Sentence length effects

So we prefer CNNs over LSTMs because these have a more gradual correlation than LSTMs

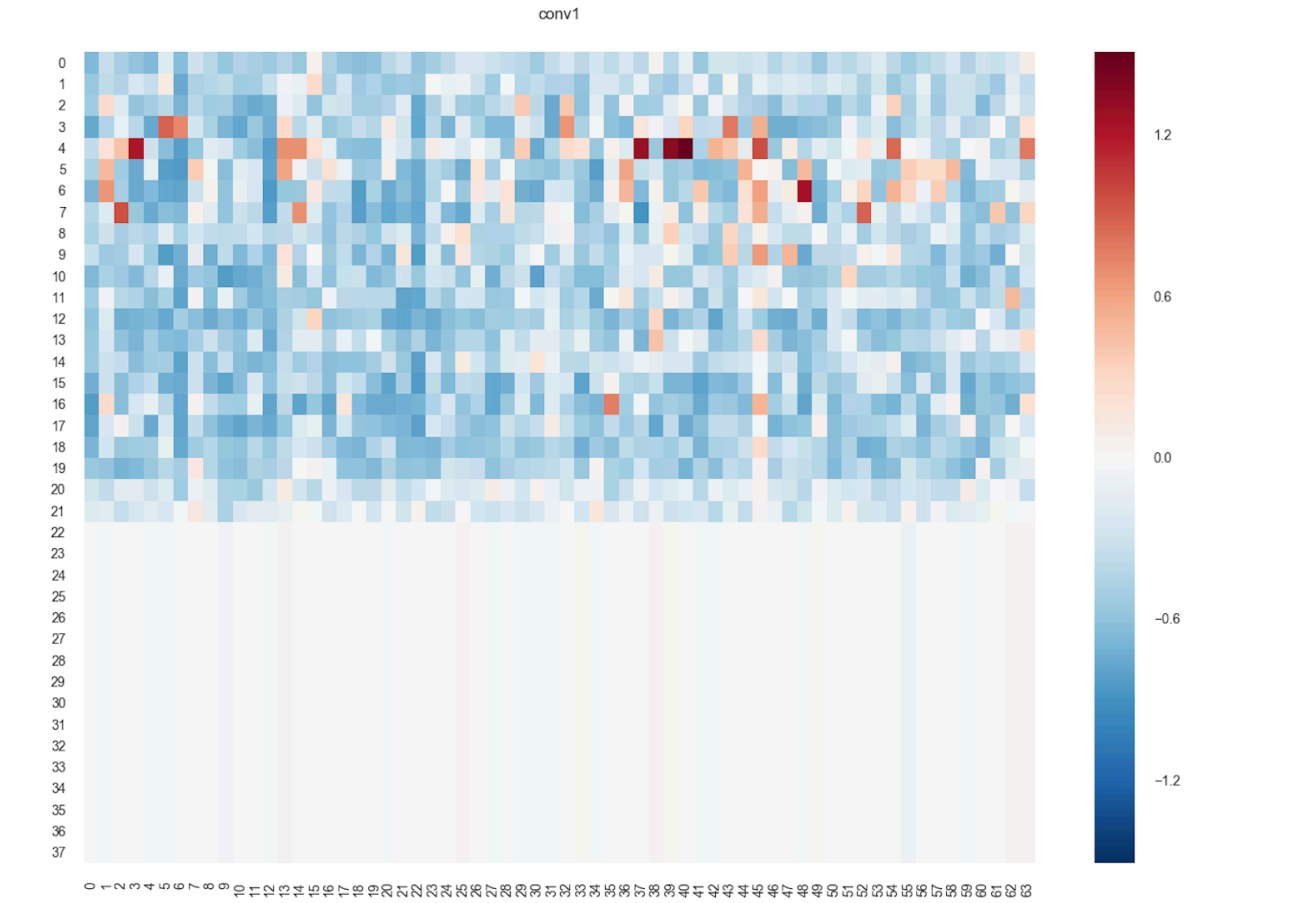
Cnn24:

	conv1	pool1	conv2	pool2	dense_final
conv1	1	0.73	0.62	0.49	0.2
pool1	0.73	1	0.8	0.73	0.44
conv2	0.62	0.8	1	0.88	0.69
pool2	0.49	0.73	0.88	1	0.7
dense_final	0.2	0.44	0.69	0.7	1

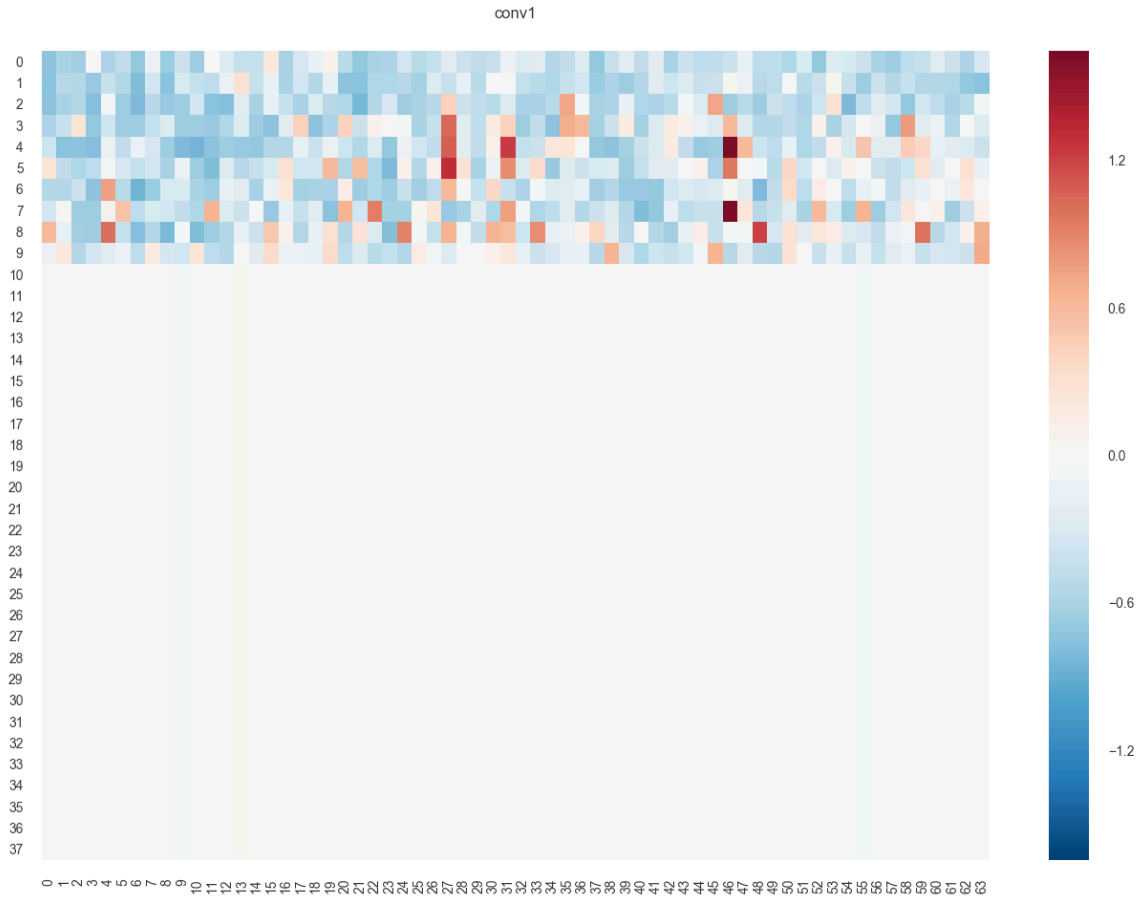
Lstm14:

	lstm1	lstm2	dense_general
lstm1	1	0.84	0.8
lstm2	0.84	1	0.92
dense_general	0.8	0.92	1

