This is AI4001

GCR : t37g47w

References

https://web.stanford.edu/class/cs224n/slides/cs224n-2022-lecture15-guu.pdf

https://web.stanford.edu/class/cs224n/slides/cs224n-2022-lecture16-CNN-TreeRNN.pdf

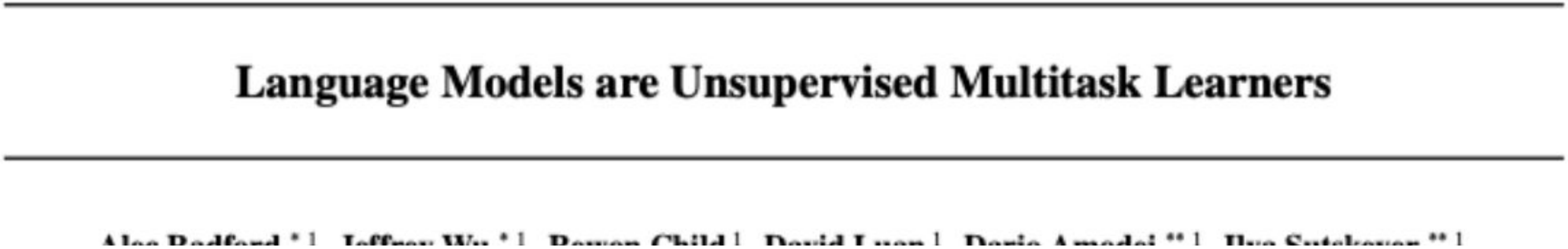
https://blog.research.google/2022/01/lamda-towards-safe-grounded-and-high.html

Emergent abilities of large language models: GPT-2 (2019)

Let's revisit the Generative Pretrained Transformer (GPT) models from OpenAI as an example:

GPT-2 (1.5B parameters; Radford et al., 2019)

- Same architecture as GPT, just bigger (117M -> 1.5B)
- But trained on **much more data**: 4GB -> 40GB of internet text data (WebText)
    - Scrape links posted on Reddit w/ at least 3 upvotes (rough proxy of human quality)



Language Models are Unsupervised Multitask Learners

Alec Radford * 1 Jeffrey Wu * 1 Rewon Child 1 David Luan 1 Dario Amodei ** 1 Ilya Sutskever ** 1

Emergent zero-shot learning

One key emergent ability in GPT-2 is **zero-shot learning**: the ability to do many tasks with **no examples,** and **no gradient updates,** by simply:

- Specifying the right sequence prediction problem (e.g. question answering):

  Passage: Tom Brady... Q: Where was Tom Brady born? A: ...

- Comparing probabilities of sequences (e.g. Winograd Schema Challenge [Levesque, 2011]):

  The cat couldn’t fit into the hat because it was too big.
  Does it = the cat or the hat?

  ≡ Is P(...because **the cat** was too big) >=
  P(...because **the hat** was too big)?

[Radford et al., 2019]

Emergent abilities of large language models: GPT-3 (2020)

