Large Memory Layers with Product Keys

Akhmad Sumekenov

Higher School of Economics

14 марта 2020 г.

Overview

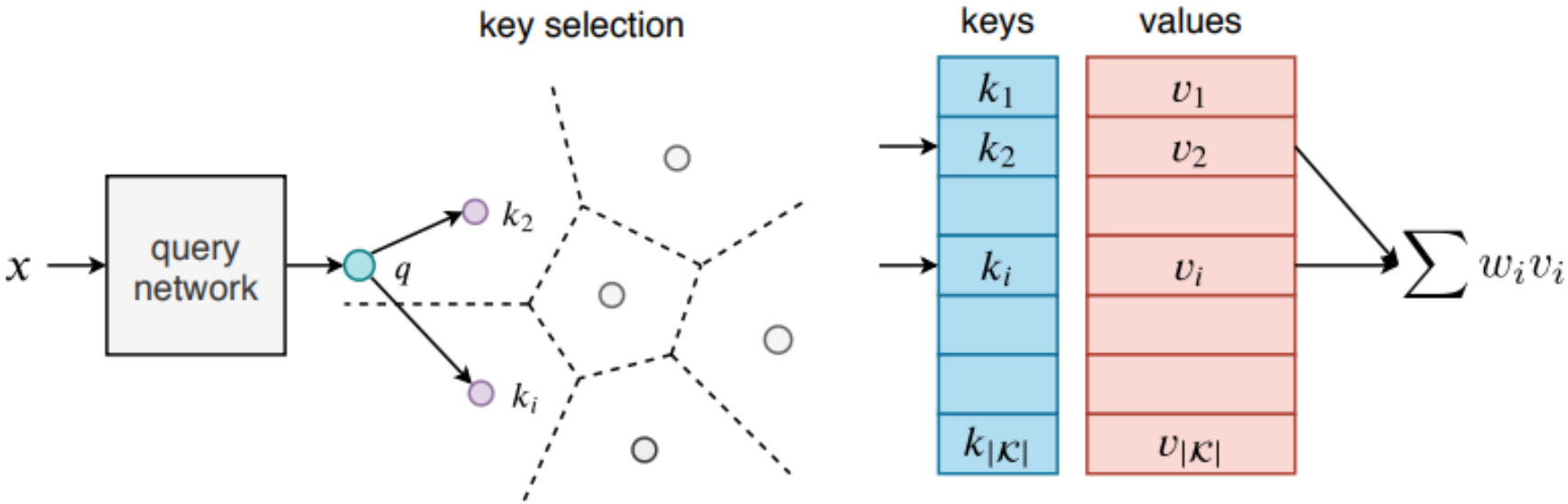
1. Зачем?
2. Как устроен обычный key-value memory layer
3. Product key-value memory layer
4. Query network
5. Standard Key Assignment and weighting
6. Product key set и sub-keys
7. Complexity
8. Эксперименты
9. Влияние частей PKVML на результаты экспериментов
10. Заключение

Зачем?

- Использовать память - хорошое улучшение trade-off между перфомансом модели и ее размером
- Например, memory-augmented трансформер с 12 слоями имеет такой же перфоманс как и 24-слойный, при этом inference time в два раза меньше
- Если не верите - авторы выложили код в открытый доступ.

Product key-value memory layer

Classic Product key-value memory layer



Query Network

- q - по сути обычный Multi-Layer Perceptron, то есть небольшая полносвязная сетка

$$q : x \mapsto q(x) \in \mathbb{R}^{d_{\mathbf{q}}}$$

- Обычно $d_{\mathbf{q}} = 512$

Standard key assignment and weighting

- $\mathcal{T}_k$ - top-k оператор (это что?)

$$t_{i_1} \geq t_{i_2} \geq \cdots \geq t_{i_n} \mapsto \mathcal{T}_k(t_1, \ldots, t_n) = \{i_1, \ldots, i_k\}$$

- Из $\mathcal{K} = \{k_1, \ldots, k_{|\kappa|}\}$ берем топ-k ключей:
  $\mathcal{I} = \mathcal{T}_k\left(q(x)^T k_i\right)$ Get k nearest neighbors
  $w = \text{Softmax}\left(\left(q(x)^T k_i\right)_{i \in \mathcal{I}}\right)$ Normalize top-k scores
  $m(x) = \sum_{i \in \mathcal{I}} w_i v_i$ Aggregate selected values

Product key set и sub-keys

- Хотя второй и третий шаги относительно быстрые в вычислении, первый шаг требует вычисления всех попарных произведений x с $\mathcal{K}$
- Что такое product key?