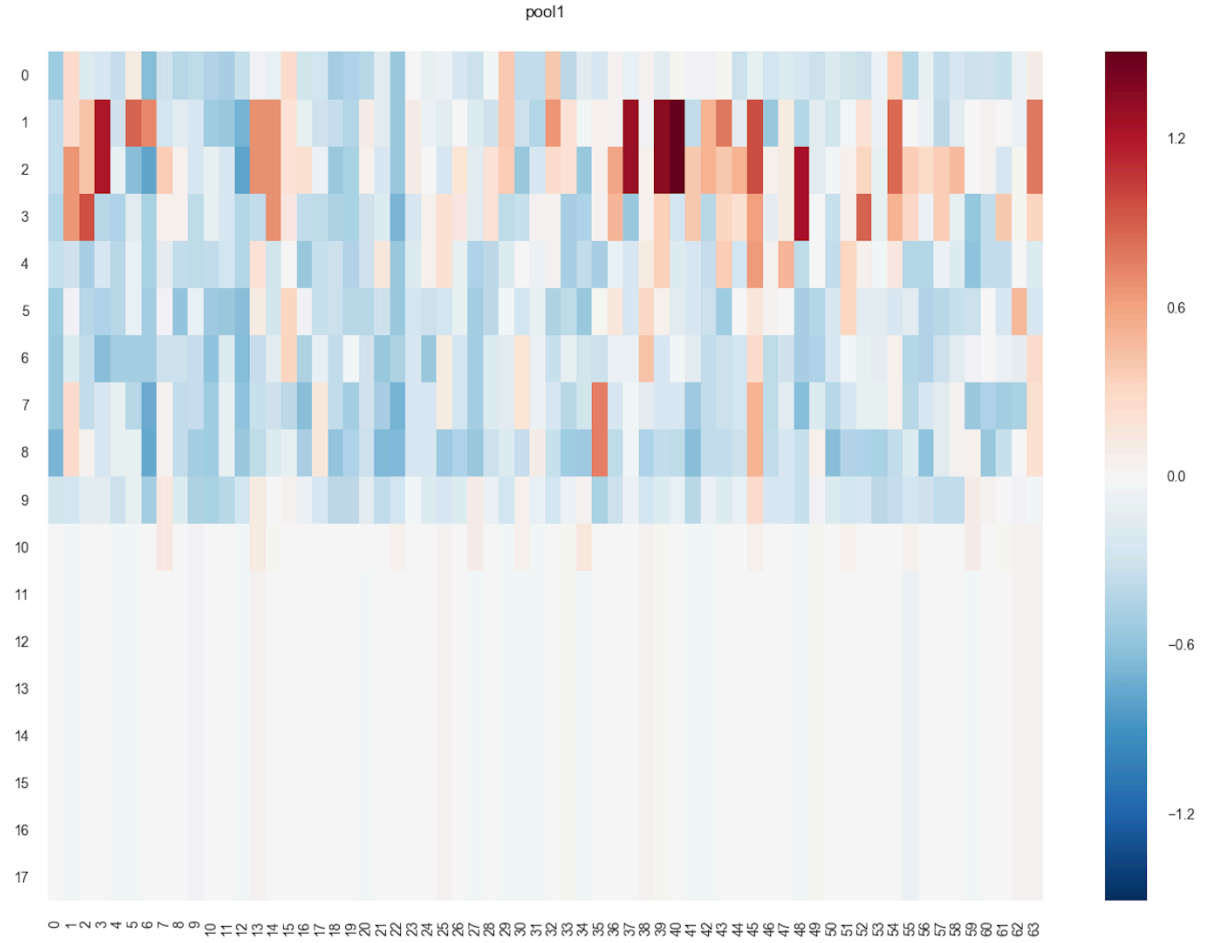
However, CNNs have this zero padding. This is conv1 for a sentence. There's a fixed length (38 words), this sentence is 22 words long. After that, all zeros as input and below you don't have exactly zero, but values close to which are the result of adding a specific bias when the network sees zeros.



Shorter sentence, more zeros:



Could we use pool layers instead? Well, they have the same issue, just less zeros:

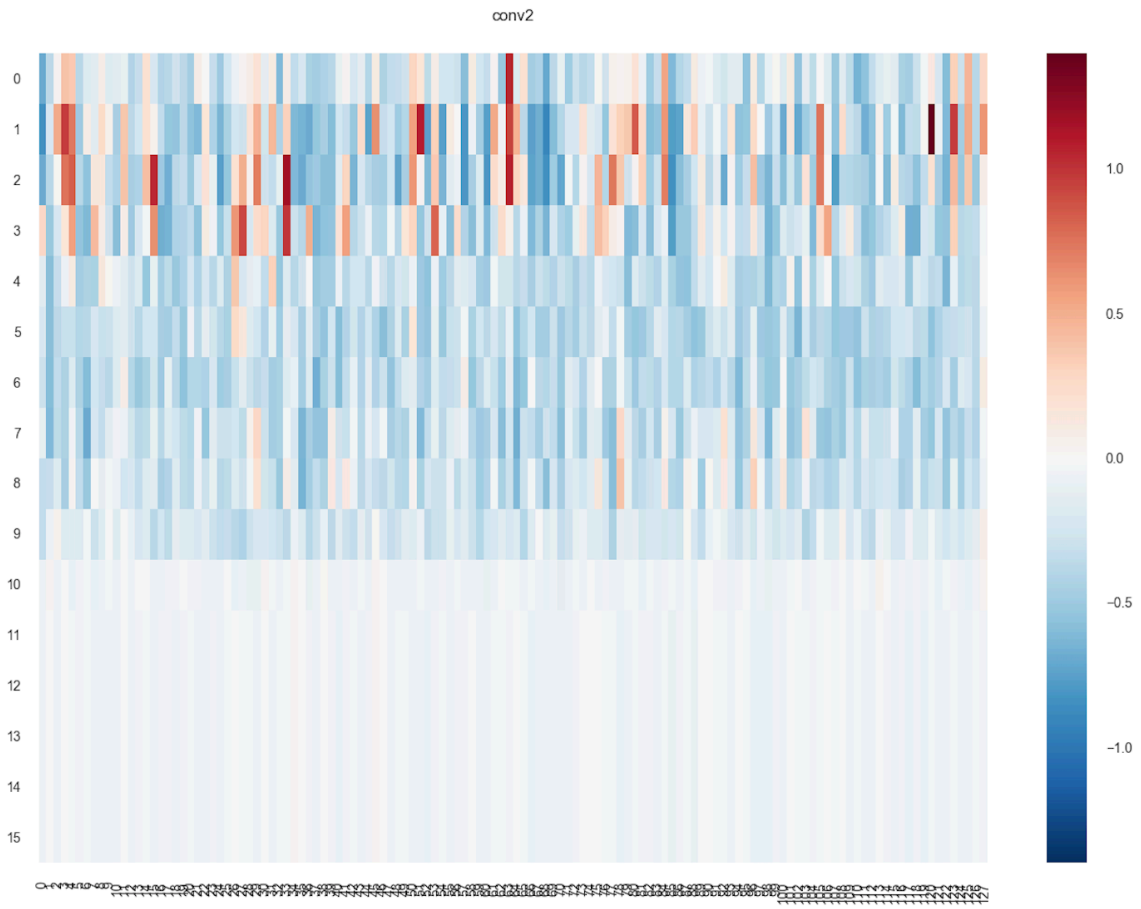


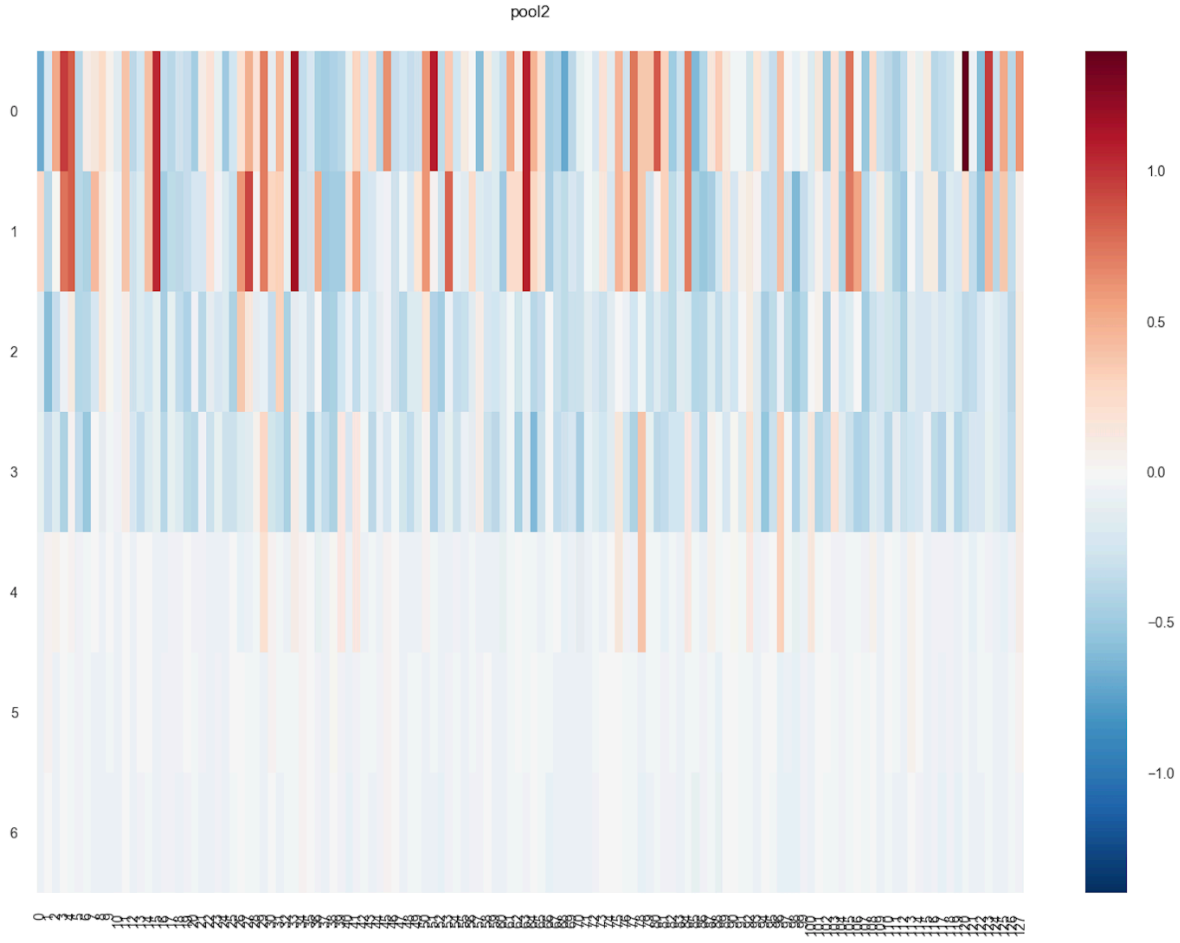
Pool size 3, stride, 2

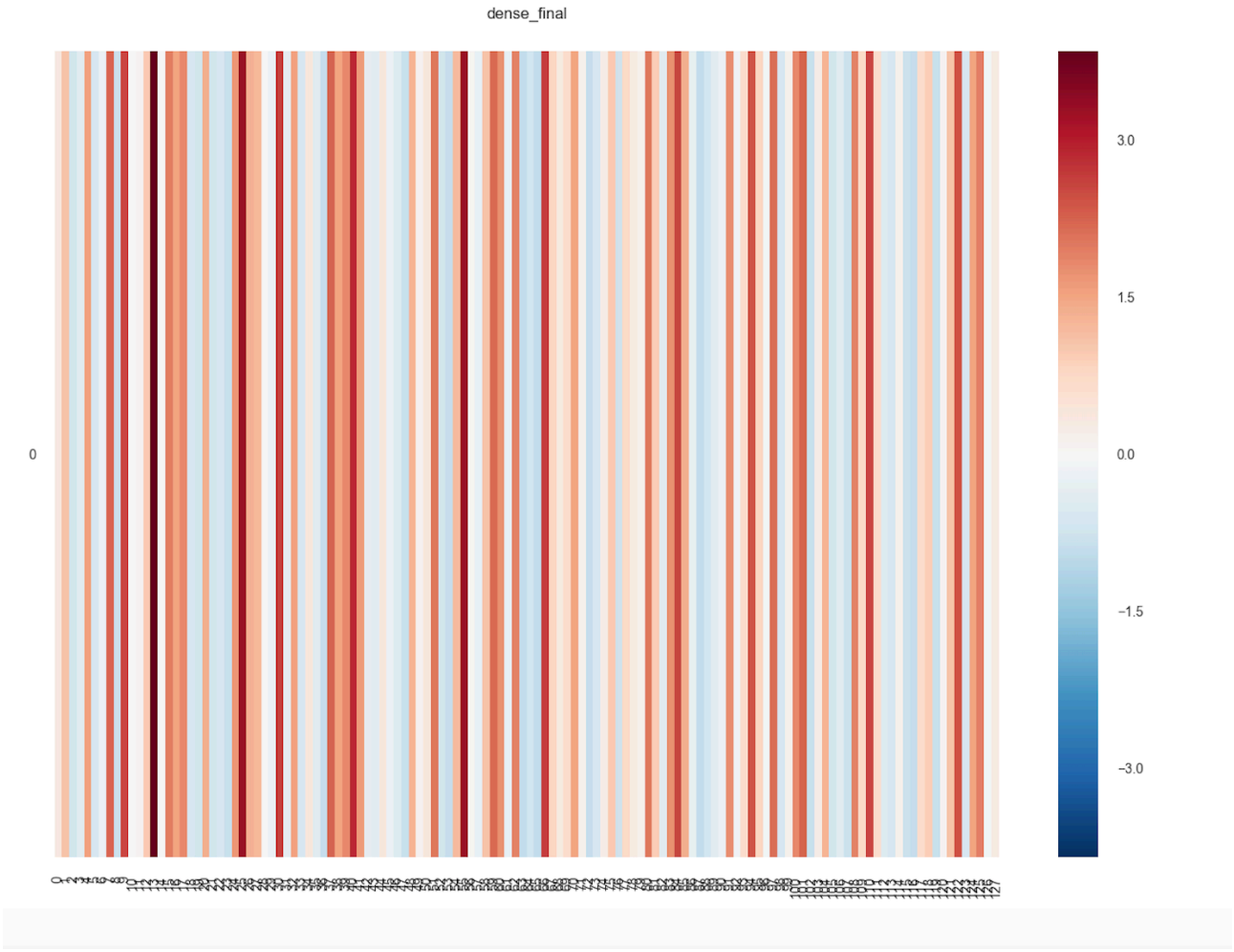
So the amount of zero elements per layer on average is:

Conv1	(67200, 36, 64)	50%	
Pool1	(67200, 18, 64)	24%	
Conv2	(67200, 18, 128)	21%	
Pool2	(67200, 9, 128)	10%	
Dense_final	(67200, 1, 128)	0%	

Here are visualization of the other layers:


conv2
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
1.0
0.5
0.0
−0.5
−1.0

