Going deeper with text

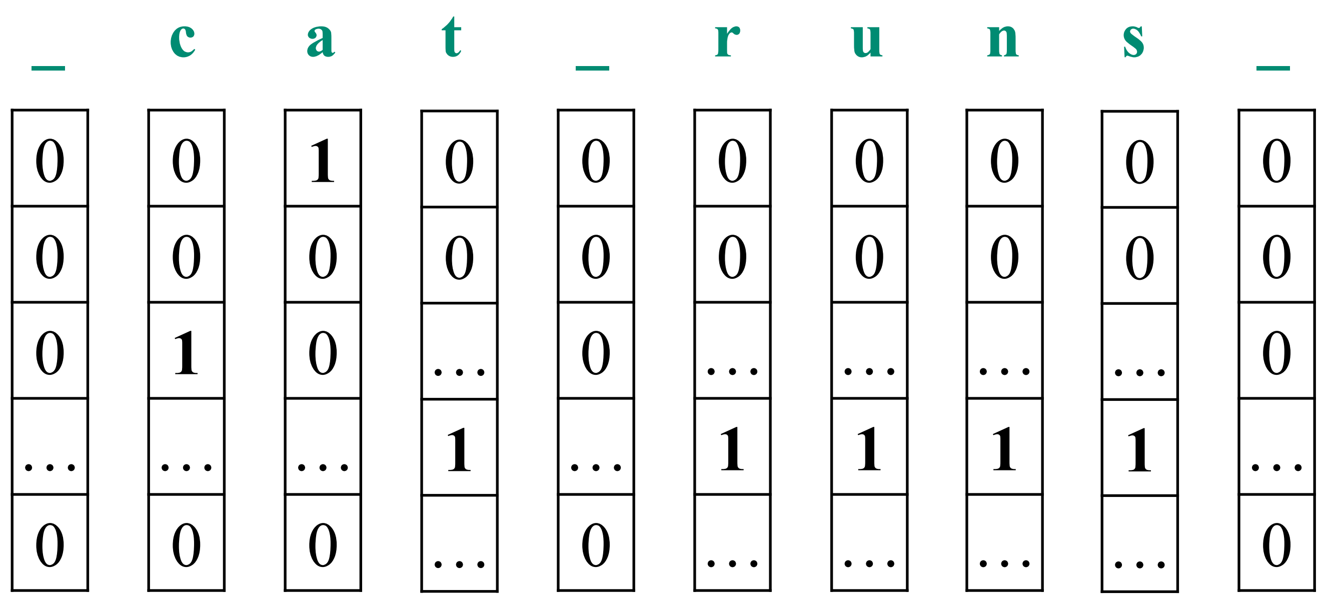
What is text?

You can think of text as a sequence of

- **Characters**
- Words
- Phrases and named entities
- Sentences
- Paragraphs
- …

