So when will spaCy support BERT?

Improving sparse transformer models for efficient self-attention

Giannis Daras

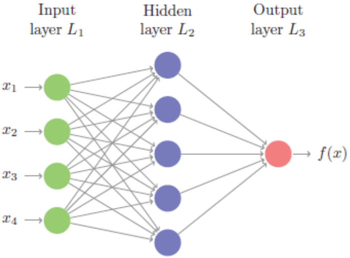
Explosion AI
Github: giannisdaras

National Technical University of Athens

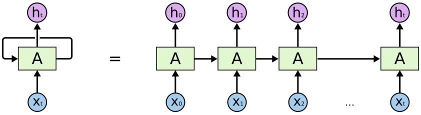
Outline

Neural network types

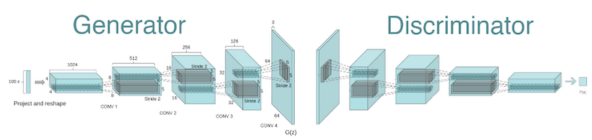
(a) FFN



(b) Reccurent neural network



(c) GAN



What are the problems with these approaches?

1. Feed forward neural nets are not taking advantage of the sequential nature of the data.

What are the problems with these approaches?

1. Feed forward neural nets are not taking advantage of the sequential nature of the data.
2. Recurrent neural networks:
   - Require multiple passes for one single sentence.

What are the problems with these approaches?

1. Feed forward neural nets are not taking advantage of the sequential nature of the data.
2. Recurrent neural networks:
    - Require multiple passes for one single sentence.
    - They are too slow.

What are the problems with these approaches?

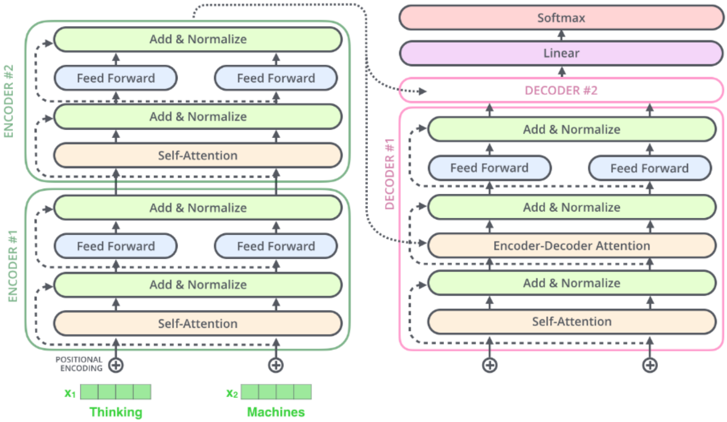
1. Feed forward neural nets are not taking advantage of the sequential nature of the data.
2. Recurrent neural networks:
   - Require multiple passes for one single sentence.
   - They are too slow.
   - Sometimes fail to understand long term dependencies.

What are the problems with these approaches?

1. Feed forward neural nets are not taking advantage of the sequential nature of the data.
2. Recurrent neural networks:
   - Require multiple passes for one single sentence.
   - They are too slow.
   - Sometimes fail to understand long term dependencies.
3. Some[GPAM+14] of them perform better in continuous problem (e.g. images).

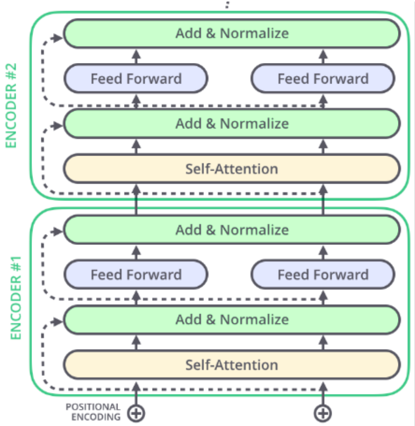
Transformer

Figure: Attention Is All You Need Transformer



Source [The Illustrated Transformer]

Transformer Encoder



Modified from: [The Illustrated Transformer], Model: [DCLT18]

Transformers advantages

Transformers:

Transformers advantages

Transformers:

- Achieve state-of-the-art in many tasks mainly due to the fact that they are not hypothesizing locality in data.
- Model long term dependencies.