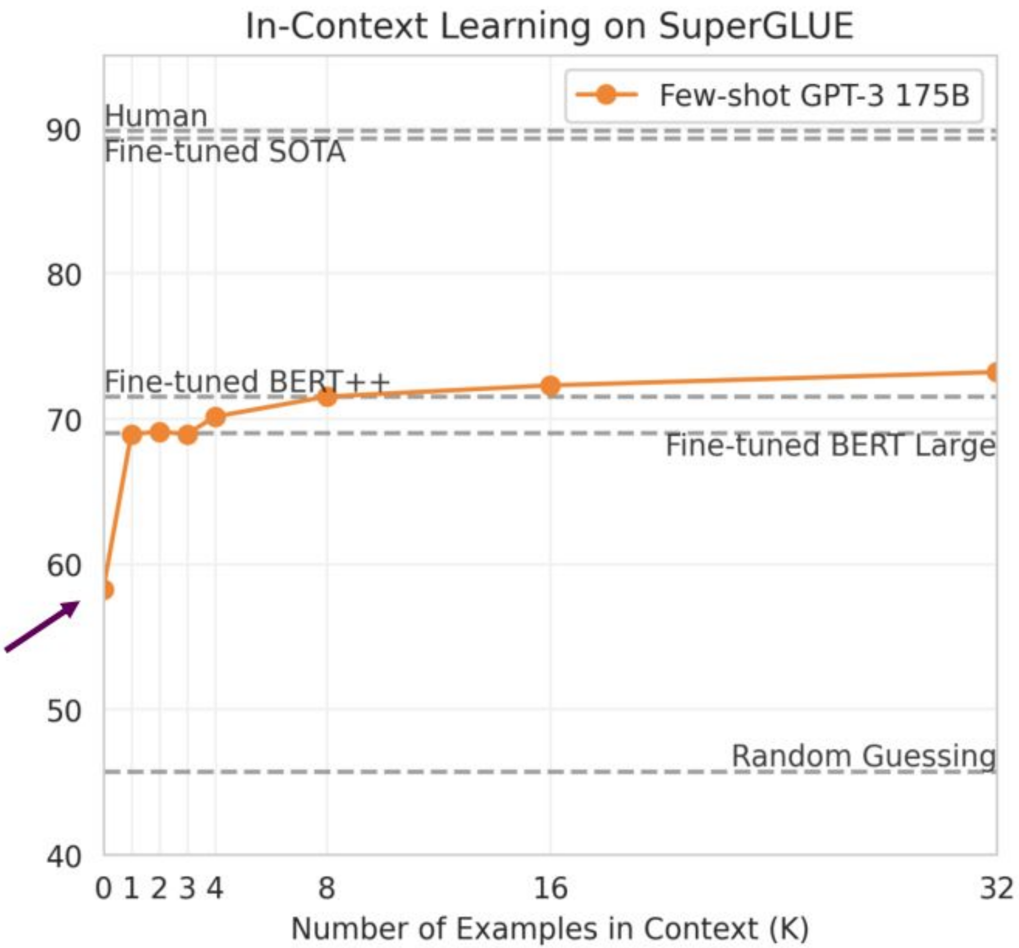
GPT-3 (175B parameters; Brown et al., 2020)

- Another increase in size (1.5B -> **175B**)
- and data (40GB -> **over 600GB**)

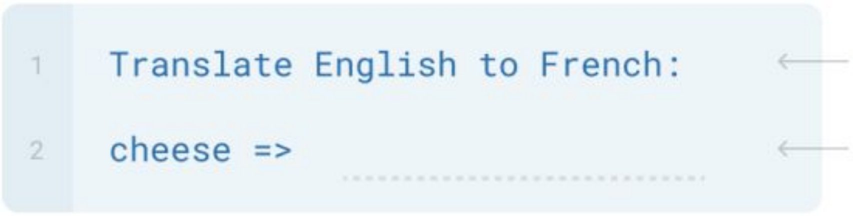
Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*

