Multi-output models

1. Methods

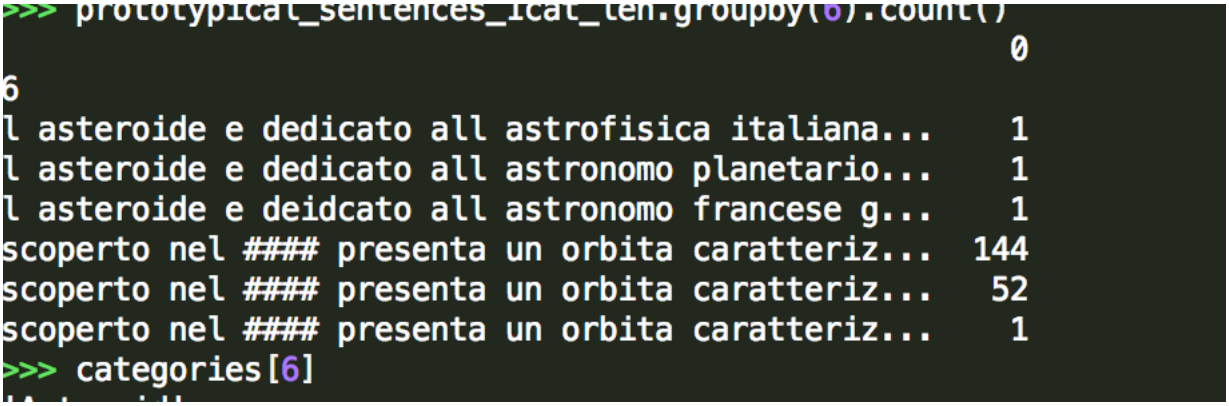
1.1. Dataset

Perhaps you want to see what I sent you last week: april_27_italian_full_article_dataset, but it's not necessary.

I Removed 4 categories. Website was performing <2% accuracy because all its articles were on different topics. Then I removed Asteroid, TennisTournament, and Wine. There aren't at least 10 prototypical sentences between 8 and 16 words. So we wouldn't have stimuli.

Here you can see that certain *long* sentences appear hundreds of times for Asteroid just changing the numbers/years (represented by # signs). 30% of all sentences in Asteroid is this one:
'scoperto nel #### presenta un orbita caratterizzata da un semiasse maggiore pari a ######## ua e da un eccentricita di ######## inclinata di ###### rispetto all eclittica'

This is because the pages are probably created by bots. So these classes performed very high (~95%) but they don't have any variance/fuzziness and most importantnly they don't have short prototypical sentences from which to extract stimuli.



For TennisTournament and Wine, it's not the exact same sentence, but they're all too long and very similar. You can see these sentences here: repeated_sentences_3_categories.txt

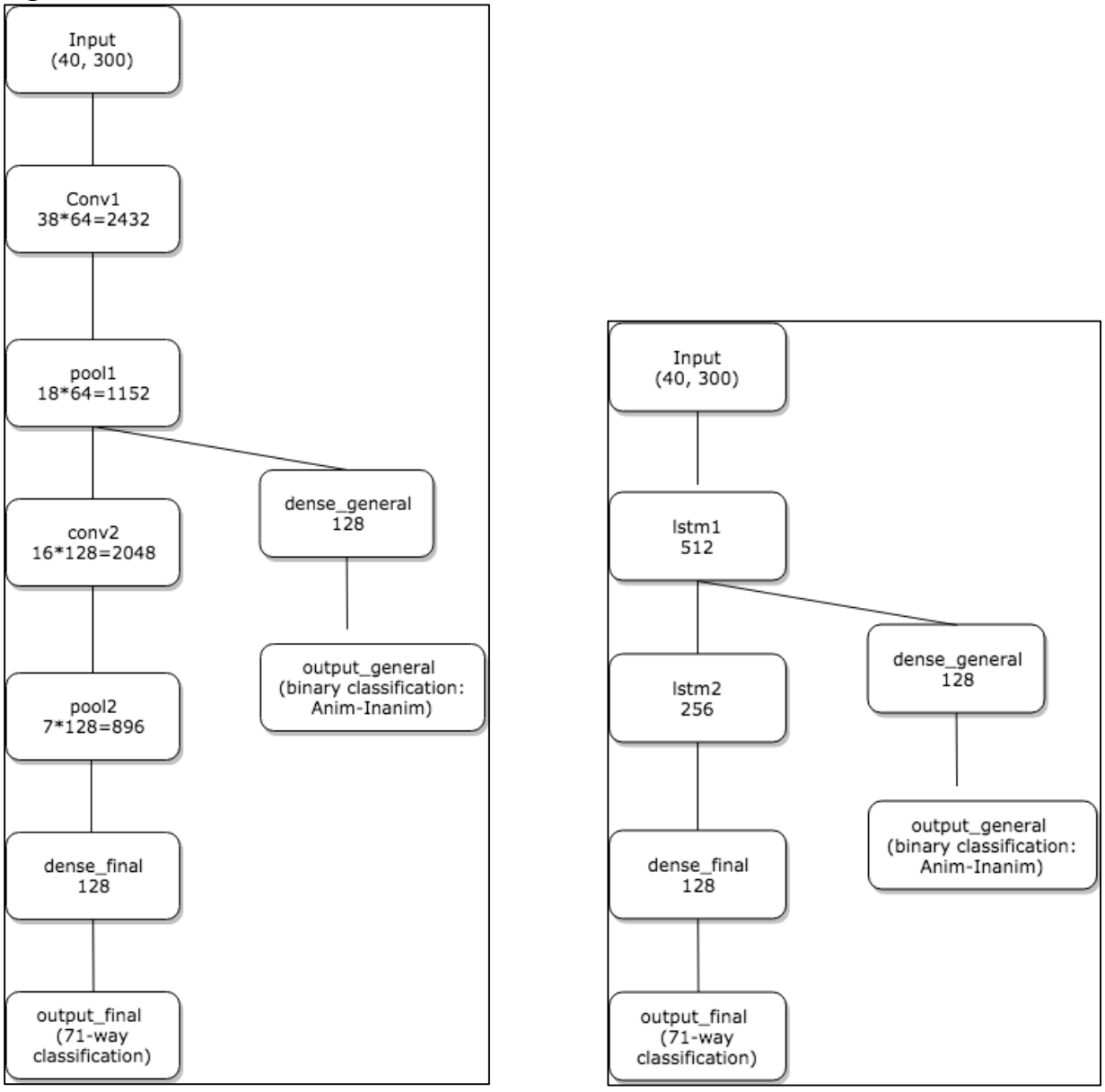
And compare to the rest of the categories: other_sentences_71_categories.txt