Transformers advantages

Transformers:

- Achieve state-of-the-art in many tasks mainly due to the fact that they are not hypothesizing locality in data.
- Model long term dependencies.
- Do that in a single pass.

Transformers advantages

Transformers:

- Achieve state-of-the-art in many tasks mainly due to the fact that they are not hypothesizing locality in data.
- Model long term dependencies.
- Do that in a single pass.

Attention layer

This is achieved with the **attention layer**.

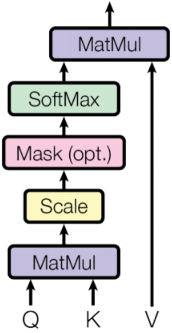
Attention operation in simple words

Every word "attends" to all the other words in the sentence in order to get a vector representation that is meaningful in the surrounding context.

What **lies** in word embeddings?

Scaled dot product attention

Figure: Scaled dot product attention



Source [VSP+17]

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V$$

Query matrix

Let's denote with:

1. d_k : the hidden size of the model (1024 in BERT large)
2. n_L: the number of tokens in the sentence

Query matrix

Query Q is a $n_L \times d_k$ matrix. Query matrix is the answer to the question: "Who is asking?".

Key matrix

Let's denote with:

1. d_k : the hidden size of the model (1024 in BERT large)
2. n_L: the number of tokens in the sentence

Key matrix

Key K is a $n_L \times d_k$ matrix. Key matrix is the answer to the question: "Who are we asking?".

Attention matrix

Let's denote with:

1. d_k : the hidden size of the model (1024 in BERT large)
2. n_L: the number of tokens in the sentence

Attention matrix

Attention matrix: $\mathrm{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right)$ is an $n_L \times n_L$ matrix. The cell $[i, j]$ of this matrix contains the answer to the question: "how important is token in the position j for token in the position i?".

Values matrix

Let's denote with:

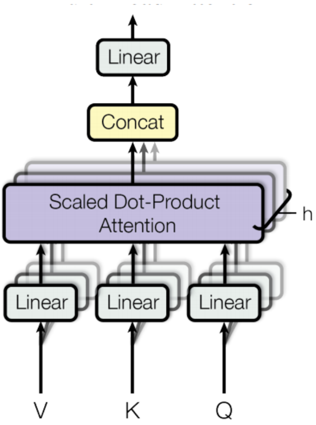
1. d_k : the hidden size of the model (1024 in BERT large)
2. n_L: the number of tokens in the sentence

Values matrix

V is an $n_L \times d_k$ matrix. We use this matrix to get model representations from softmax-ed product for the next layers in the stack.

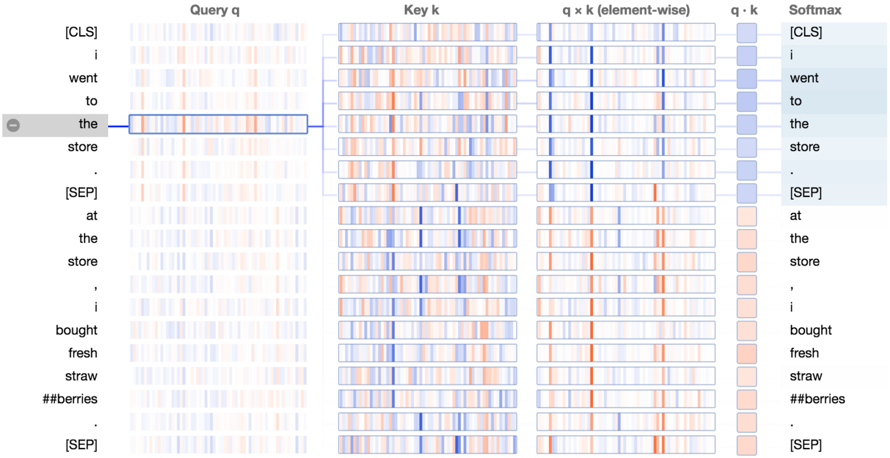
Multiheaded Attention

Figure: Multiheaded attention



Source [VSP+17]

Visualizing attention



Sources [Deconstructing Bert], [Vig19]

Attention in terms of complexity

Memory complexity of attention matrix computation?

Attention in terms of complexity

Memory complexity of attention matrix computation?

Attention matrix memory complexity

Attention matrix has $n_L \times n_L$ elements, where n_L denotes the number of elements in the sentence.