$$\mathcal{K} = \left\{ \left(c, c' \right) | c \in \mathcal{C}, c' \in \mathcal{C}' \right\}$$

- Общее количество ключей

$$|\mathcal{K}| = |\mathcal{C}| \times \left| \mathcal{C}' \right|$$

- $\mathcal{C}$ и $\mathcal{C}'$ это subkeys размера $d_{\mathbf{q}}$. Разделим query q на q_1 и q_2, и далее:

$$\mathcal{I}_{\mathcal{C}} = \mathcal{T}_k \left(\left(q_1(x)^T c_i \right)_{i \in \{1 \ldots |C|\}} \right), \quad \mathcal{I}_{\mathcal{C}'} = \mathcal{T}_k \left(\left(q_2(x)^T c'_j \right)_{j \in \{1 \ldots |\mathcal{C}'|\}} \right)$$

- Гарантировано, что среди этих k^2 ключей будет наш топ-k

Product key set и sub-keys

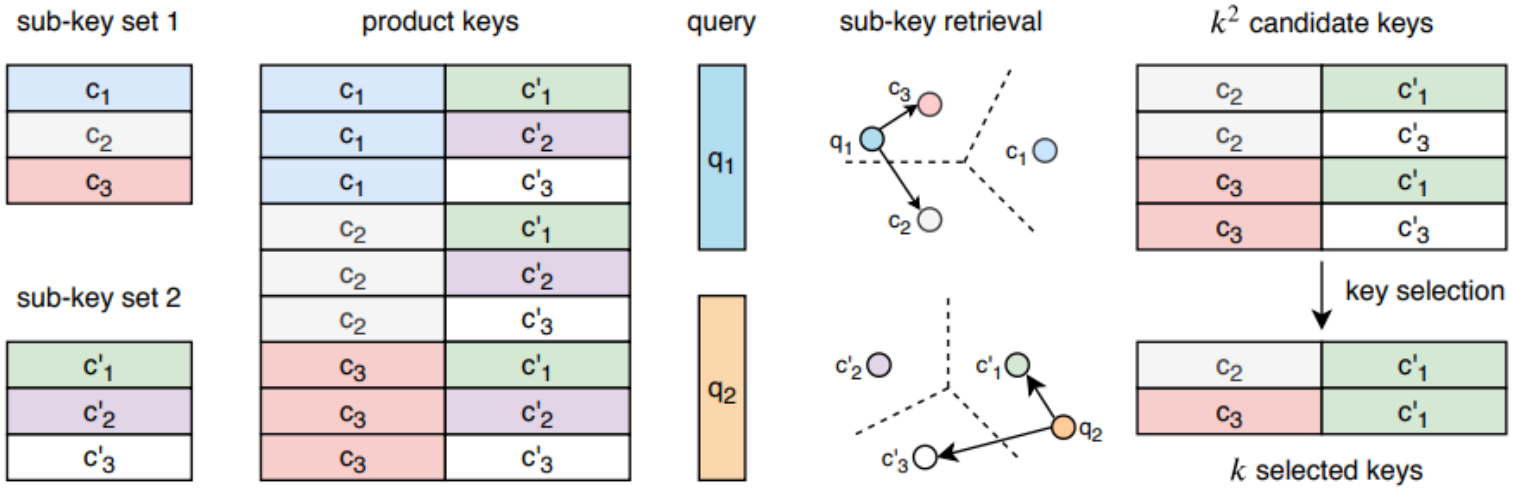


Figure 2: **Illustration of the product keys.** We define two discrete subsets of keys (sub-key set 1 and sub-key set 2). They induce a much larger set of keys, which are never made explicit (product keys). Given a query, we split it into two sub-queries (q_1 and q_2). Selecting the k closest keys ($k = 2$ in the figure) in each subset implicitly selects $k \times k$ keys. The k keys maximizing the inner product with the query are guaranteed to belong to this subset, on which the search can be done efficiently.