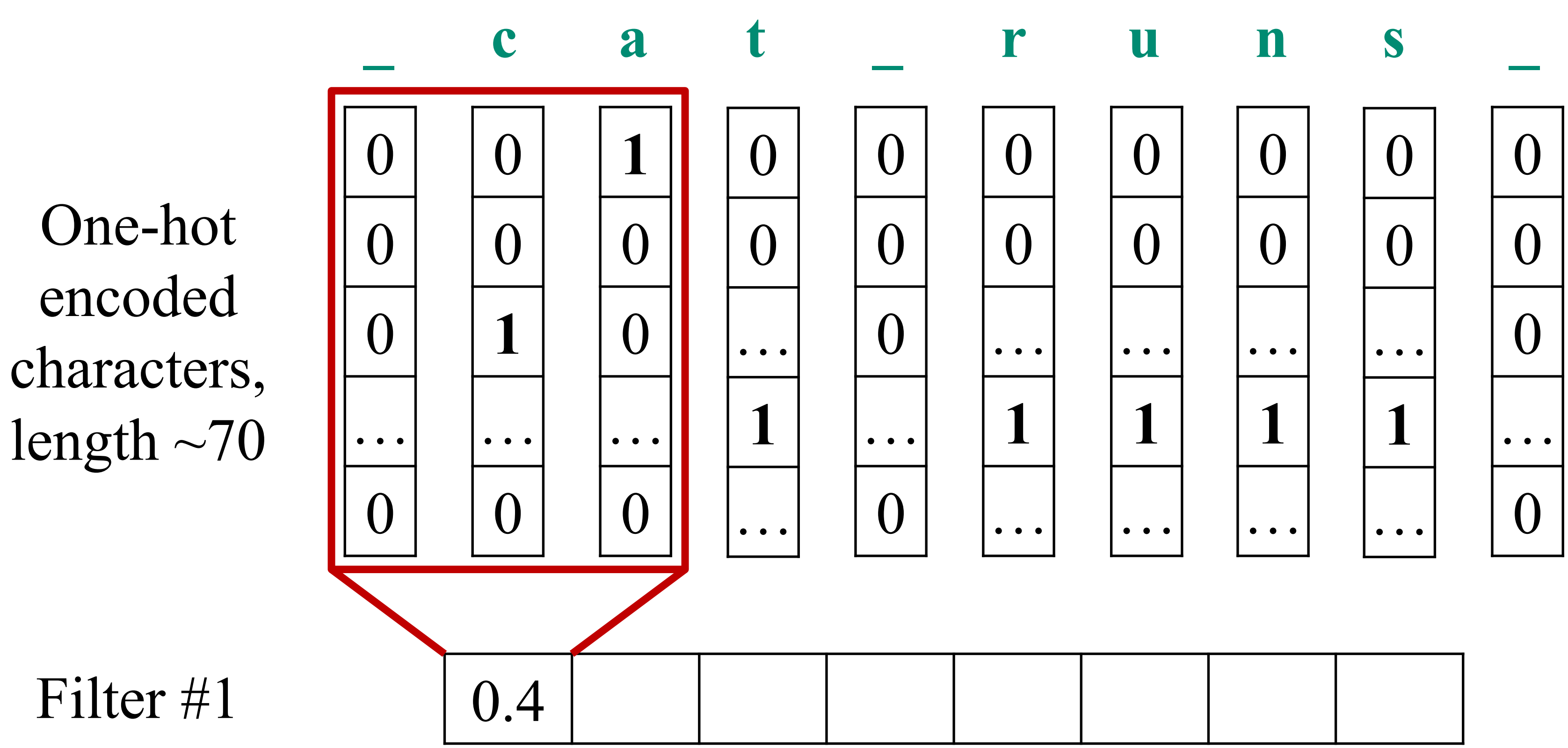
Text as a sequence of characters

One-hot encoded