Attention in terms of complexity

Attention matrix memory complexity

Attention matrix has $n_L \times n_L$ elements, where n_L denotes the number of elements in the sentence.

Time complexity of attention matrix computation?

Attention in terms of complexity

Attention matrix memory complexity

Attention matrix has $n_L \times n_L$ elements, where n_L denotes the number of elements in the sentence.

Time complexity of attention matrix computation?

Attention matrix computation complexity

If we compute products in parallel, we need $O(n_L^2)$ operations to calculate the attention matrix.

Can we do better?

Open AI proposed attention in steps.

Can we do better?

Attention in steps

Instead of having each token attend to all the others, we can break attention to discrete steps and use closure.

Can we do better?

Attention in steps

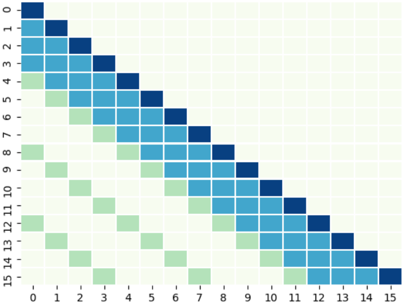
Instead of having each token attend to all the others, we can break attention to discrete steps and use closure.

Closure

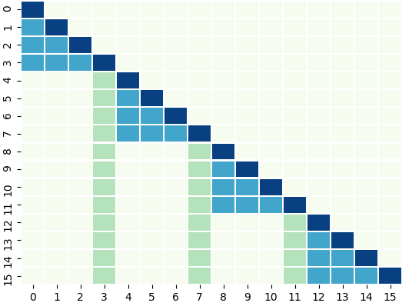
If token a attends to token b and at a previous attention step, token b attended to token c, then a has attended to token c (through b).

Generative Sparse Transformers Proposed Patterns

Strided Pattern

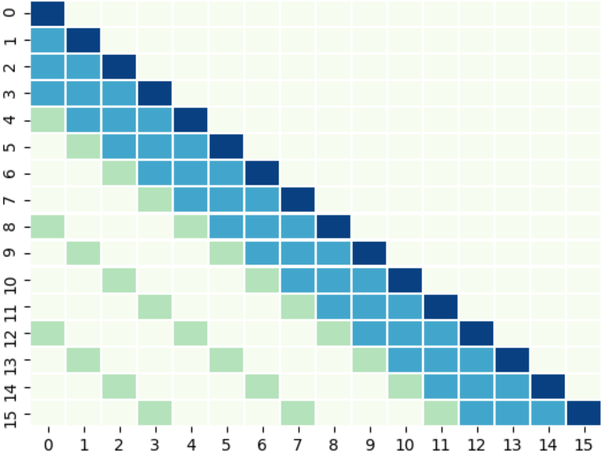


Fixed Pattern

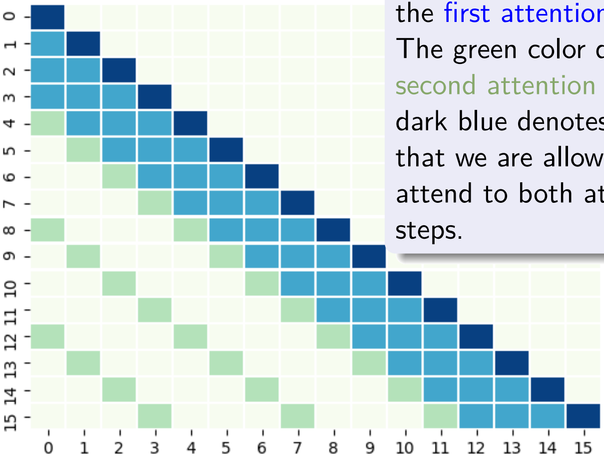


Source [CGRS19]

Strided pattern - Explained



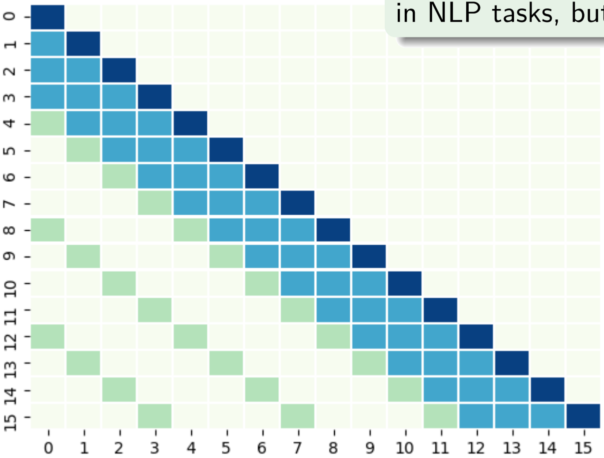
Strided pattern - Explained



Colors

The light blue color denotes the first attention step. The green color denotes the second attention step. The dark blue denotes the cells that we are allowed to attend to both attention steps.