Product key set и sub-keys

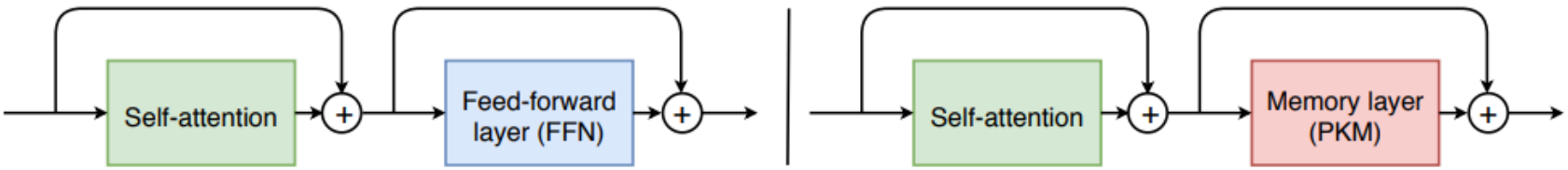


Figure 3: **Left:** A typical transformer block is composed by a self-attention layer followed by an FFN layer (a two layer network). **Right:** In our system, we replace the FFN layer with a product key memory layer, which is analogous to a sparse FFN layer with a very large hidden state. In practice, we only replace the FFN layer in N layers, where typically $N \in \{0, 1, 2\}$.

Complexity

- В классическом случае:

$$\mathcal{O}\left(|\mathcal{K}| \times d_{\mathrm{q}}\right)$$

- В случае product keys:

$$\mathcal{O}\left(\left(\sqrt{|\mathcal{K}|} + k^2\right) \times d_{\mathrm{q}}\right)$$

- Почему?

Multi-head

- Multi-head query : вход идет на несколько query network
- Выход - сумма по всем:

$$m(x) = \sum_{i=1}^{H} m_i(x)$$

Experiments

- Метрика - perplexity
- Метрики для использования : доля использованных значений: $\#\{z_i \neq 0\}$ и KL между распределением использованных ключей и равномерным распределением : $\log(|\mathcal{K}|) + \sum z_i \log(z_i)$
- Датасет - public Common Crawl
- Модель - Transformer с 12, 16 и 24 слоями.

Experiments

Dimension	1024				1600	
N memories	0	1	2	3	0	1
12 layers	17.7	15.6	14.8	14.5	15.0	13.7
16 layers	16.7	14.9	**14.1**	-	14.4	**13.2**
24 layers	16.0	14.6	-	-	14.0	-

Experiments

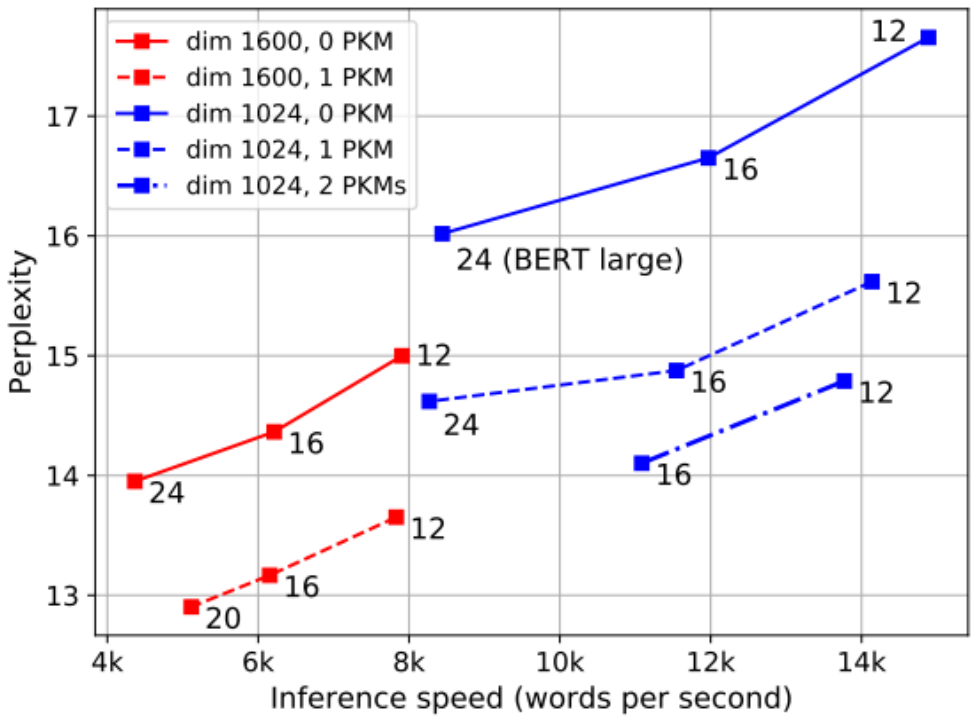


Figure 4: **Trade-off between speed and perplexity on the test set.** Labels on the graph represent the number of layers. Adding memory layers significantly improves the performance and has a negligible impact on the inference speed. Models with 12 layers and a Product Key Memory (PKM) outperform 24-layer models of the same dimension, while being almost twice faster at inference. In particular, a 12-layer model of dimension 1024 with a memory outperforms a model of 24 layers of the same dimension (same configuration as BERT large).