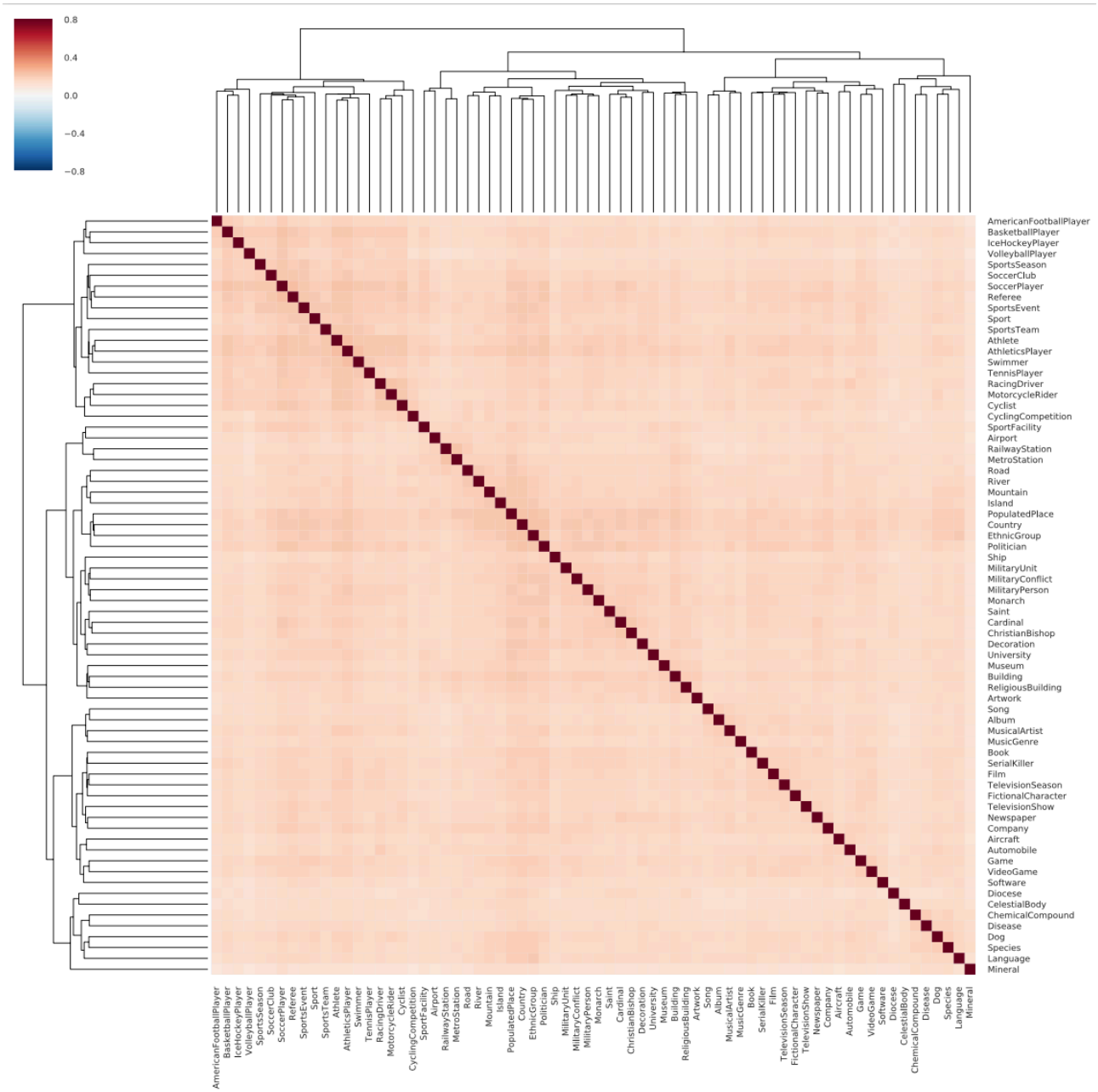

pool2
0
1
2
3
4
5
6
1.0
0.5
0.0
−0.5
−1.0


dense_final
0
3.0
1.5
0.0
−1.5
−3.0

When we correlate feature vectors, we flatten them. And we obtain all zeros at the end.

So one sentence with 5 words has many more than one sentence with 15 words.

So if you correlate sentences from conv1, we get our typical clustermap with long legs:

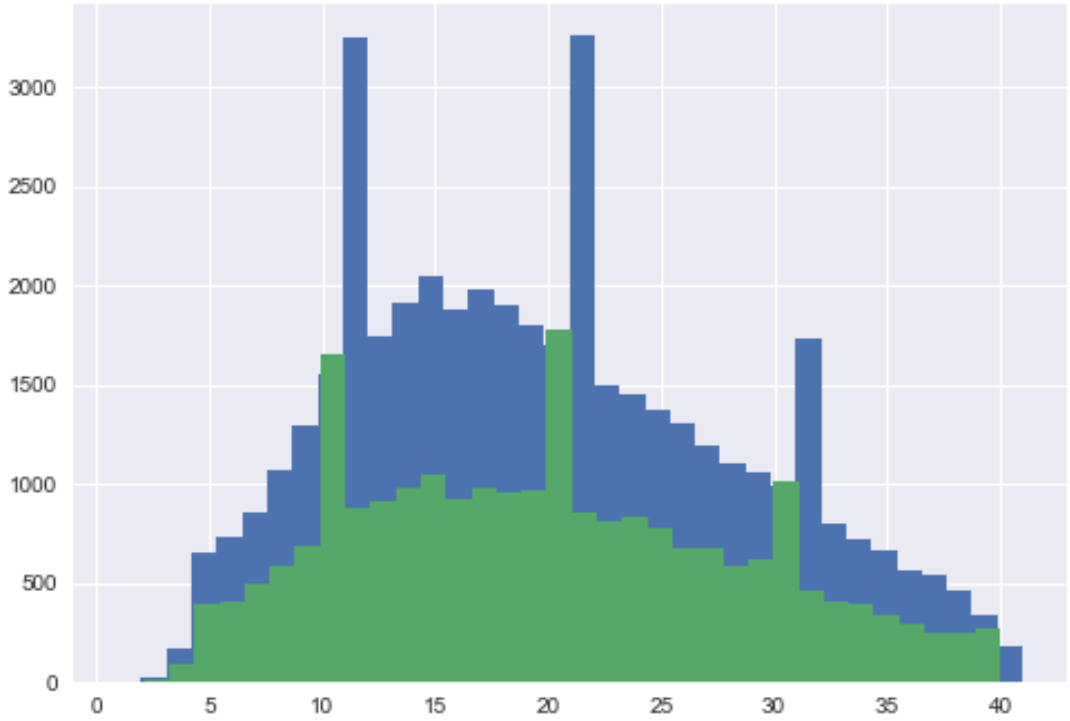


I understand feature vectores have a high correlation because of all those zeros?