So our final dataset is 71 classes of 6000 sentences each.

1.2. **Multi-output models**

I tuned the *first layer* to high-level categories Animate and Inanimate and the rest of the layers (i.e., including the *final layer)* to the DBpedia (low-level) classes. This is done by adding a first loss function after layer 1 and weighing it N times more important than the main loss function. See **Figure 1**.

Figure 1



a) CNN_multitask b) LSTM_multitask

I was calling these multitask models, because multitask models have more than one task (e.g., classifying a sentence and POS tagging the words) to more than one outputs (one compares predictions with the labels from one task and the other output compares prediction for the other task). In our case, we have the same task for both (i.e., text classification) and we have to types of text classification: 1 in 71 and 1 in 2. Not sure this

The reason I want to do this is:

1. it will be a specific difference between the layers that we will understand and be able to measure (not like the typical black boxes neural networks create), which is we are tuning layer 1 to those general categories of animate-inanimate and the output layer is more tuned to the more low-level classes of DBpedia. And we will be able to measure how high-level or low-level the layers are with SVMs with dense_general and dense_final vectors for high-level or low-level classifications (see 2.2.).
2. we should obtain a big difference between the RSMs of first layers (conv1, pool1, dense_general) and last layer (dense_final). And pool2 will be intermediate.

To do this, I assigned a Animate or Inanimate to each DBpedia class (e.g., DBpedia class SoccerPlayer is animate, ReligiousBuilding is inanimate).

.

1.3. Doubts with labelling these:

0=Inanimate
1=Animate

You can see sentence examples in the dataset I sent you a while ago in English or in **other_sentences_71_categories.txt** (ctrl-F "Class------")

Company	0	?
Country	0	?
Diocese	0	?
Disease	0	?
MilitaryUnit	1	?
SportsTeam	1	?

For all labels, see **general_labels.csv**

1.4. Models

I ran many different models. I'm going to show you four models:

- **Cnn20_multitask:** one using multi-output (like in Figure 1, learns from both outputs) vs.
- **Cnn21:** exactly the same architecture, but only uses one loss function like before (it has the extra branch but doesn't learn from that output, it's just to compare as a control).
- **Lstm11_multitask** one using multi-output (like in Figure 1, learns from both outputs) vs.
- **Lstm12:** exactly the same architecture, but only uses one loss function like before (it has the extra branch but doesn't learn from that output, it's just to compare as a control). The LStm isn't smaller, it's just that the CNN always computes a convolution followed by maxpooling. I'm comparing conv layers with pool layers to see which we end up using. Conv layershave richer features but can be large and redundant. Maxpool takes the most important features.

2. Results

2.1. Performance

CNN20_multitask
output_general_acc: 0.79 (chance=0.5, 1 in 2)
output_final_acc: 0.53 (chance=0.01, 1 in 71)

This means that the the multitask models does pretty well on both tasks.

CNN21
output_general_acc: 0.46
output_final_acc: 0.57

When you don't tune the first output_layer, then you get a chance result (0.46, obviously). And the final output layer does even better because it doesn't have to do two tasks

LSTM11_multitask
output_general_acc: 0.84
output_final_acc: 0.65

Same, just that the lstm I'm using performs better (I can improve both models more, so these relative results aren't final, just this tendency).

output_general_acc: 0.48
output_final_acc: 0.69

So here we see that 1. Multitask models effectively can learn animate-inanimate distinction much better than the non-multitask model.

I weighted the animacy task 100 x the DBpedia classification task.
If you weight less important like 10x1 you get similar results (but with improved overall accuracy):

CNN17_multitask
output_general_acc: 0.8
output_1_acc: 0.56

But we'll see the 100x1 ratio is important for another metric (SVM, 2.2.)