Влияние частей PKVML на результаты экспериментов

Table 2: **Perplexity and memory usage for different memory sizes, with and without BatchNorm.** Adding a batch normalization layer in the query network encourages the model to use more keys. This is not necessary for small memories of size 16k and 65k where the usage is already close to 100% without batch normalization, but for memories of size 147k of more, batch normalization improves the memory usage significantly, along with the perplexity.

Memory size	16k		65k		147k		262k		590k		1M	
BatchNorm	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes
Perplexity	22.8	23.0	21.7	21.9	20.9	20.7	20.5	19.8	20.0	18.7	19.8	**18.0**
Usage (%)	**100**	**100**	99.0	**100.0**	83.8	99.6	64.4	97.9	38.0	90.3	25.8	80.3
KL	**0.56**	**0.56**	0.69	0.58	0.94	0.65	1.20	0.68	1.70	0.83	2.06	0.95

Влияние частей PKVML на результаты экспериментов

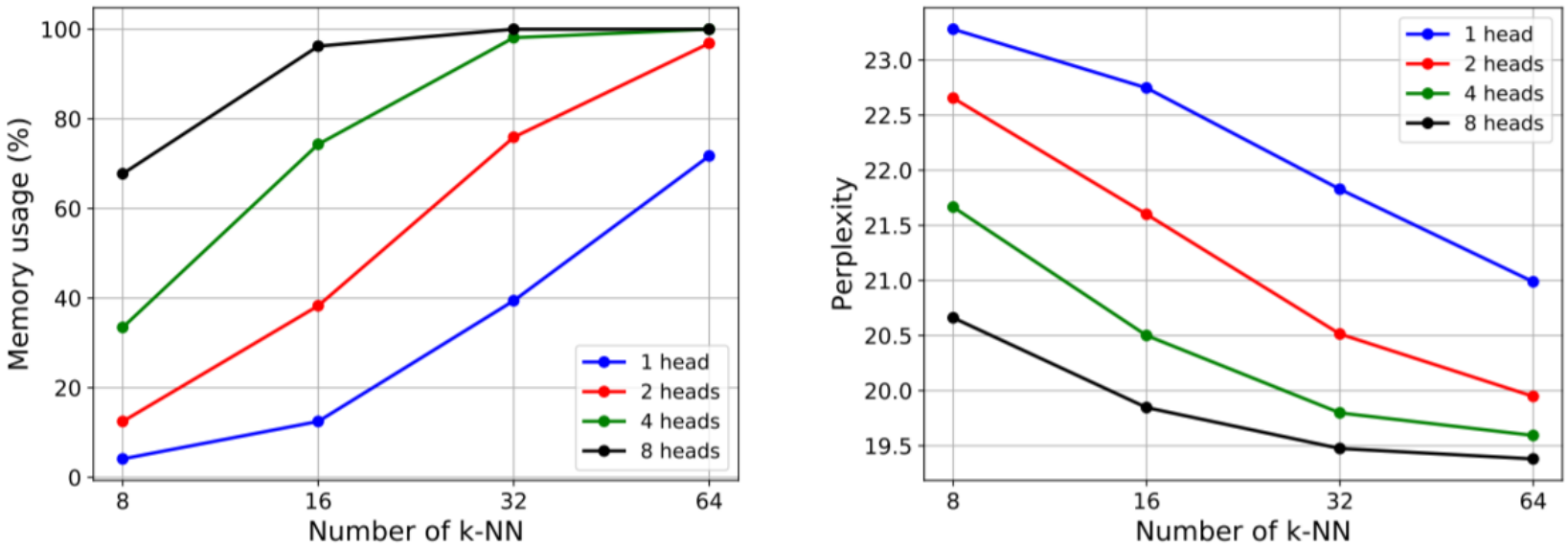


Figure 6: **Memory usage and perplexity** for different number of heads, and number of k-NN. Increasing the number of heads or k-NN increases both performance and the fraction of used memory slots.

Выводы

- Описан слой PKM
- Архитектура позволяет сократить inference время без потери точности
- Разреженность и разделение на subkeys позволило сделать основной шаг намного быстрее

References

Guillaume Lample et al. (2019)

Large Memory Layers with Product Keys

The End

Вопросы

- Что такое Top-k operator?
- Объясните что такое product key и что он ускоряет
- Какая complexity у этапа нахождения top-k в случае product keys?
- Что такое query network и какая у него обычно архитектура?