Emergent few-shot learning



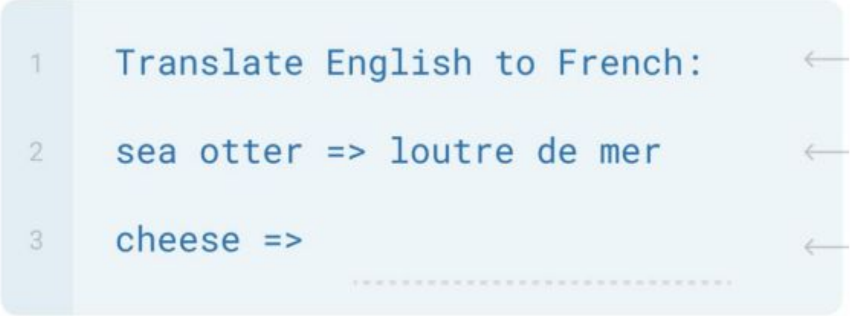
Zero-shot



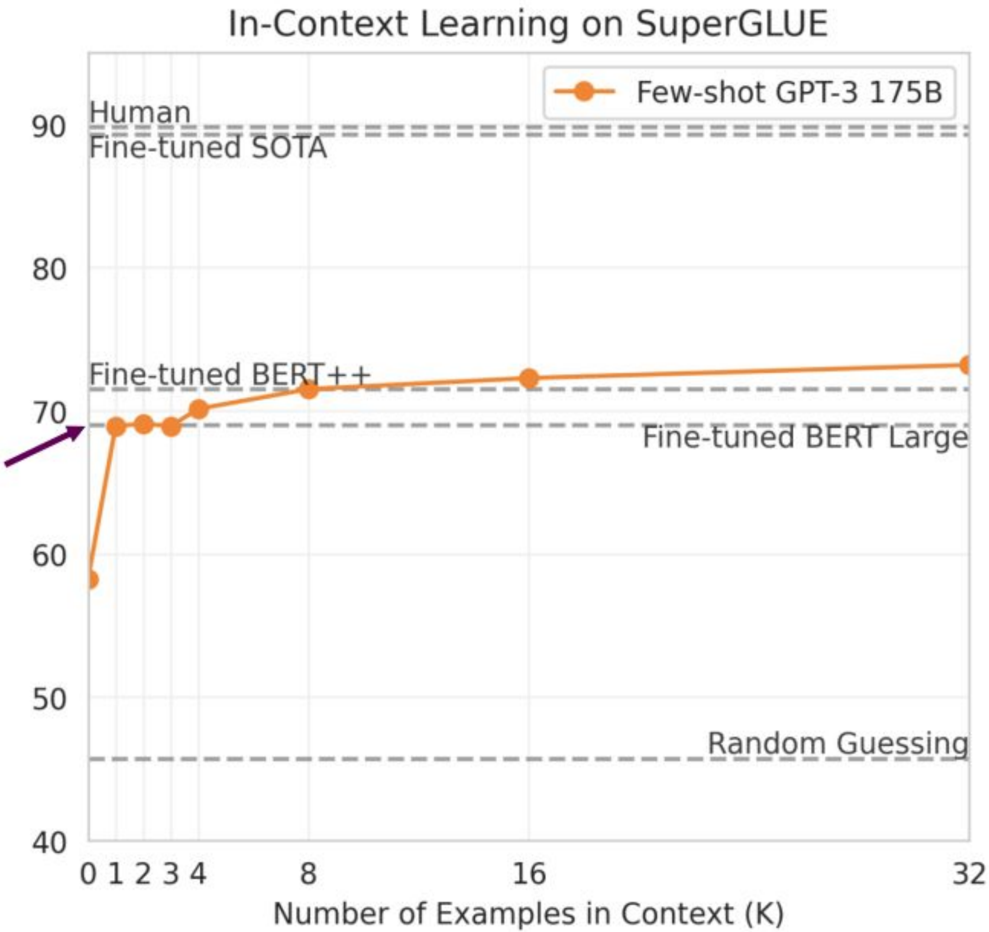
```
Translate English to French:
cheese =>
```

Emergent few-shot learning

One-shot

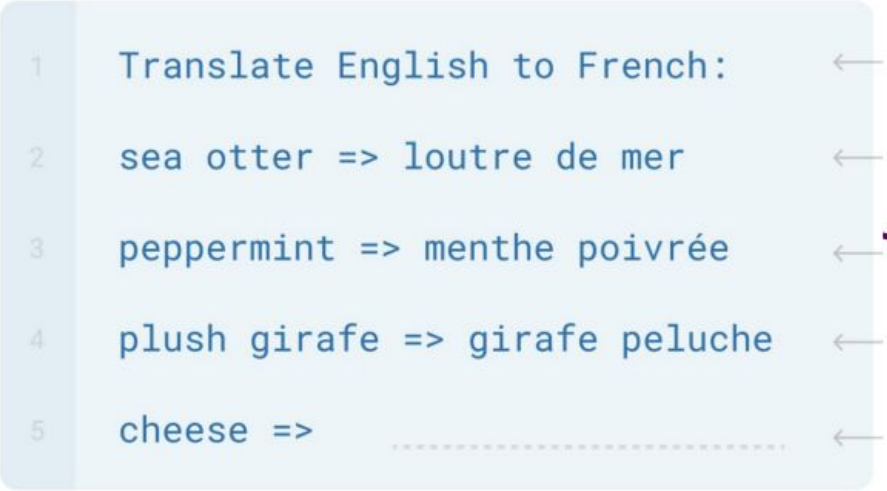


```
Translate English to French:
sea otter => loutre de mer
cheese =>
```