2.2. Correlation between sentences in different layers

Steps:
1. Find prototypical sentences of categories: the ones that, on average, correlate the most with other sentences of their category.
2. Choose 20 random sentences from top 200 prototypical sentences per category
3. Correlation matrix of 20sentences*71categories = matrix 71*71
4. Correlate correlation matrix of each layer (method=Spearman)

Cnn20_multitask (4 layers we could use in bold)

	conv1	pool1	dense_general	conv2	pool2	dense_final

conv1	1.00	0.60	0.12	0.56	0.44	0.17
pool1	0.60	1.00	0.30	0.68	0.66	0.34
dense_general	0.12	0.30	1.00	0.43	0.41	0.50
conv2	0.56	0.68	0.43	1.00	0.83	0.67
pool2	0.44	0.66	0.41	0.83	1.00	0.67
dense_final	**0.17**	0.34	**0.50**	**0.67**	0.67	**1.00**

Cnn21

	conv1	pool1	dense_general	conv2	pool2	dense_final
conv1	1.00	0.61	0.11	0.52	0.40	0.13
pool1	0.61	1.00	0.45	0.65	0.63	0.31
dense_general	0.11	0.45	1.00	0.40	0.33	0.39
conv2	0.52	0.65	0.40	1.00	0.81	0.68
pool2	0.40	0.63	0.33	0.81	1.00	0.65
dense_final	**0.13**	0.31	**0.39**	**0.68**	0.65	**1.00**

Lstm11_multitask

	lstm1	dense_general	lstm2	Dense_final
lstm1	1.00	0.73	0.74	0.62
dense_general	0.73	1.00	0.67	0.54
lstm2	0.74	0.67	1.00	0.87
Dense_final	**0.62**	**0.54**	**0.87**	**1.00**

Here lstm1 correlates more with dense_final perhaps because the auxiliary loss function affects the dense_general layer more than the lstm1 layer. But this doesn't seem to happen in CNNs, perhaps because dense_general is a 128-element vector, and conv1 and pool2 are large matrices, so this may affect the correlations.

	lstm1	dense_general	lstm2	Dense_final
lstm1	1.00	0.57	0.78	0.75
dense_general	0.57	1.00	0.62	0.58
lstm2	0.78	0.62	1.00	0.91
dense3	**0.75**	**0.58**	**0.91**	**1.00**

From this alone, I'd go with CNNs since they are more gradual.

We should get similar results with our final stimuli, because they will also be hi in prototypicality.

2.3. Support Vector Machine

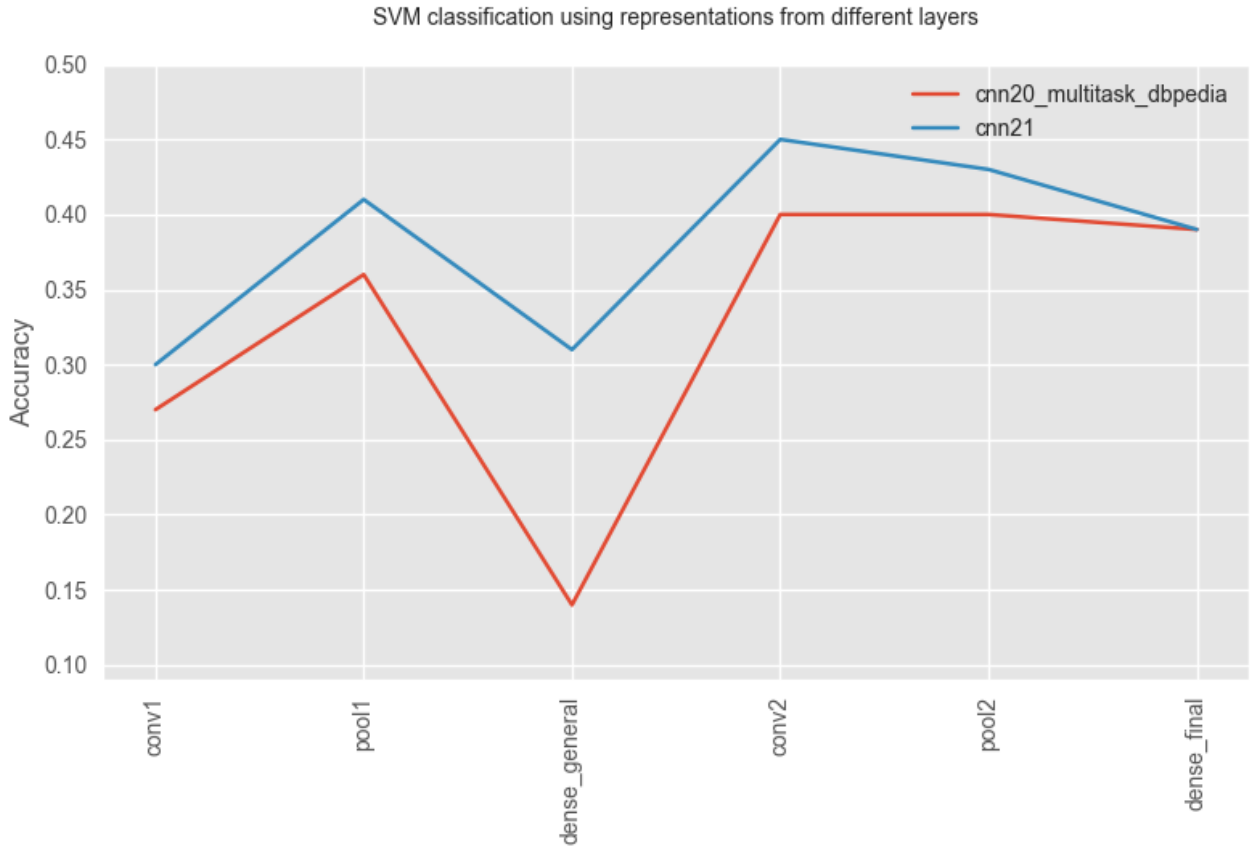
1. Pass a couple of hundred sentences on the trained model to obtain feature vectors at each layer

2. The feature vector from the last layer is tuned to 1 in 71 dbpedia class classification, so one can assume that the ***last*** feature vector is representing the category of the sentence (e.g., SoccerPlayer, Dog). How can we prove this empirecally? We can use the feature vector to train an SVM to classify sentences into the 71 classes. One hypothesizes that the last layer will do better than the first layer. We'll call this "SVM for dbpedia categorization":