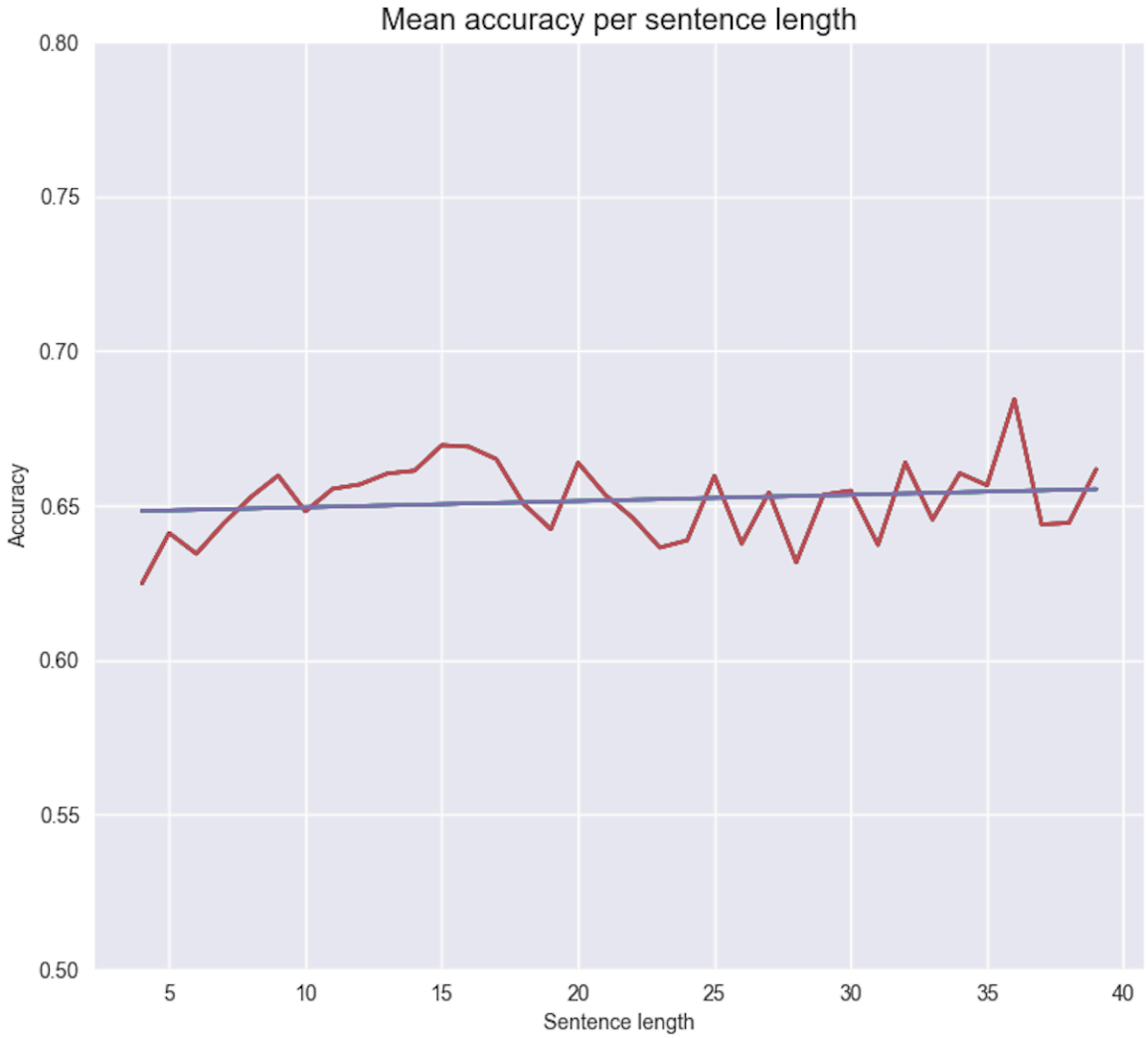
Is this vector usable?

Sentence length effects:

Correct predictions (blue) and incorrect (green) by sentence length:



Certain sentence lengths are more frequent, so if you average predictions(0 or 1) per sentence lengths (4237 for sentences with 5 words, 6432 for 6 words, etc.) you can plot the relative average accuracy by sentence length:



There’s no sentence length effect on overall accuracy. Good.

If we replace zero values with NaNs before performing correlations

With zeros (normal):

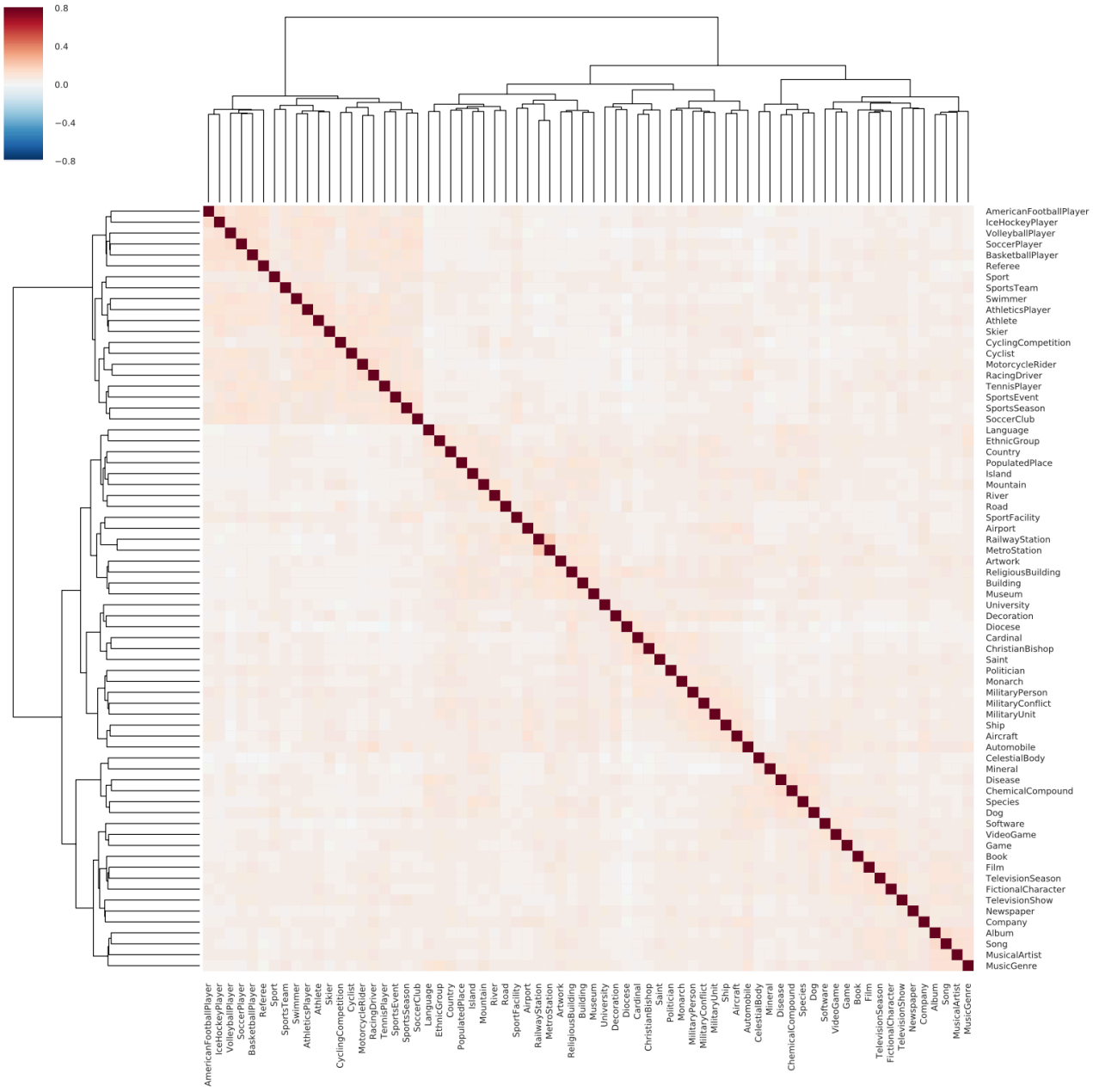
	conv1	pool1	conv2	pool2	dense_final
conv1	1	0.73	0.62	0.49	0.2
pool1	0.73	1	0.8	0.73	0.44
conv2	0.62	0.8	1	0.88	0.69
pool2	0.49	0.73	0.88	1	0.7
dense_final	0.2	0.44	0.69	0.7	1

Without zeros, with NaNs instead of zeros, You get this pattern:

	conv1	pool1	conv2	pool2	dense_final
conv1	1	0.64	0.52	0.45	0.41
pool1	0.64	1	0.81	0.74	0.68
conv2	0.52	0.81	1	0.93	0.88
pool2	0.45	0.74	0.93	1	0.93
dense_final	0.41	0.68	0.88	0.93	1

So it seems the zeros are actually driving a lot of that dissimalrity we want.