Emergent few-shot learning

Few-shot



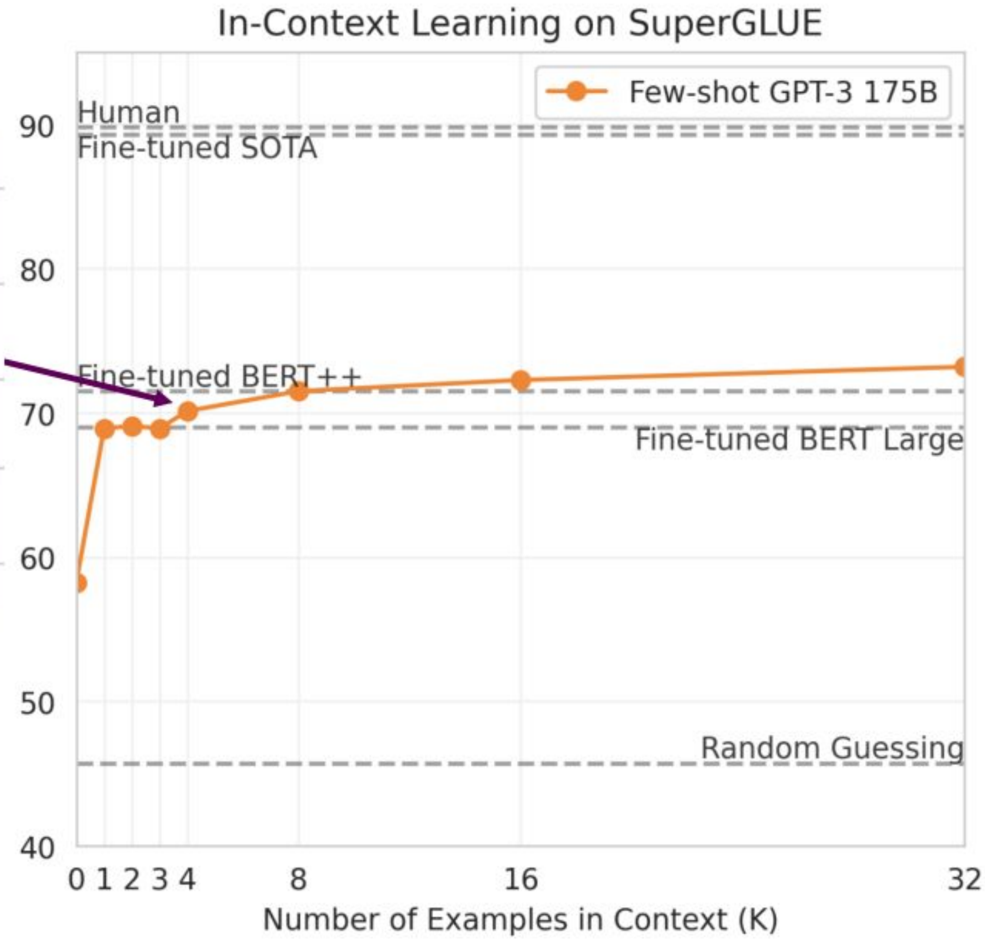
```
Translate English to French:

sea otter => loutre de mer

peppermint => menthe poivrée

plush girafe => girafe peluche

cheese =>
```



Some example tasks that AI cannot solve today

- Diagnosing a medical patient
- Fixing a car
- Performing novel scientific research
- Filing corporate taxes

"Intelligence" is required, but **domain knowledge** is just as important.

The part of the intestine most commonly affected by Crohn's disease is ______

GPT-2: the rectum
Correct answer: the ileum

What is missing from Transformers right now?

- **We can automatically acquire knowledge from the web, but...**
- ... a lot of it is noisy or incorrect: misinformation, rumors, opinions.
- ... we cannot trace the model's knowledge back to an attributable source.