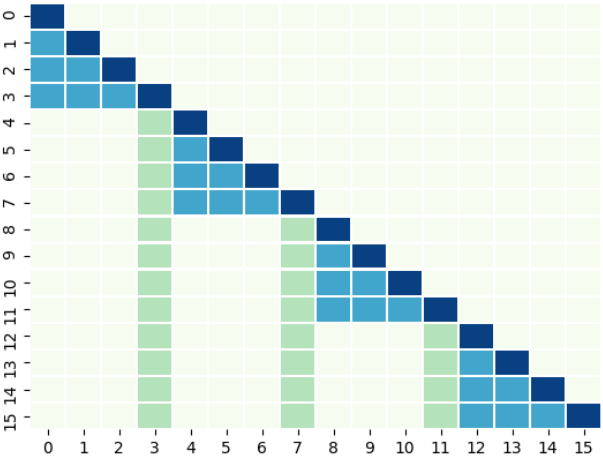
Strided pattern - Explained



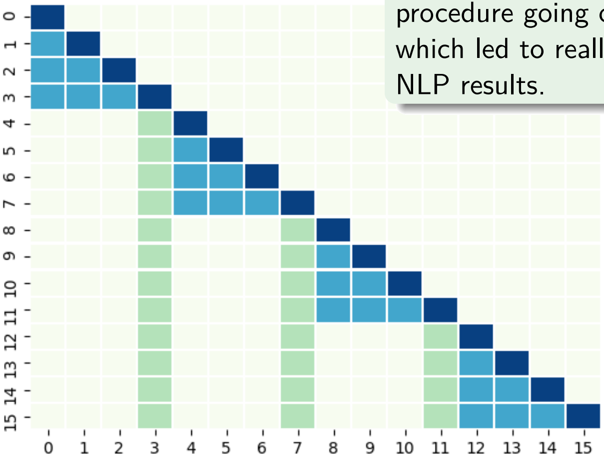
Performance

This pattern didn't go well in NLP tasks, but why?

Fixed pattern - Explained



Fixed pattern - Explained



Performance

There is a summarization procedure going on here, which led to really good NLP results.

Information flow graphs

OpenAI proposed two patterns, **but**:

- Didn't tell us how to choose the proposed patterns.

Information flow graphs

OpenAI proposed two patterns, **but**:

- Didn't tell us how to choose the proposed patterns.
  - Do we know if a pattern is going to be better than another, without running the experiments?

Information flow graphs

OpenAI proposed two patterns, **but**:

- Didn't tell us how to choose the proposed patterns.
  - Do we know if a pattern is going to be better than another, without running the experiments?
  - What is the objective we are trying to get to when we design patters for sparse transformers?

Information flow graphs

OpenAI proposed two patterns, **but**:

- Didn't tell us how to choose the proposed patterns.
  - Do we know if a pattern is going to be better than another, without running the experiments?
  - What is the objective we are trying to get to when we design patters for sparse transformers?
- Proposed patterns are **not** bidirectional.

Information flow graphs

OpenAI proposed two patterns, **but**:

- Didn't tell us how to choose the proposed patterns.
  - Do we know if a pattern is going to be better than another, without running the experiments?
  - What is the objective we are trying to get to when we design patters for sparse transformers?
- Proposed patterns are **not** bidirectional.

Information Flow Graphs

To solve those problems, we associate with the attention patterns graphs, we call Information Flow Graphs.