characters, length ~70

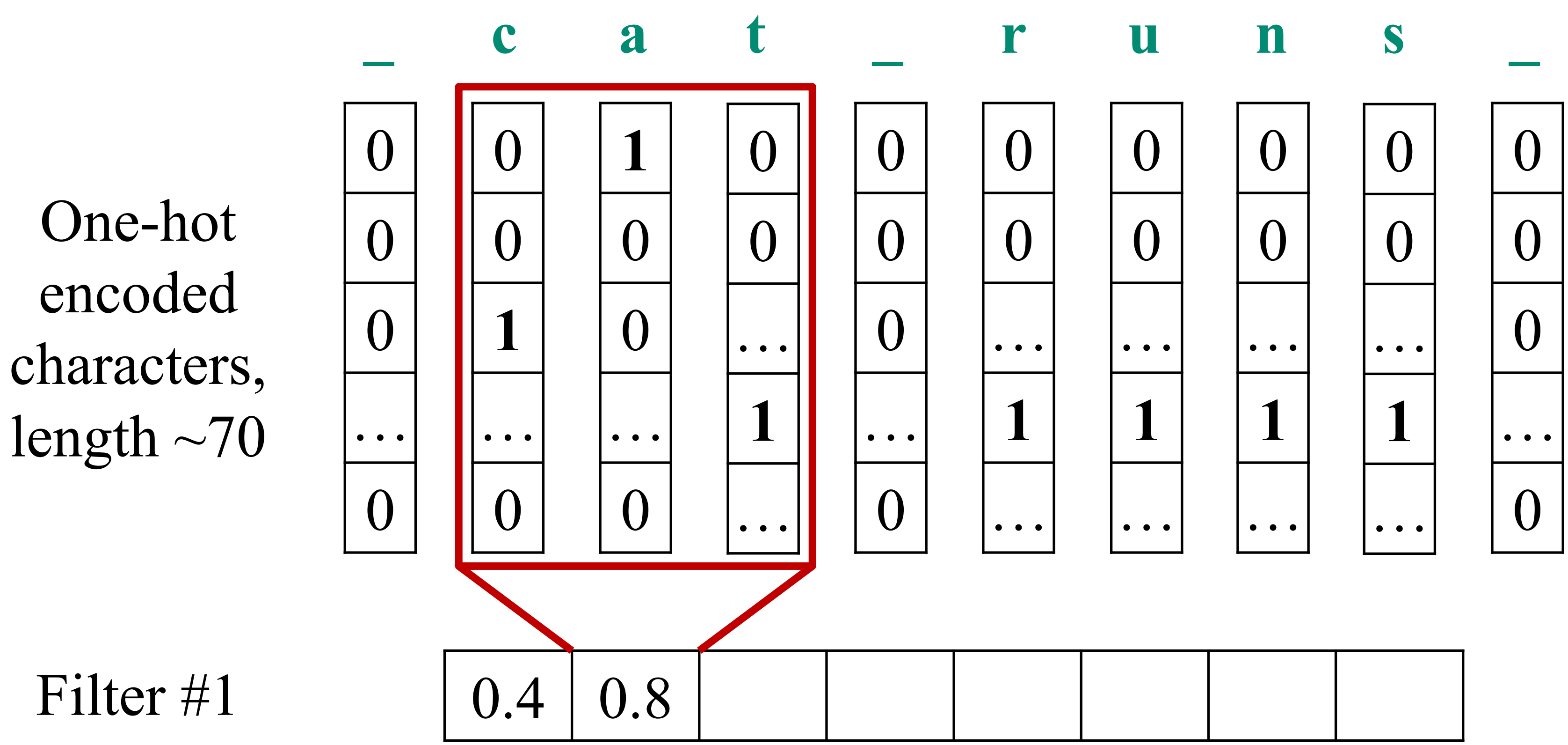


Let’s start with character *n*-grams!