Result:

Figure 2. Cnn20_multitask vs cnn21 for dbpedia 71 way classification

Cnn20 [0.27, 0.36, 0.14, 0.4, 0.4, 0.39]
Cnn21 [0.3, 0.41, 0.31, 0.45, 0.43, 0.39]

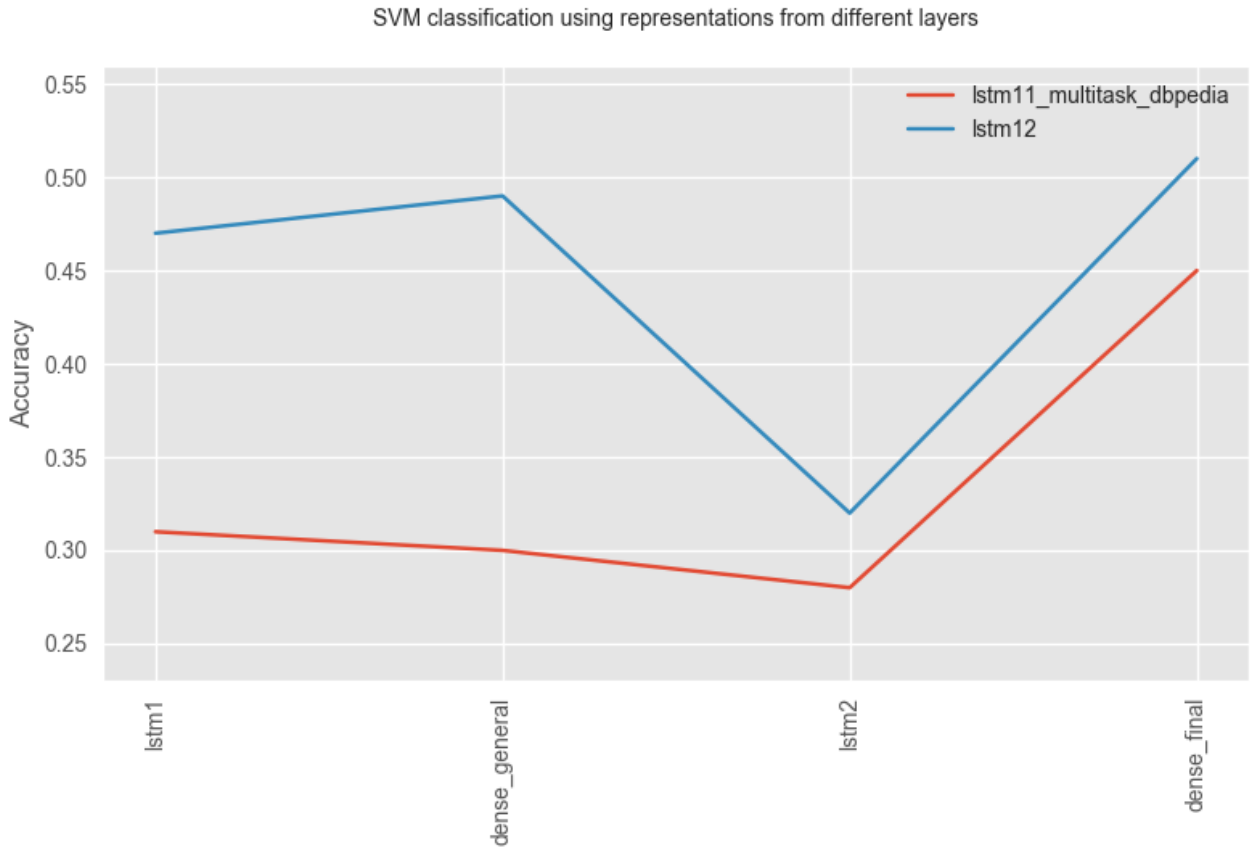


So, pretty obvious as expected:

- cnn21 performs better than cnn20 (because the latter is distracted with tuning this other subtask)
- and the model is better using the last layers than the first to predict the 71 dbpedia classes.
- The low performance in dense_general I think is because it's not getting backpropagation from conv2 during training (it's not learning to classify 71 categories in the CNN).

And I think conv2 does better then dense_final perhaps because more features are useful in SVMs for this specific task.

Figure 3. lstm11_multitask vs. lstm12 for dbpedia 71 way classification
lstm11 [0.31, 0.3, 0.28, 0.45]
lstm12 [0.47, 0.49, 0.32, 0.51]