- **We can edit individual facts inside a Transformer's memory, but...**
- ... it doesn't work reliably yet.
- ... current approaches break down after multiple edits.

- **We can store knowledge inside feedforward layers, but...**
- ... current memory capacity is too small, and scaling up is expensive!

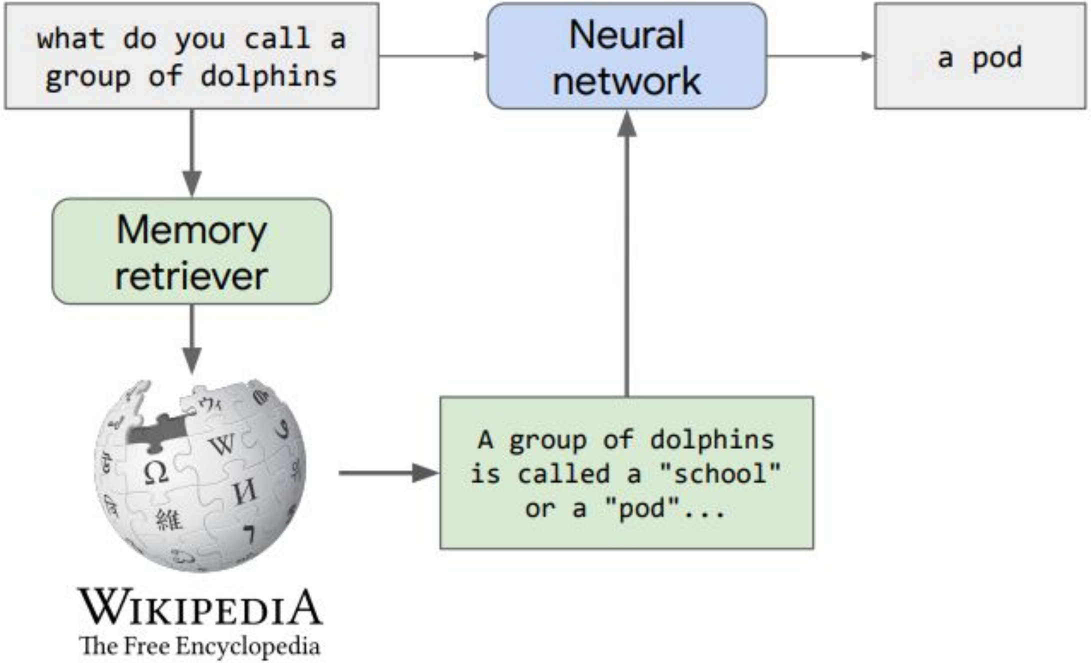
Wish list

- **Fast and modular knowledge editing**
  - Robustly update the model N times without breaking its behavior on other tasks.
- **Attribution and interpretability**
  - Trace a model's knowledge back to a particular document / training example.
- **Efficient scaling**
  - Increase the model's memory size by 10x without paying 10x more compute.

Example: use GPT-3 to do question answering over your company / school wiki.

- Original GPT-3 training run cost >$12M.
- We can't afford this for every company / school.
- Company / school info is always changing (e.g. COVID requirements).

What is a memory-augmented model?



what do you call a group of dolphins
Neural network
a pod
Memory retriever
WIKIPEDIA
The Free Encyclopedia
A group of dolphins is called a "school" or a "pod"...

A memory could be:

- *Document on the web*
- *Record in a database*
- *Training example*
- *Entity embedding*
- ...

Potentially meets our wish list:

- *Easily edit knowledge*
- *Attribution*
- *Efficient scaling*

LaMDA 2

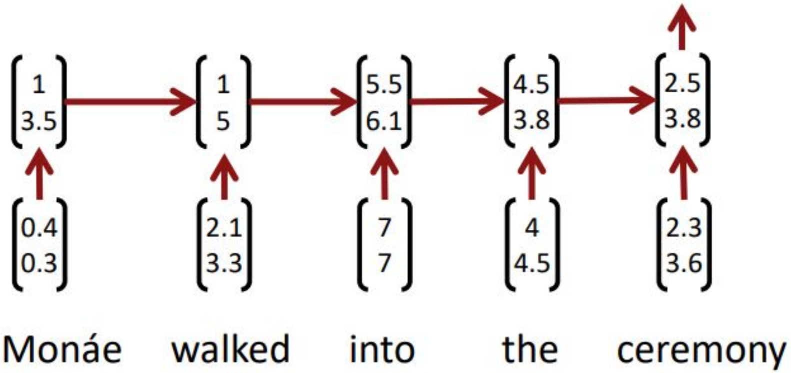
What are some applications?