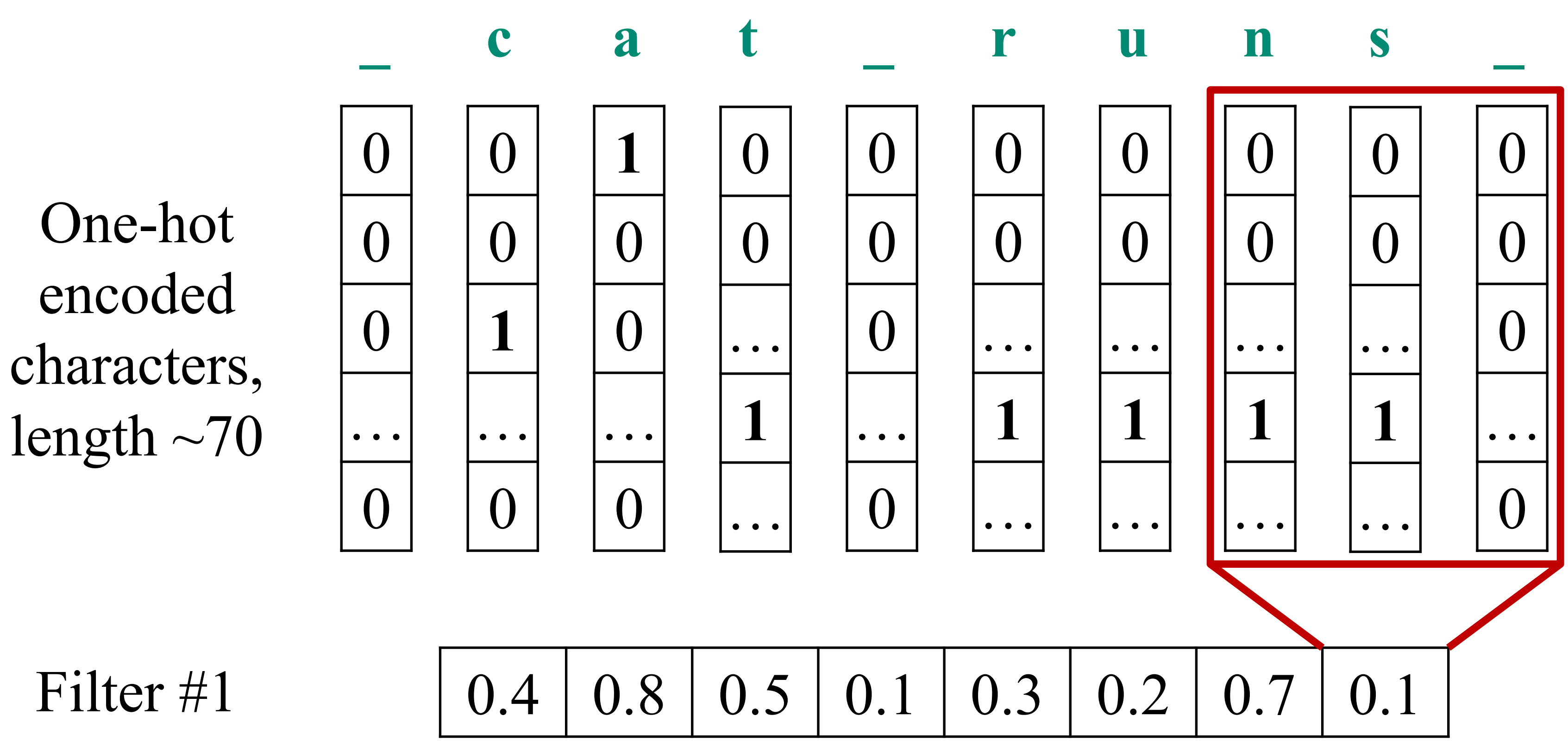
1D convolutions on characters



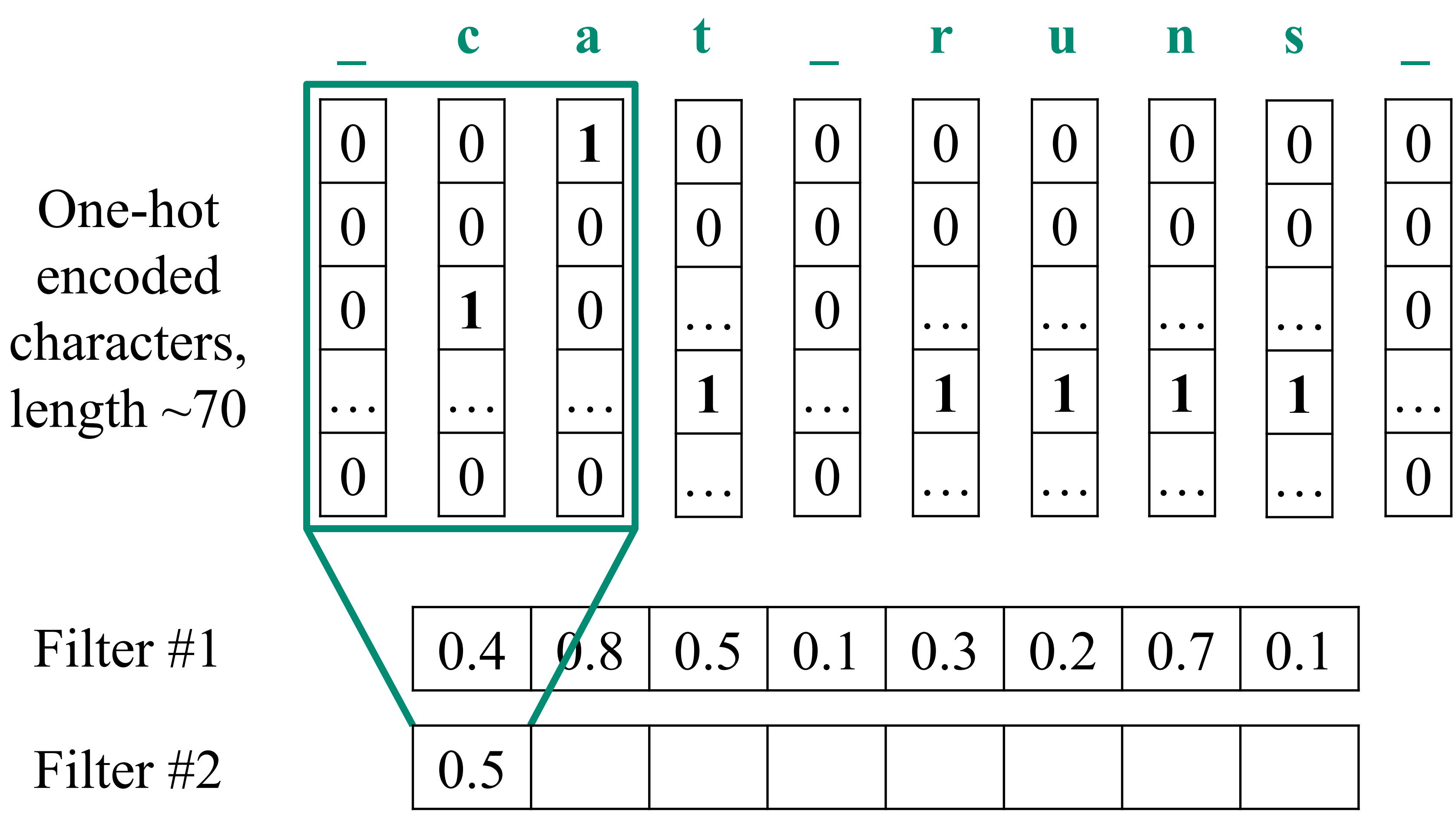
1D convolutions on characters



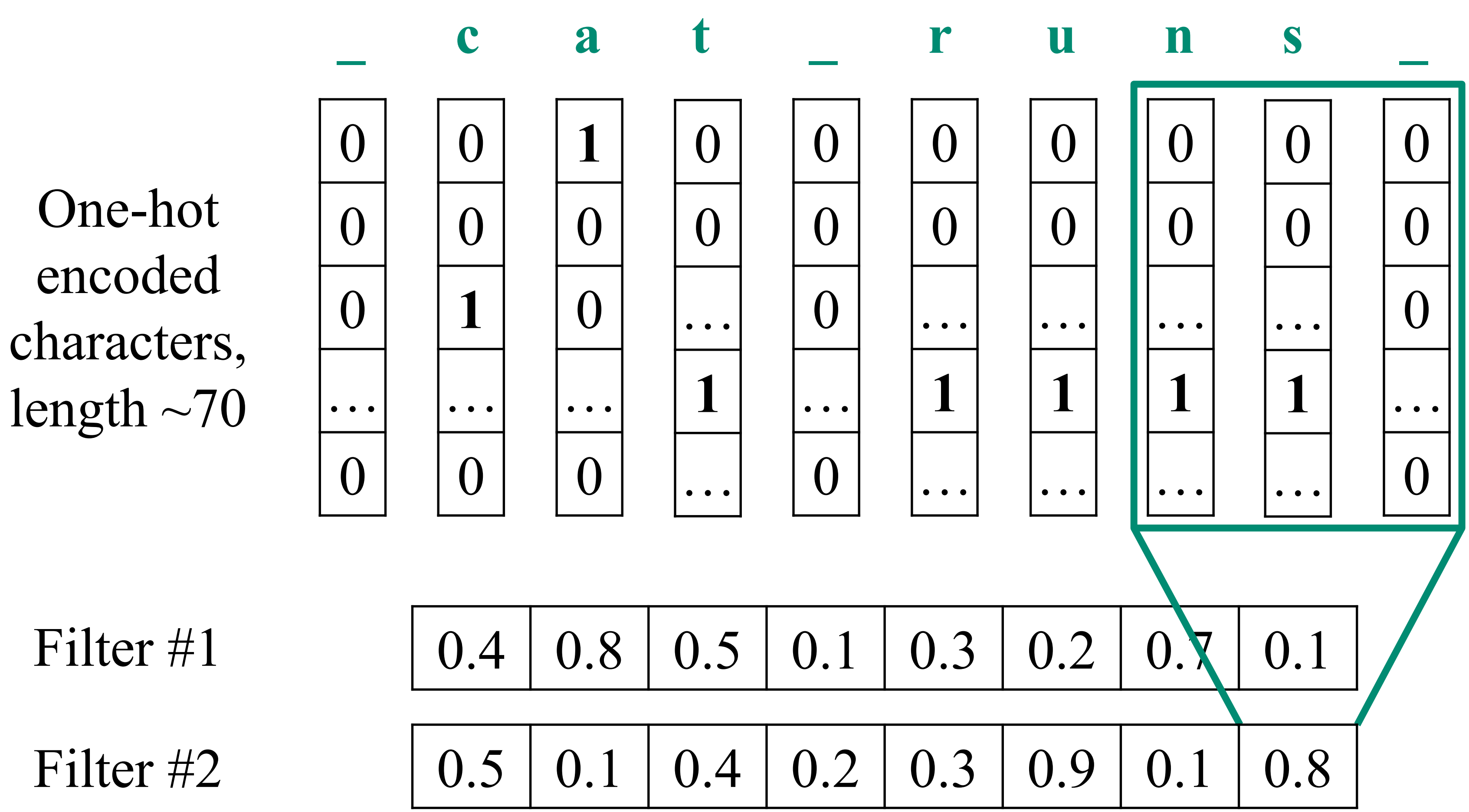
1D convolutions on characters



1D convolutions on characters



1D convolutions on characters



1D convolutions on characters

One-hot encoded characters, length ~70

_	c	a	t	_	r	u	n	s	_
0	0	**1**	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	**1**	0	…	0	…	…	…	…	0
…	…	…	**1**	…	**1**	**1**	**1**	**1**	…
0	0	0	…	0	…	…	…	…	0

Filter #1	0.4	0.8	0.5	0.1	0.3	0.2	0.7	0.1
Filter #2	0.5	0.1	0.4	0.2	0.3	0.9	0.1	0.8
Filter #3	0.4	0.7	0.3	0.7	0.5	0.5	0.9	0.4

~1024 filters

What's next? Let's add pooling!