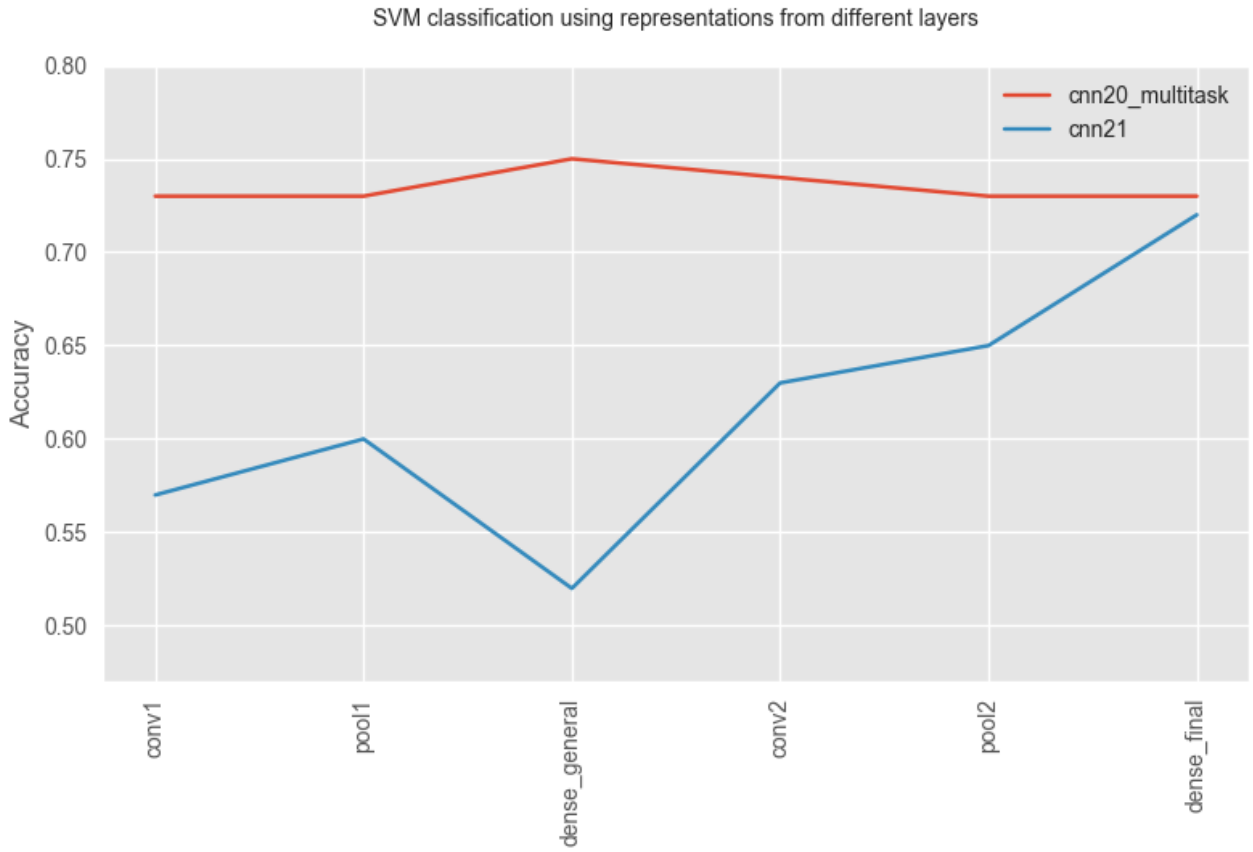


Here we get two same main results:

- Lstm12 performs better than lstm11 (because the latter is distracted with tuning this other subtask)
- and the models are getting better using the last layer than the first to predict the 71 dbpedia classes.
- But the drop is in lstm2 (the equivalent to conv2). I don't understand this behavior unless it was getting input from dense_general, which its not. I'll figure it out.

Here is the juicy part :)

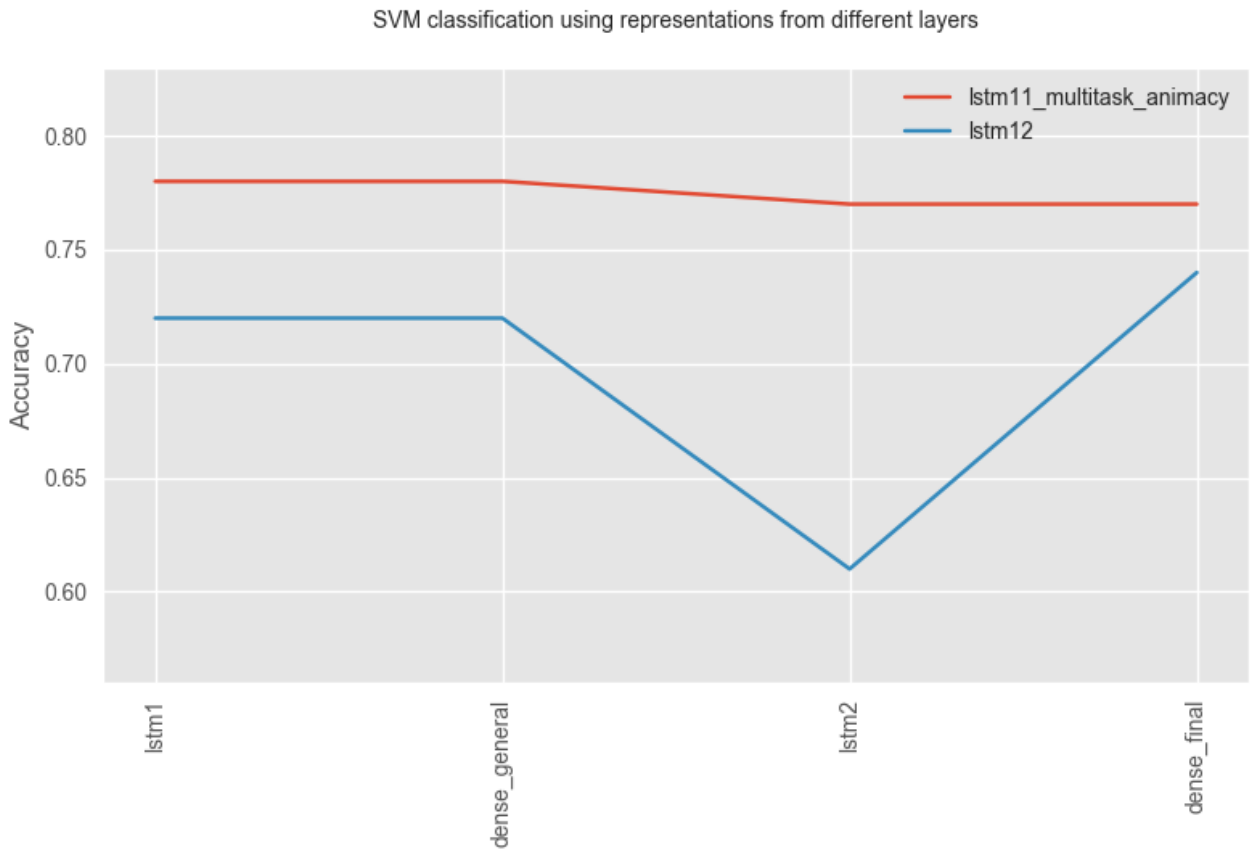
Figure 4. Cnn20_multitask vs. cnn21 for animacy 2-way classification