Information Flow Graphs

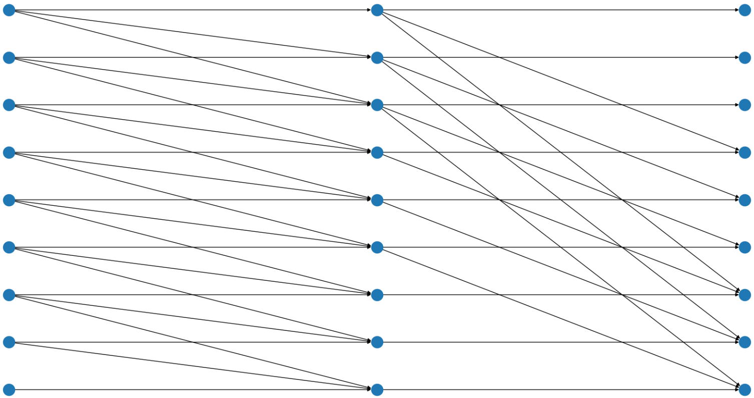
Information Flow Graphs are graphs that show how the information "flows" in the sparse attention factorization.

Information Flow Graphs

Let's explain it with a visual. How does the information flow graph of Open AI proposed strided pattern look like?

Information Flow Graphs

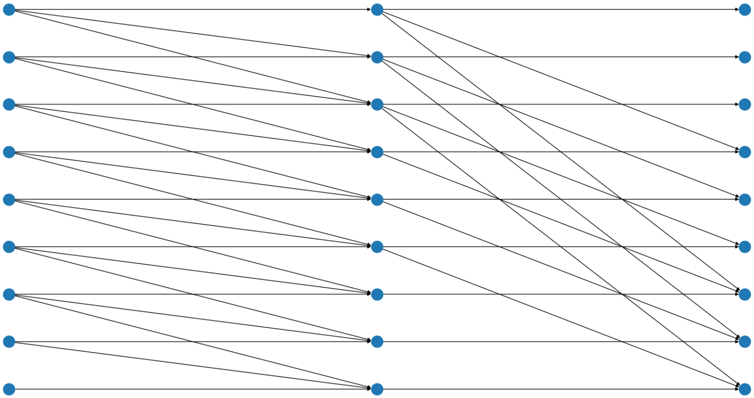
Information Flow Graph of OpenAI strided pattern



Information Flow Graphs

Losing information

Can all tokens at step 2 attend to all tokens of step 0?



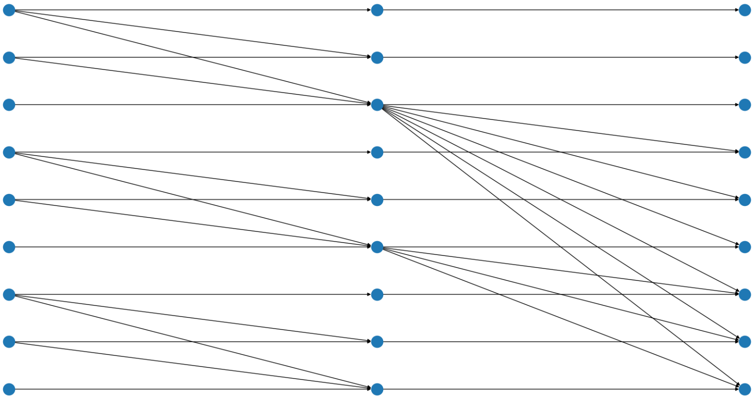
Information Flow Graphs

Losing information

We lose information with the fixed pattern as well.

Information Flow Graphs

Information Flow Graph of Open AI fixed pattern



Comments:

Source of information loss

We are losing information because the patterns are not bi-directional.

Comments:

Source of information loss

We are losing information because the patterns are not bi-directional.

Bi-directional patterns

We propose bidirectional patterns ☺.

Comments:

Source of information loss

We are losing information because the patterns are not bi-directional.

Bi-directional patterns

We propose bidirectional patterns ☺.

Constraints

We should consider constraints on the design of IFGs.

Constraints in designing full information attention patterns

Edge constraints

Edge constraints:

Edge constraints:

- Dense attention is 1 step and uses $O(n^2)$ edges.

Constraints in designing full information attention patterns

Edge constraints

Edge constraints:

- Dense attention is 1 step and uses $O(n^2)$ edges.
- Open AI proposed patterns with 2 steps and in total $O(n\sqrt{n})$ edges.