Max pooling

Filter #1	0.4	0.8	0.5	0.1	0.3	0.2	0.7	0.1
Filter #2	0.5	0.1	0.4	0.2	0.3	0.9	0.1	0.8
Filter #3	0.4	0.7	0.3	0.7	0.5	0.5	0.9	0.4

0.8			

Max pooling

Filter #1	0.4	0.8	0.5	0.1	0.3	0.2	0.7	0.1
Filter #2	0.5	0.1	0.4	0.2	0.3	0.9	0.1	0.8
Filter #3	0.4	0.7	0.3	0.7	0.5	0.5	0.9	0.4

0.8	0.5		

Max pooling

Filter #1	0.4	0.8	0.5	0.1	0.3	0.2	0.7	0.1
Filter #2	0.5	0.1	0.4	0.2	0.3	0.9	0.1	0.8
Filter #3	0.4	0.7	0.3	0.7	0.5	0.5	0.9	0.4

0.8	0.5	0.3	0.7

Max pooling

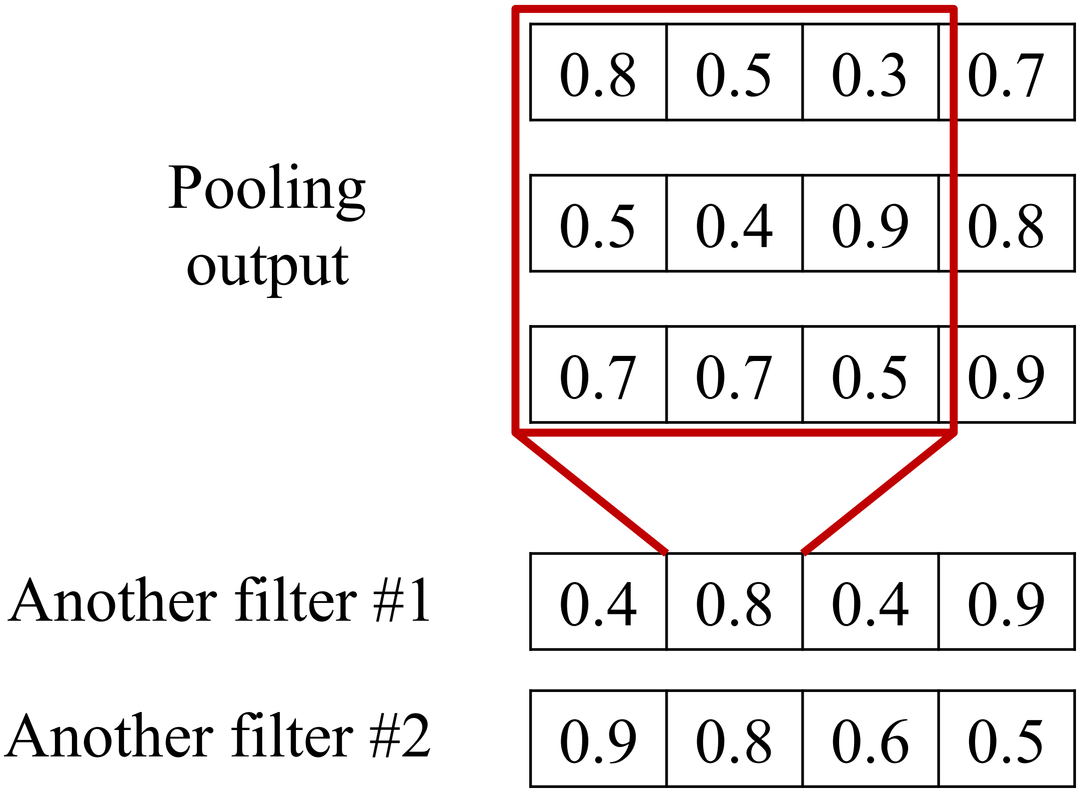
Filter #1	0.4	0.8	0.5	0.1	0.3	0.2	0.7	0.1
Filter #2	0.5	0.1	0.4	0.2	0.3	0.9	0.1	0.8
Filter #3	0.4	0.7	0.3	0.7	0.5	0.5	0.9	0.4