By tuning the dense_general layer to animate-inanimate distinction, this feature vector is good at distinguishing animate from inanimate. Here we prove it empirically. Not only that it makes the whole model a bit better until the end. The auxilarary animacy loss function is backpropagated not just to dense_general but to pool1 and conv1. But then these also influence every layer after them during training.

The point is: **our dense_general layer has *rich* animate-inanimate features (Fig. 4) –and actually the highest out of all the layers if you look closely— and it has *weak* dbpedia space classification (Fig. 2) while the last layer has a rich dbpedia space classification –and as rich as a normal CNN (see Fig. 2).**

Figure 5. lstm11_multitask vs. lstm12 for animacy 2-way classification



Same result just with the lstm2 effect that I don’t understand.

2.4. RSA

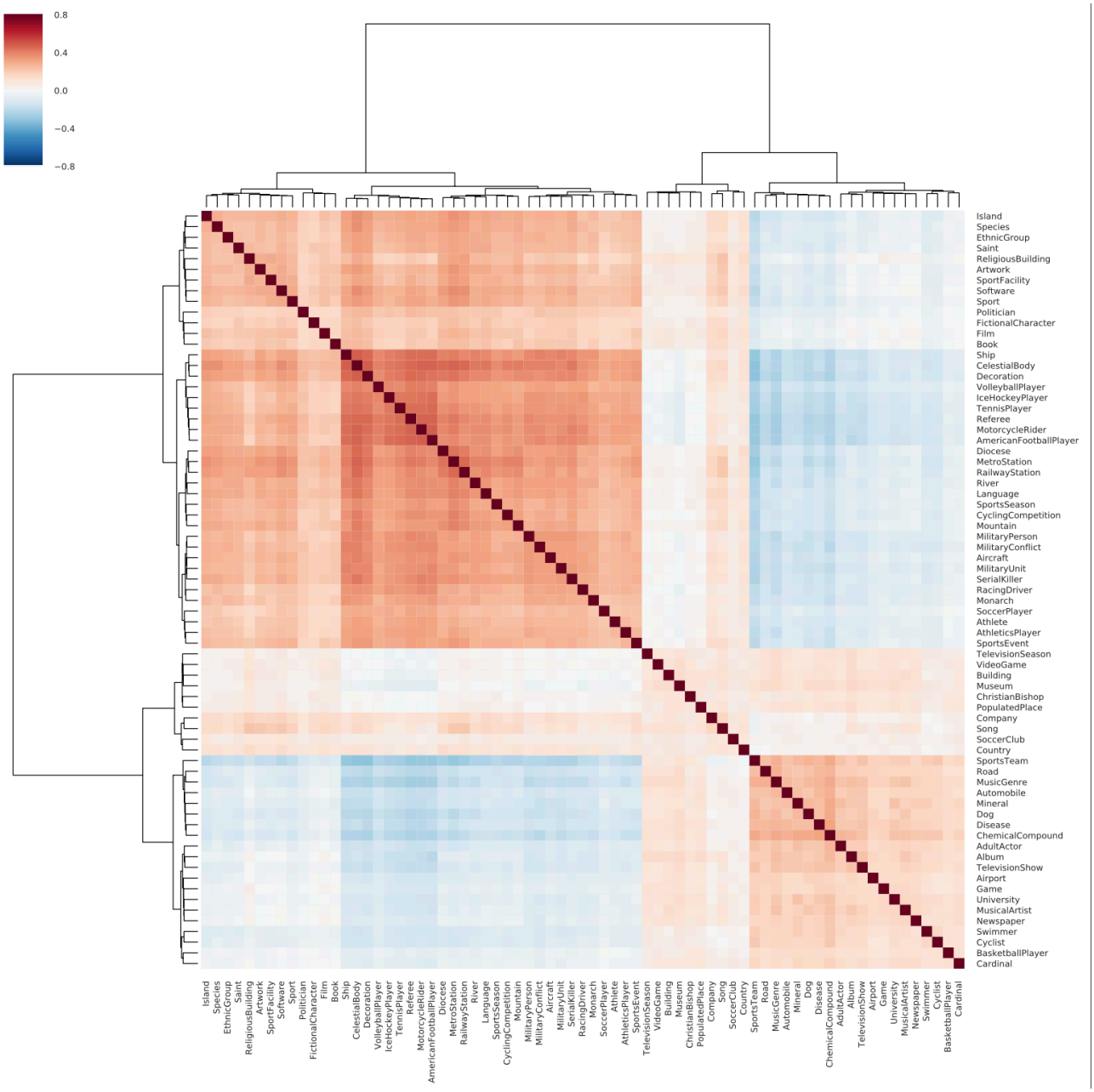
See all RSA in directory. See them in this order:

Conv1
Pool1
Dense_general
Conv2
Pool2
Dense_final

The LSTM ones are similar, but I think we'll use the CNN.