Edge constraints:

- Dense attention is 1 step and uses $O(n^2)$ edges.
- Open AI proposed patterns with 2 steps and in total $O(n\sqrt{n})$ edges.
- We prove that with a fixed in-degree we cannot design 2-steps patterns that have full information and use less than $O(n\sqrt{n})$ edges.

Constraints in designing full information attention patterns

Edge constraints

Edge constraints:

- Dense attention is 1 step and uses $O(n^2)$ edges.
- Open AI proposed patterns with 2 steps and in total $O(n\sqrt{n})$ edges.
- We prove that with a fixed in-degree we cannot design 2-steps patterns that have full information and use less than $O(n\sqrt{n})$ edges.
- We propose bi-directional patterns with fixed in degree that use $O(n\sqrt{n})$ edges.

Constraints in designing full information attention patterns

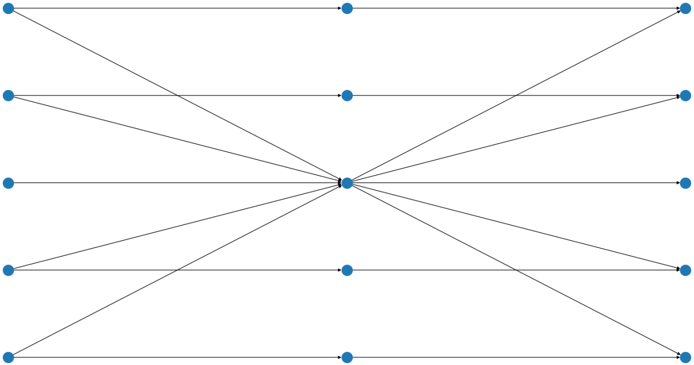
Flow constraints

Why do we need fixed in degree? Answer: This implies flow constraints that we are setting under the hood.

Constraints in designing full information attention patterns

Flow constraints

This clearly doesn't feel right, but it has full information:



Constraints in designing full information attention patterns

Flow constraints

Constant in degree

A fix to flow problem is to use constant in degree (as low as possible).

Constraints in designing full information attention patterns

Flow constraints

More generic approach

Allow each token to output flow max 1, and calculate the total output flow.

Full information patterns

Fixed patterns

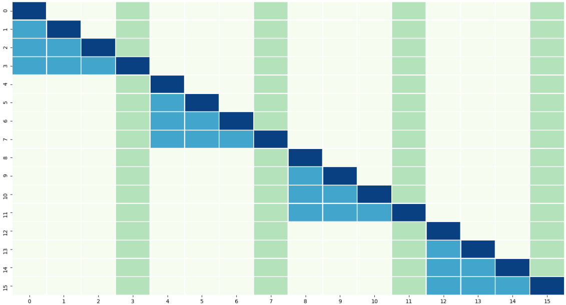
Fixed patterns

We consider natural extensions of the fixed patterns.

Fixed patterns

We consider natural extensions of the fixed patterns.

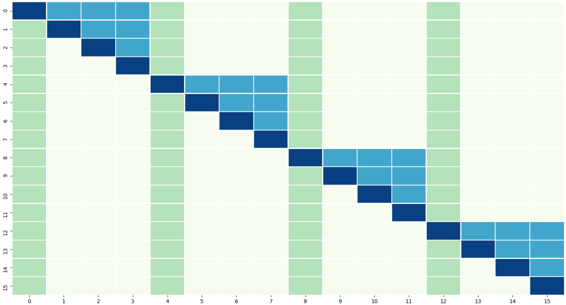
Figure: LTR Full Information Sparse Attention Pattern



Fixed patterns

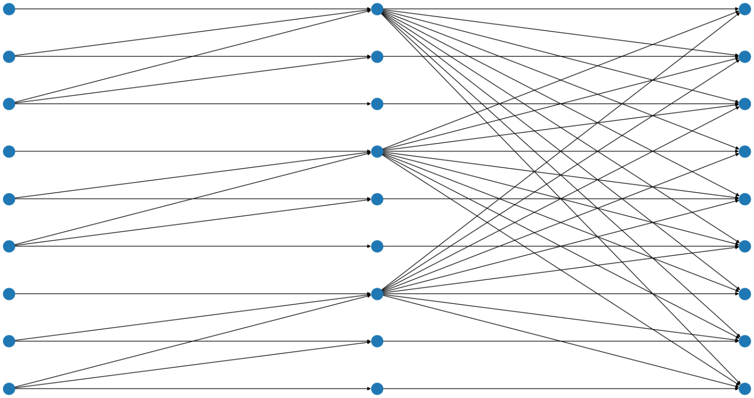
We consider natural extensions of the fixed patterns.

