plot the log-probability for different different groups of words clustered by frequency. We observe that adding the memory improves the performance on all words, particularly on rare words, which is what could be expected: the memory is useful to store rare facts or rare named entities that are usually difficult for the model to retrieve. These observations are similar to what we observed by adding more layers, but no memory. We thank the reviewer for suggesting this.

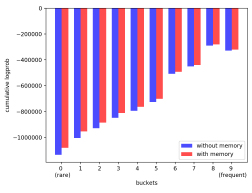

Figure 1 (zoom in for details): Cumulative log-probability (higher is better) for words with different frequency, for two models with/without memory, but otherwise identical configuration. Improvements in log-probability when adding a memory layer are higher for rare words (left) than for frequent words (right).