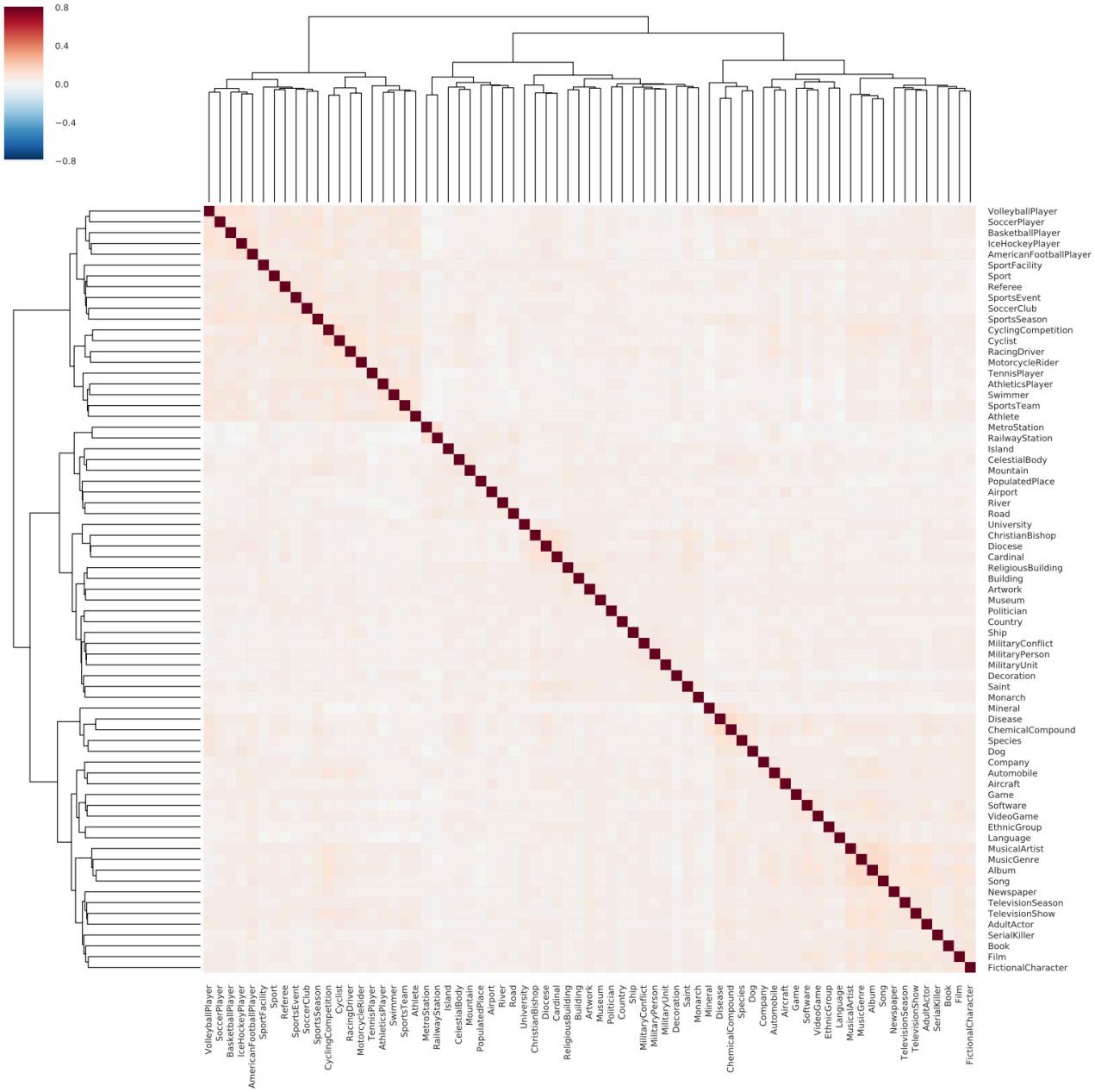
I was hoping maybe dense_general would split animate_vs_inanimate, but there are certain links that are very strong because of shared words (e.g., MilitaryPerson, MilitaryConflict):