You get this RSM using NaNs:



I'm think its dropping the nans. All vectors have them. I do Spearman with this method: https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.corr.html

One can assume that all those zeros in the conv and pool layers are distorting the similarity, right?

Now if you put NaNs instead of zeros in LSTM (lstm14) you actually improve dissimalrity between layers, but I don't think this is a valid thing to do, right?

With zeros:

lstm:

	lstm1	lstm2	dense_general
lstm1	1	0.84	0.8
lstm2	0.84	1	0.92
dense_general	0.8	0.92	1

With NaNs:

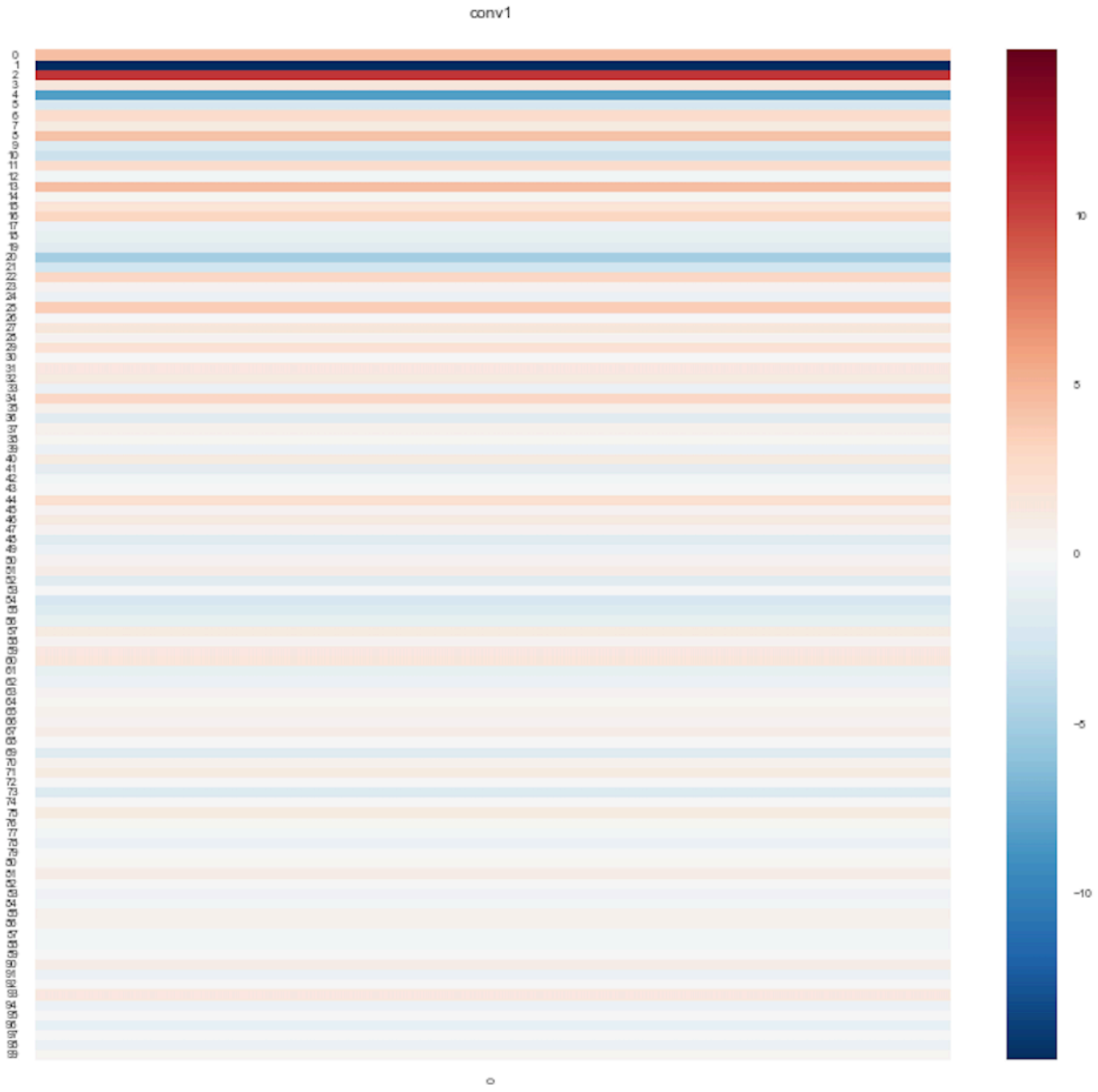
	lstm1	lstm2	dense_general
lstm1	1	0.6	0.65
lstm2	0.6	1	0.82
dense_general	0.65	0.82	1

Alternatives/solutions:

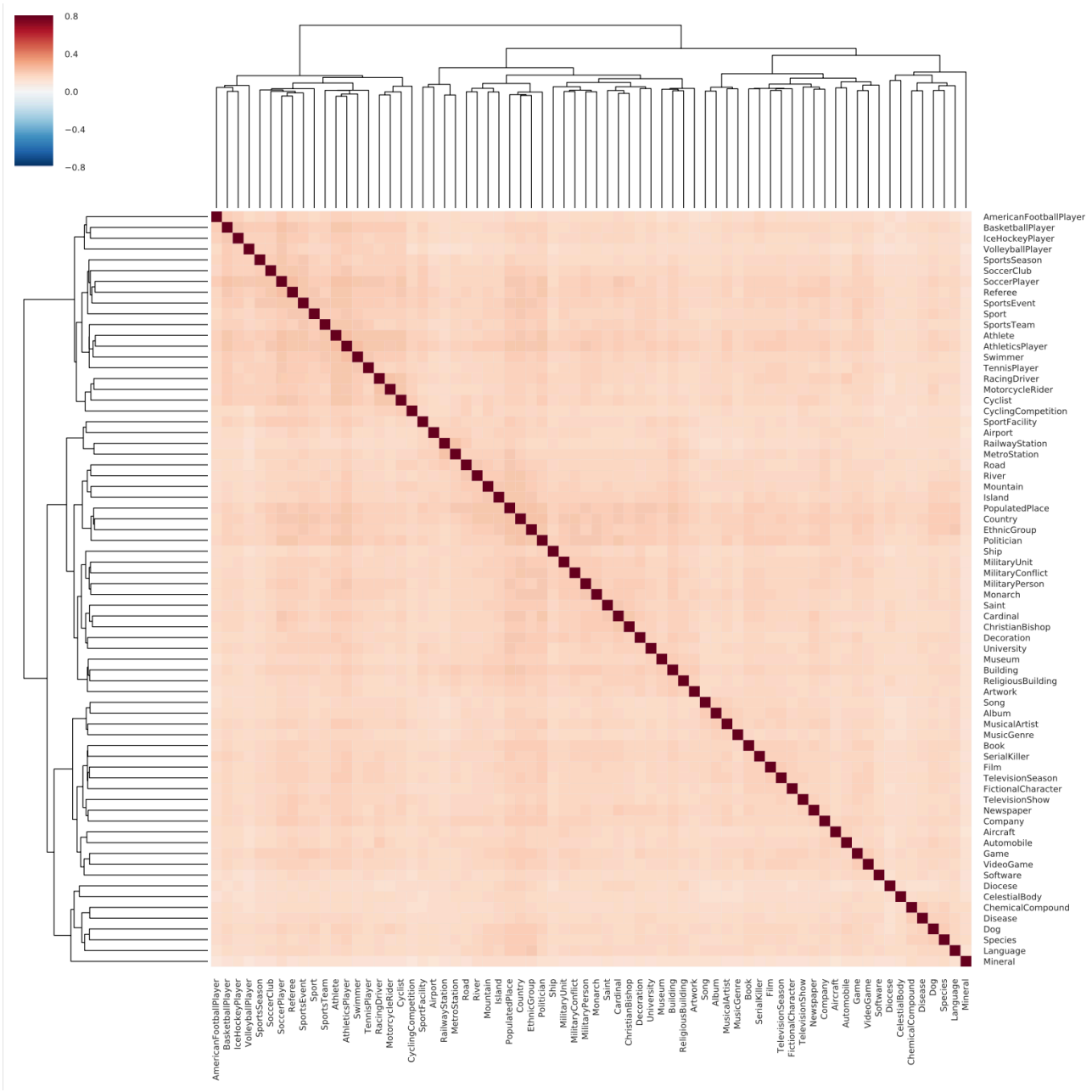
1. **Dimensionality reduction for CNN layers.**

Dimensionality reduction. Reduce feature vector conv1 (2000+ elements with 50% zeros) to 100. With 100 principal components you explain 99% of the variance for the final dense layer (128 elements) and 44% for conv1.