- **Open-domain dialog / question answering**
  - Retrieve documents on the web.
- **Code generation**
  - Retrieve code snippets from Stack Overflow.
- **Image generation**
  - Retrieve reference pictures of people, places, etc.
- **Fact checking**
  - Retrieve documents that support or refute a claim.

ConvNets

1. From RNNs to Convolutional Neural Nets

- Recurrent neural nets cannot capture phrases without prefix context
- Often capture too much of last words in final vector



- E.g., softmax for word prediction is usually calculated based on the last step

From RNNs to Convolutional Neural Nets

- Main Convolutional Neural Net (CNN/ConvNet) idea:
  - What if we compute vectors for every possible word subsequence of a certain length?

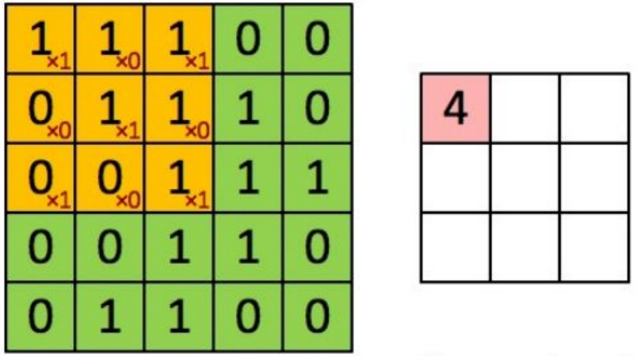
- Example: "tentative deal reached to keep government open" computes vectors for:
  - tentative deal reached, deal reached to, reached to keep, to keep government, keep government open

- Regardless of whether phrase is grammatical
  - Not very linguistically or cognitively plausible

- Then group them afterwards (more soon)

What is a convolution anyway?

- 1d discrete convolution generally: $(f * g)[n] = \sum_{m=-M}^{M} f[n-m]g[m].$
- Convolution is classically used to extract features from images
  - Models position-invariant identification
  - Go to cs231n!
- 2d example →
- Yellow color and red numbers show filter (=kernel) weights
- Green shows input
- Pink shows output



$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$	0	0
$0_{\times 0}$	$1_{\times 1}$	$1_{\times 0}$	1	0
$0_{\times 1}$	$0_{\times 0}$	$1_{\times 1}$	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved Feature

From Stanford UFLDL wiki

A 1D convolution for text

tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3

Apply a **filter** (or **kernel**) of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

t,d,r	−1.0
d,r,t	−0.5
r,t,k	−3.6
t,k,g	−0.2
k,g,o	0.3

1D convolution for text with padding

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Ø,t,d	−0.6
t,d,r	−1.0
d,r,t	−0.5
r,t,k	−3.6
t,k,g	−0.2
k,g,o	0.3
g,o,Ø	−0.5

Apply a **filter** (or **kernel**) of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

3 channel 1D convolution with padding = 1 and 3 filters

Ø	**0.0**	**0.0**	**0.0**	**0.0**
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	**0.0**	**0.0**	**0.0**	**0.0**

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1

Apply 3 **filters** of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

Could also use (zero) padding = 2

Also called “wide convolution”

conv1d, padded with max pooling over time

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1

max p	0.3	1.6	1.4

Apply 3 **filters** of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

Other (maybe less useful) notions: stride = 2

∅	**0.0**	**0.0**	**0.0**	**0.0**
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
∅	**0.0**	**0.0**	**0.0**	**0.0**