Figure 6. cnn_20_multitask_RSA_ward_dense_general



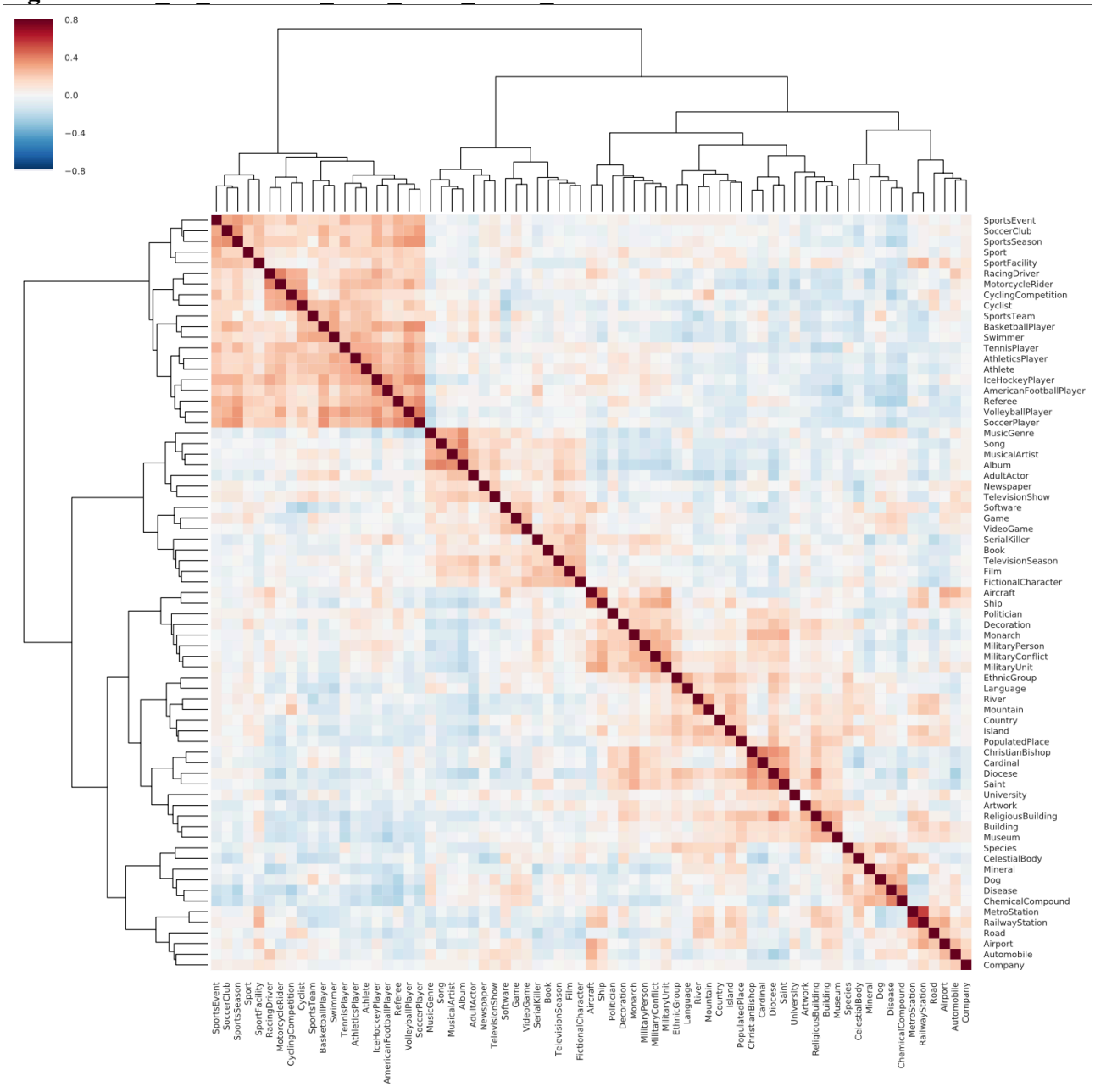
But these representations, whatever they are, they are tuned to animate-inanimate (as seen in the SVM classifications), and are very different than our normal model:

Figure 8. cnn_21_RSA_ward_dense_general



And the output layer clusters well:

Figure 8. cnn_20_multitask_RSA_ward_dense_final



Maybe check out conv1 from both cnn21_multitask and cnn20_multitask.

Conclusions

We successfully obtained models that have differences between the layers (2.2.). More importantly, we theoretically understand the representations of hidden layers more because we tuned them to Animate-Inanimate distinction. I'll also use theoretical RSMs group animate vs inanimate and compare to our RSMs. We empirecally understand the differences thanks to the SVMs (2.3, Fig. 4 sums it up).

LSTMs perfomed higher, but I can tune the hyperparameters of the CNN, and I should get 5-15% increase. I Prefer CNNs to LSTMS because 1. there is a more gradual incremental correlation between the last layer and the rest so we will obtain different search lights (see 2.2) and 2. When using SVMs, we see that the representations are closer to what we want: a model that uses a more higer-level distinction to then got down the hierarchy. 3. They have shown what we want in vision: layers that start out with representations that are shared across categories (lines and shapes in vision, animate and inamite feautres in our model) and end in more specific categories like the DBpedia ones.

TO-DO:

1. Write to Ev about stimuli (I may wait until Monday to not bother her).
2. Tune final hyperparemters for better accuracy.
3. Sentence length effects? Compare accuracy across different sentence lengths.
4. Show example of stimuli sentences and how 1. the RSMs would look like, 2. How each layer would correalte, 3. How they would predict SVMs.
5. Set up experiment in psychtoolbox.
6. Get a few subjects for pilot.
7. Run pilot of 2-4 subjects on 23 or 24th.
8. In parallel I'm going to run a combined CNN-LSTM model, and a deeper CNN (9 layers) to just to make sure we're not missing out on anything.

Questions

1. Of course, the main question is do we want our layer1 to be tuned to animate vs. inanimate? I'd say yes if we assume we're mapping a hierarchical structure. And it's a way to know what our layers represent (instead of being a black box).
2. Would it be nice to obtain a mapping from anterior to posterior? Would this maybe mess it up?
3. Maybe we should reduce our space a bit more because there are many more inanimate than animate and this can create problem typical of unbalanced datasets.
4. Would we want our layer 2 to be tuned to intermediate hierarchy (animate: human vs non-human, inanimate: natural vs artificial; or person-place, sports-nonsports, or every general category: sports-art-religious-nature-etc).
5. If our goal is to map a high-level representation and a low-level representation, why don't we just create a simple linear model of a 74-way classification with an SVM, and a simple linear model of a 2-way classification with those same sentences but using high-level tags like animate-inanimate? We could always compare the searchlight of both.

6. General comments on the clustering and RSA of CNN20_multitask (fig. 6 and 8)?
7. Should we eliminate general-redundant DBpedia classes like Athlete or AthleticsPlayer since it gets confused a lot with other similar classes or other more specific DBpedia classes

(e.g., Athlete gets confused 153 with AthleticsPlayer and 42 times with BasketballPlayer).