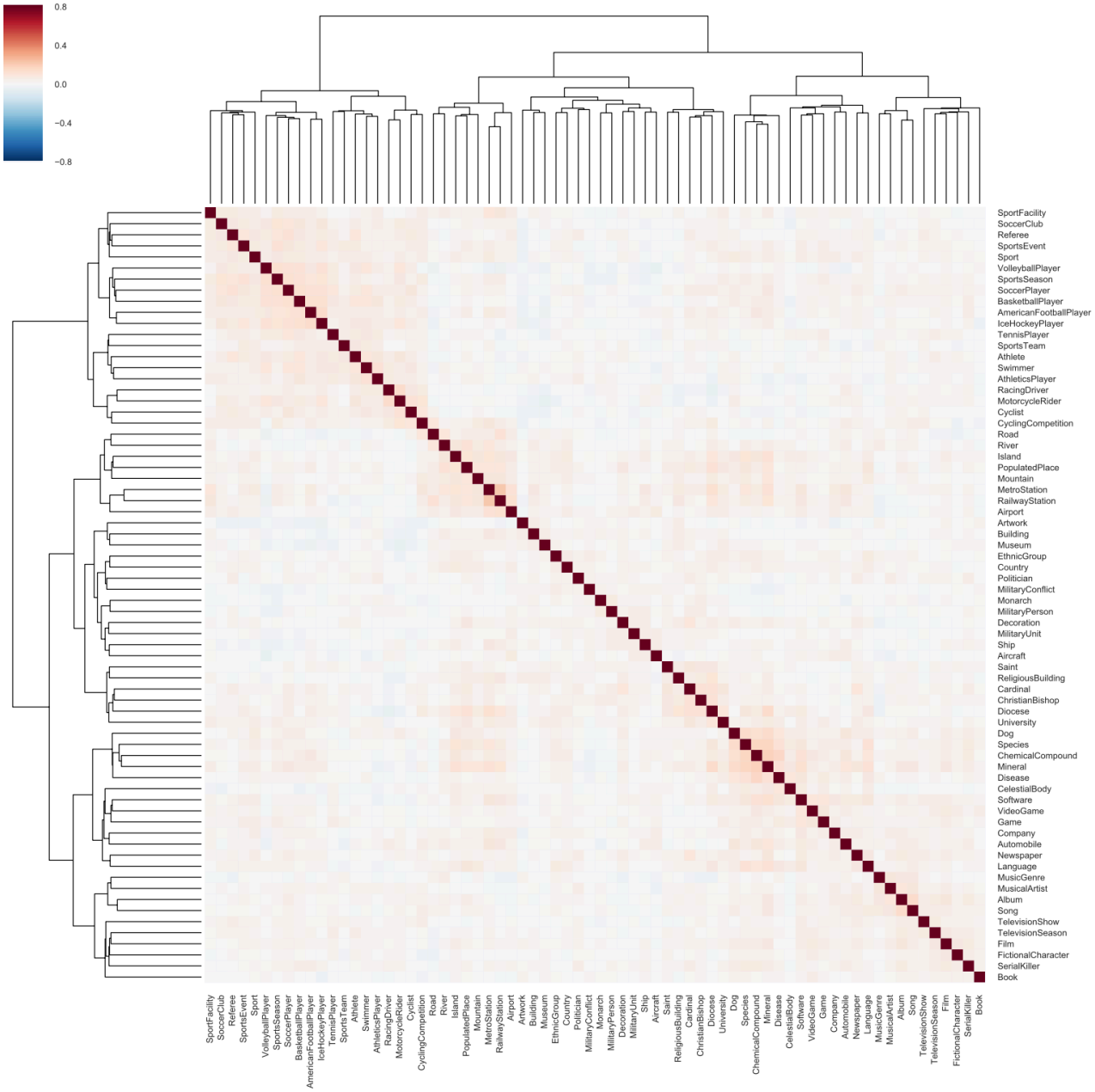
Remember the first figure for single sentence for conv1 with all those empty rows?
Well, here it is compressed to 100 elements:



Remember what conv1 looks like before compression:


0.8
0.4
0.0
−0.4
−0.8
AmericanFootballPlayer
BasketballPlayer
IceHockeyPlayer
VolleyballPlayer
SportsSeason
SoccerClub
SoccerPlayer
Referee
SportsEvent
Sport
SportsTeam
Athlete
AthleticsPlayer
Swimmer
TennisPlayer
RacingDriver
MotorcycleRider
Cyclist
CyclingCompetition
SportFacility
Airport
RailwayStation
MetroStation
Road
River
Mountain
Island
PopulatedPlace
Country
EthnicGroup
Politician
Ship
MilitaryUnit
MilitaryConflict
MilitaryPerson
Monarch
Saint
Cardinal
ChristianBishop
Decoration
University
Museum
Building
ReligiousBuilding
Artwork
Song
Album
MusicalArtist
MusicGenre
Book
SerialKiller
Film
TelevisionSeason
FictionalCharacter
TelevisionShow
Newspaper
Company
Aircraft
Automobile
Game
VideoGame
Software
Diocese
CelestialBody
ChemicalCompound
Disease
Dog
Species
Language
Mineral

Here is conv1 when you reduce the dimensions from 2000+ to 100:



the clusters are more clear, the distance is less “leggy”. You retain 99% of the variance for the last dense_final layer, but only 44% of conv1.

See ./rsm_pca/ for all RSMs.

Correlations between compressed vs. non-compressed RSMs:

Conv1	0.98
Pool1	0.98
Conv2	0.99
Pool2	0.99
Dense_final	0.99

The clustering can have small differences, but they’re basically the same.

I could also try TSNE.

Correlation between layers with PCA:

	conv1	pool1	conv2	pool2	dense_final
conv1	1	0.51	0.44	0.37	0.29
pool1	0.51	1	0.57	0.52	0.4
conv2	0.44	0.57	1	0.68	0.48
pool2	0.37	0.52	0.68	1	0.51
dense_final	0.29	0.4	0.48	0.51	1

Solution 2: Use LSTM with mask_zero

I ran a new LSTM (lstm15) masking the zeros, something you can't do in CNNs:

With zeros:

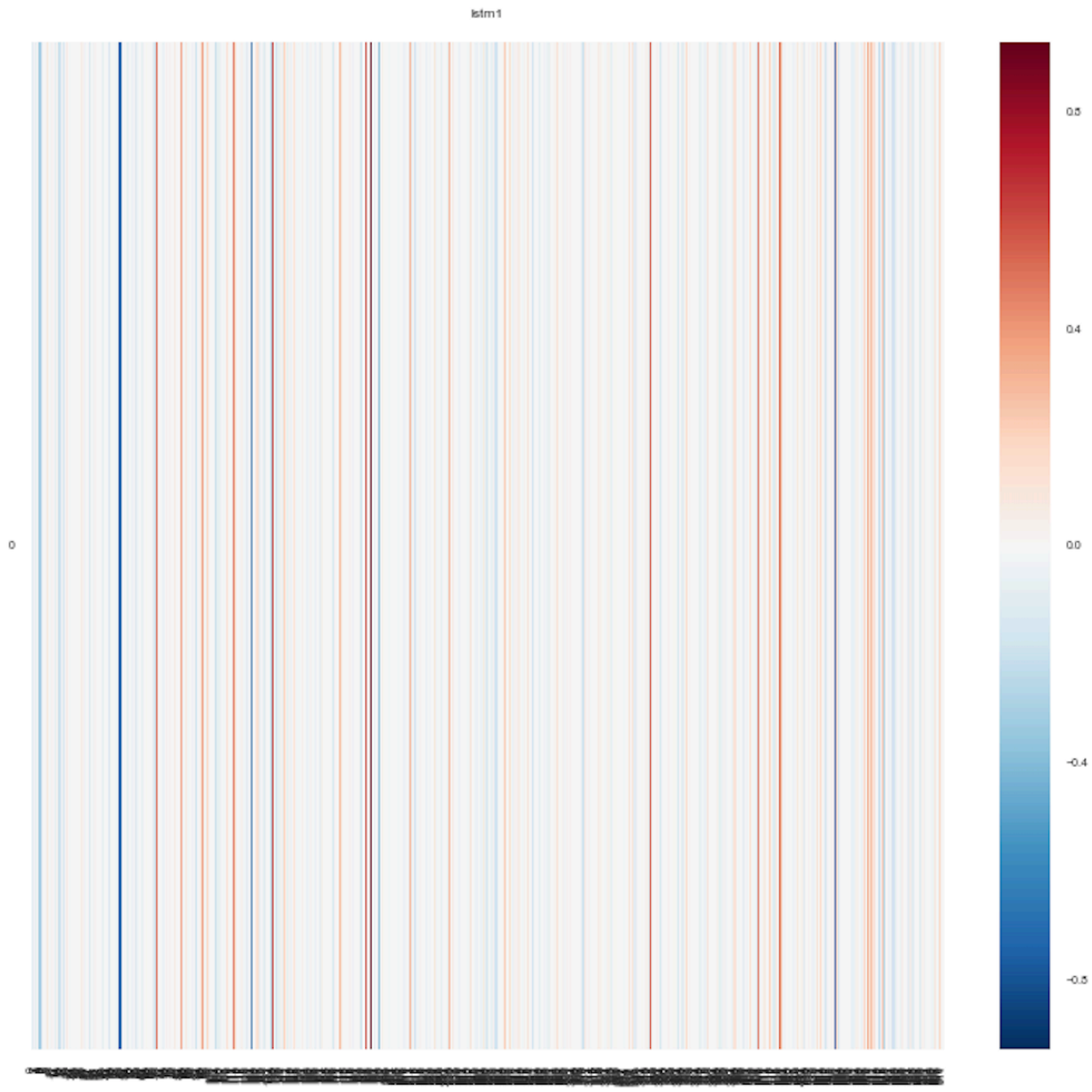
	lstm1	lstm2	dense_general
lstm1	1	0.84	0.8
lstm2	0.84	1	0.93
dense_general	0.8	0.93	1

Without zeros, with Nans:

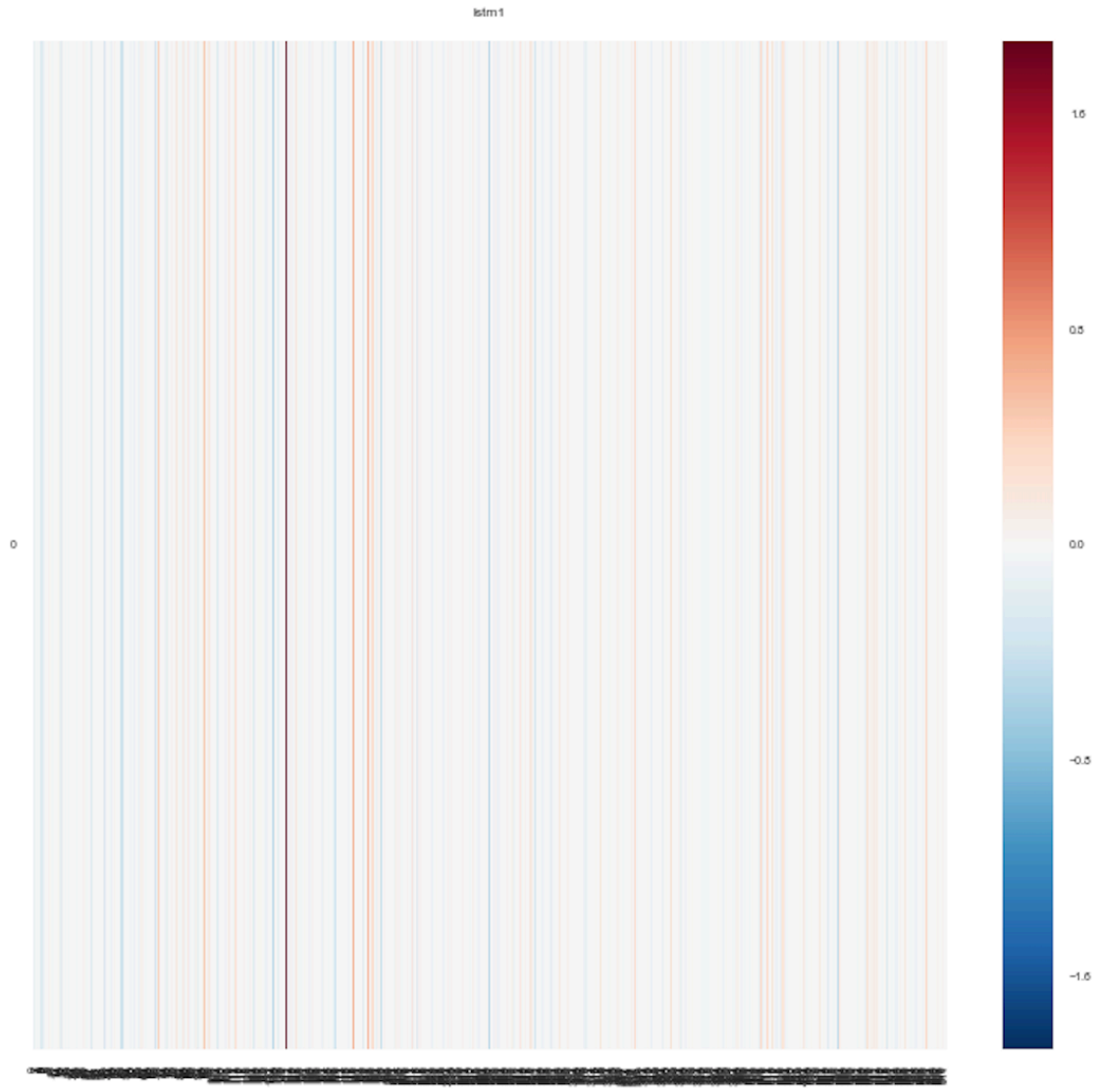
	lstm1	**lstm2**	**dense_general**
lstm1	**1**	**0.53**	**0.61**
lstm2	**0.53**	**1**	**0.78**
dense_general	**0.61**	**0.78**	**1**

It doesn't change this aspect regarding lstm14.

Now lets look at a single sentence lstm14, lstm1 (layer1):



Same sentence in LSTM15, masking the zeros:



There's a difference. They correlate at 0.58.

And of course, the zeros matter in the first layer. Ask you go to the second layer or final layer, the difference is minor:

Lstm2: 0.81
Dense_final: 0.91.

And this is standard practice to mask the zeros.

Questions:

- Should we do PCA with CNNs or lstm? I'd go with PCA. Cichy uses it at one point to make a RDM of all layers in one so he reduces them to 117 to make them all fit.
- Is it okay to put NaNs instead of the zero values?