Pooling output

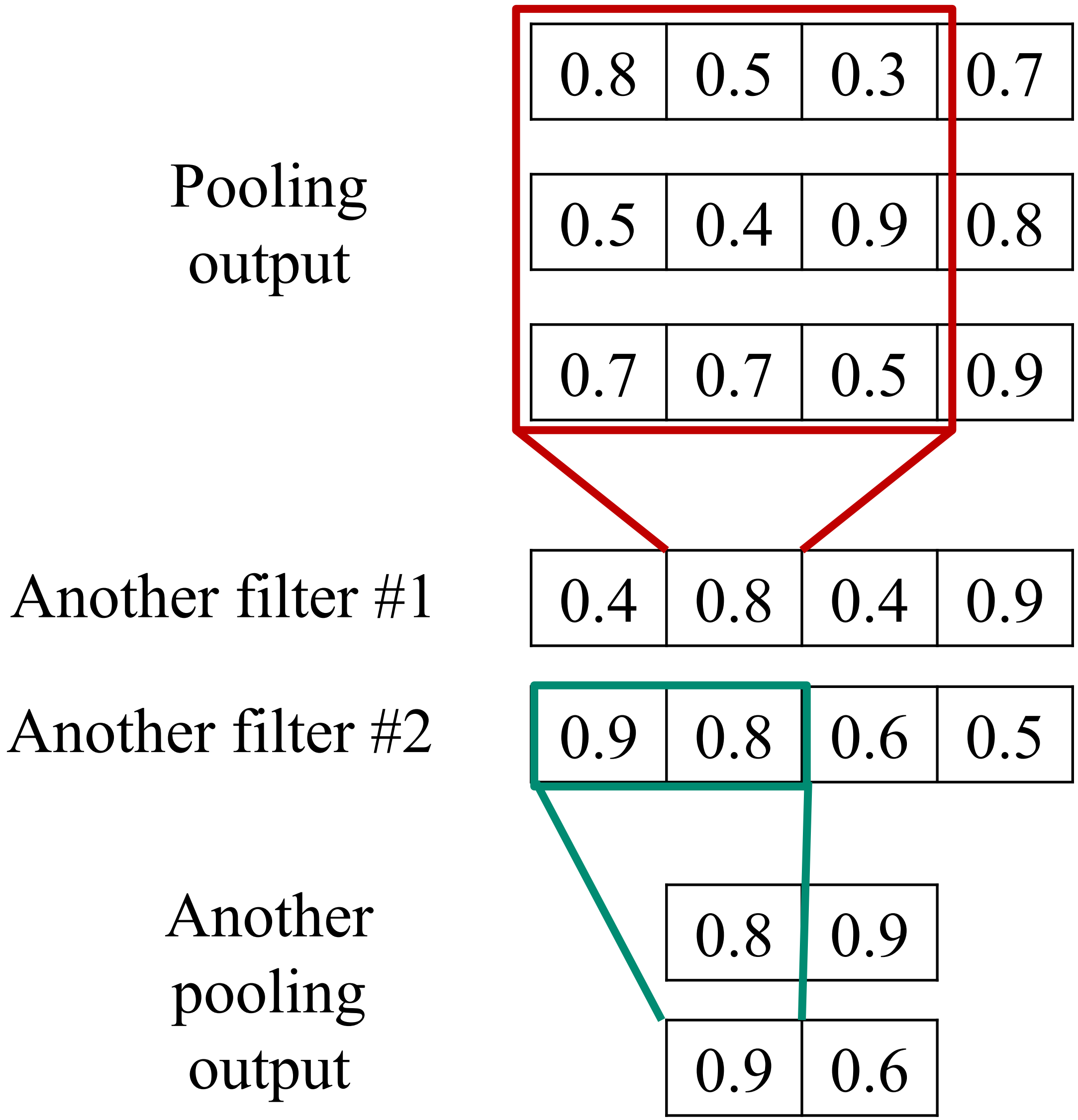
0.8	0.5	0.3	0.7
0.5	0.4	0.9	0.8
0.7	0.7	0.5	0.9