∅,t,d	−0.6	0.2	1.4
d,r,t	−0.5	−0.1	0.8
t,k,g	−0.2	0.1	1.2
g,o,∅	−0.5	−0.9	0.1

Apply 3 **filters** of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

1	−1	2	−1
1	0	−1	3
0	2	2	1

Local max pool, stride = 2

Ø	0.0	0.0	0.0	0.0
tentative	0.2	0.1	−0.3	0.4
deal	0.5	0.2	−0.3	−0.1
reached	−0.1	−0.3	−0.2	0.4
to	0.3	−0.3	0.1	0.1
keep	0.2	−0.3	0.4	0.2
government	0.1	0.2	−0.1	−0.1
open	−0.4	−0.4	0.2	0.3
Ø	0.0	0.0	0.0	0.0

Apply 3 **filters** of size 3

3	1	2	−3
−1	2	1	−3
1	1	−1	1

1	0	0	1
1	0	−1	−1
0	1	0	1

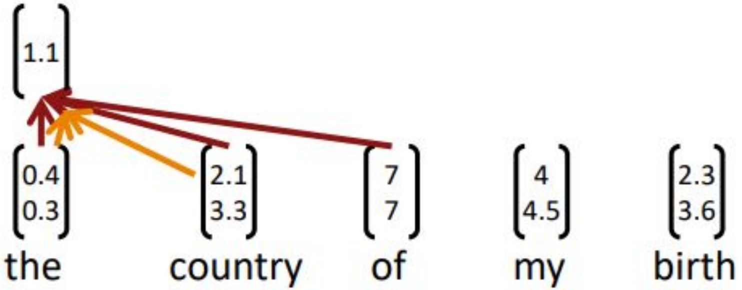
1	−1	2	−1
1	0	−1	3
0	2	2	1

Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1
Ø	−Inf	−Inf	−Inf

Ø,t,d,r	−0.6	1.6	1.4
d,r,t,k	−0.5	0.3	0.8
t,k,g,o	0.3	0.6	1.2
g,o,Ø,Ø	−0.5	−0.9	0.1

Single Layer CNN for Sentence Classification

- A simple use of one convolutional layer and **pooling**
- Word vectors: $\mathbf{x}_i \in \mathbb{R}^k$
- Sentence: $\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus x_2 \oplus \cdots \oplus \mathbf{x}_n$ (vectors concatenated)
- Concatenation of words in range: $\mathbf{x}_{i:i+j}$ (symmetric more common)
- Convolutional filter: $\mathbf{w} \in \mathbb{R}^{hk}$ (over window of *h* words)
- Note, filter is a vector
- Filter could be of size 2, 3, or 4 words:

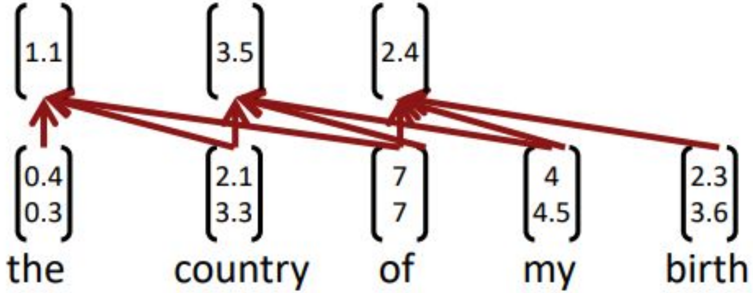


Single layer CNN

- Filter **w** is applied to all possible windows (concatenated vectors)
- To compute feature (one *channel*) for CNN layer:

$$c_i = f(\mathbf{w}^T \mathbf{x}_{i:i+h-1} + b)$$

- Sentence: $\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \ldots \oplus \mathbf{x}_n$
- All possible windows of length *h*: $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}\}$
- Result is a feature map: $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$