Figure: RTL Full Information Sparse Attention Pattern



Information Flow Graph

RTL Information Flow Graph



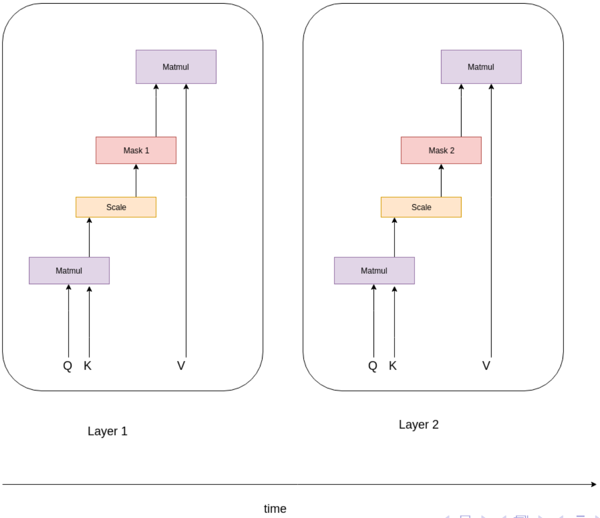
Attention modes

One thing we didn't answer is how we actually do attention in steps. We will explore three different ways.

Attention modes

Attention with sequential interleave

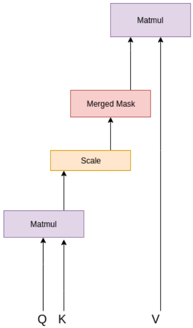
We use the same dense attention code, but the i uses the $i \mod p$ pattern.

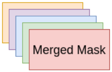


Attention modes

Attention with merged head

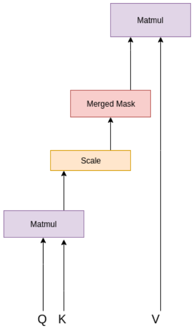
All heads attend to the positions that the masks combined will allow.

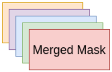




Attention with merged head

All heads attend to the positions that the masks combined will allow.

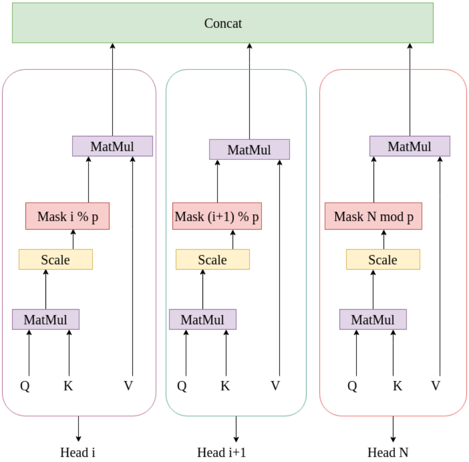




Why this makes sense?

This only makes sense if we are using multiple stacked layers. If not, the closure property we used to design patterns doesn't hold.

Figure: Separate heads separate patterns



Attention modes

Which mode to use?

In practice, we found that different heads is best because it allows exploration of different patterns at the same time, but this is still something to be validated.

Lessons to take home

- Transformer models are too big for industrial NLP frameworks, such as spaCy.

Lessons to take home

- Sparse Transformers as spaCy models have small running overhead compared to CNNs.

Lessons to take home

- The pre-training support of spaCy could possibly become more powerful.

Credits

People involved in this research:

- Giannis Daras
- Alex Dimakis
- George Paraskevopoulos
- Alex Potamianos

References I

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever, *Generating long sequences with sparse transformers*, arXiv preprint arXiv:1904.10509 (2019).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, arXiv preprint arXiv:1810.04805 (2018).

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, *Generative adversarial nets*, Advances in neural information processing systems, 2014, pp. 2672–2680.

Jesse Vig, *A multiscale visualization of attention in the transformer model*, arXiv preprint arXiv:1906.05714 (2019).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, *Attention is all you need*, Advances in neural information processing systems, 2017, pp. 5998–6008.