Provides a little bit of position invariance for character n-grams

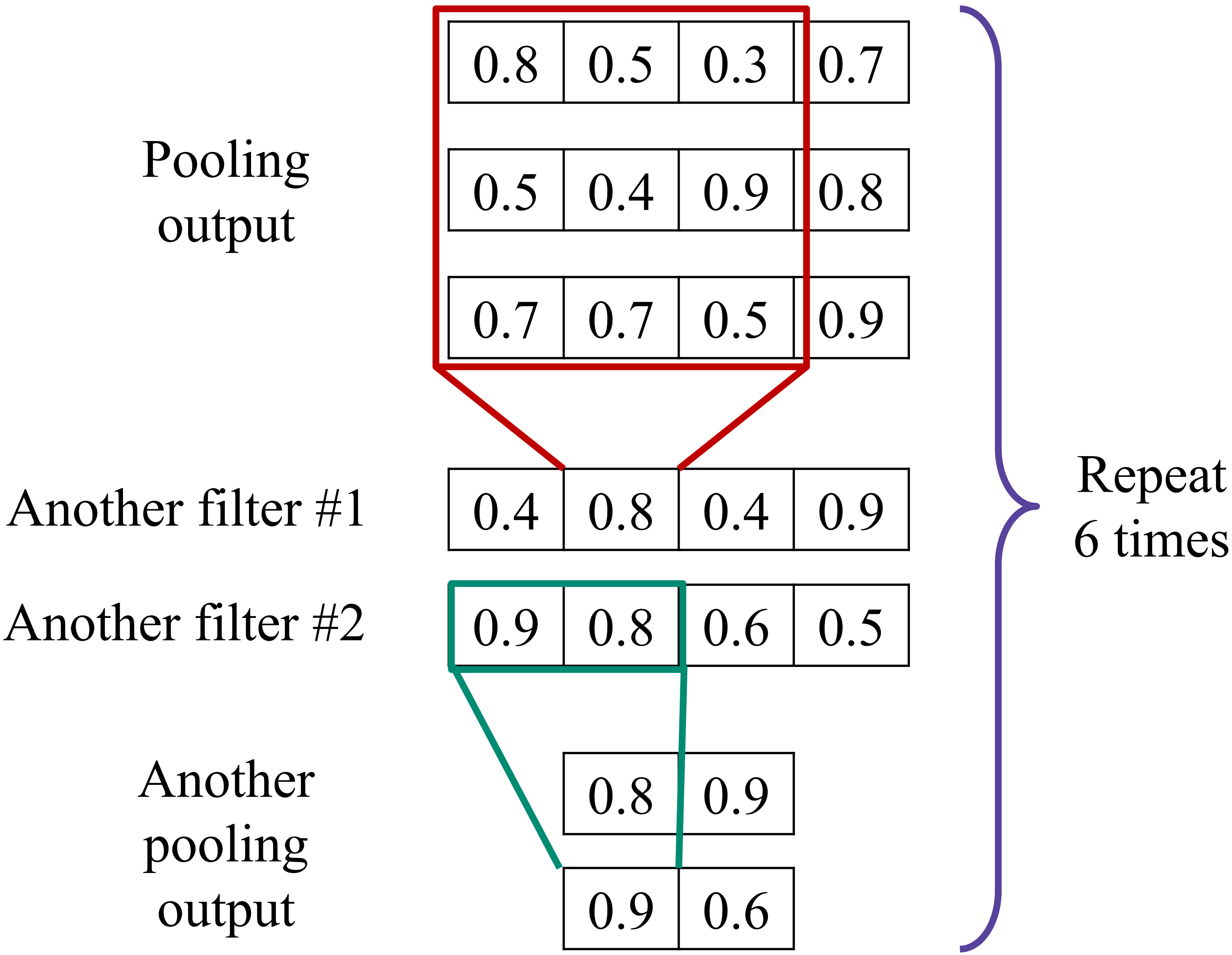
Repeat 1D convolution + pooling



Repeat 1D convolution + pooling



Repeat 1D convolution + pooling



Final architecture

- Let's take only first **1014** characters of text
- Apply 1D convolution + max pooling **6** times
  - Kernels widths: 7, 7, 3, 3, 3, 3
  - Filters at each step: 1024
- After that we have a $\mathbf{1024 \times 34}$ matrix of features
- Apply MLP for your task

https://arxiv.org/pdf/1509.01626.pdf

Experimental datasets

Categorization or sentiment analysis

	Dataset	Classes	Train Samples
Smaller	AG's News	4	120,000
	Sogou News	5	450,000
	DBPedia	14	560,000
	Yelp Review Polarity	2	560,000
Bigger	Yelp Review Full	5	650,000
	Yahoo! Answers	10	1,400,000
	Amazon Review Full	5	3,000,000
	Amazon Review Polarity	2	3,600,000

https://arxiv.org/pdf/1509.01626.pdf

Experimental results

Small dataset - classical models are better.

Errors on test set for classical models:

Model	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW	11.19	7.15	3.39	7.76	42.01	31.11	45.36	9.60
BoW TFIDF	10.36	6.55	2.63	6.34	40.14	28.96	44.74	9.00
ngrams	7.96	2.92	1.37	**4.36**	43.74	31.53	45.73	7.98
ngrams TFIDF	**7.64**	**2.81**	**1.31**	4.56	45.20	31.49	47.56	8.46

Errors on test set for deep models:

Model	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
LSTM	13.94	4.82	1.45	5.26	41.83	29.16	40.57	6.10
Sm. Full Conv.	11.59	8.95	1.89	5.67	38.82	30.01	40.88	5.78
Lg. Full Conv. Th.	9.51	-	1.55	4.88	38.04	29.58	40.54	5.51
Sm. Full Conv. Th.	10.89	-	1.69	5.42	**37.95**	29.90	40.53	5.66
Lg. Conv.	12.82	4.88	1.73	5.89	39.62	29.55	41.31	5.51
Sm. Conv.	15.65	8.65	1.98	6.53	40.84	29.84	40.53	5.50
Lg. Conv. Th.	13.39	-	1.60	5.82	39.30	**28.80**	40.45	**4.93**
Sm. Conv. Th.	14.80	-	1.85	6.49	40.16	29.84	**40.43**	5.67

Deep models work better for large datasets!

https://arxiv.org/pdf/1509.01626.pdf

Summary

- You can use convolutional networks on top of characters (called learning from scratch)
- It works best for large datasets where it beats classical approaches (like BOW)
- Sometimes it even beats LSTM that works on word level