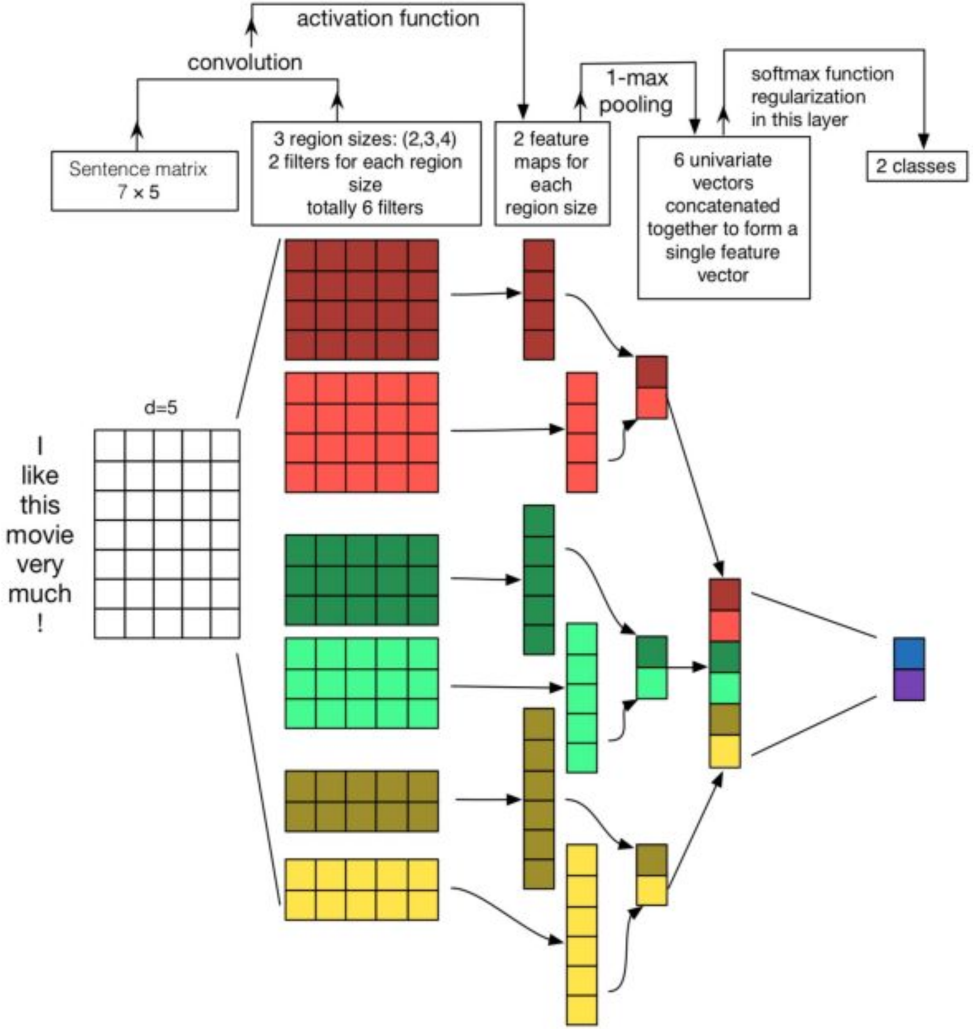
Kim (2014)

From:

Zhang and Wallace (2015) A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification

https://arxiv.org/pdf/1510.03820.pdf

(follow on paper, not famous, but a nice picture)



activation function
convolution
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
1-max pooling
6 univariate vectors concatenated together to form a single feature vector
softmax function regularization in this layer
2 classes
d=5
I
like
this
movie
very
much
!

3. Model comparison: Our growing toolkit

- **Bag of Vectors**: Surprisingly good baseline for simple classification problems.
  - Especially if followed by a few ReLU layers! (See paper: Deep Averaging Networks)
- **Window Model**: Good for single word classification for problems that do not need wide context. E.g., POS, NER
- **CNNs:** good for classification, need zero padding for shorter phrases, somewhat implausible/hard to interpret, **easy to parallelize on GPUs.** Efficient and versatile
- **Recurrent Neural Networks**: Cognitively plausible (reading from left to right), not best for classification (if just use last state), much slower than CNNs, good for sequence tagging and classification, good for language models, can be amazing with attention
- **Transformers:** Great for language models, great for sentence calculations. In general, still the best thing since sliced bread.
  - But, FWIW, recent Vision Transformer work argues that CNNs and transformers have complementary advantages, and you can usefully use both

Thank You

Picture Time