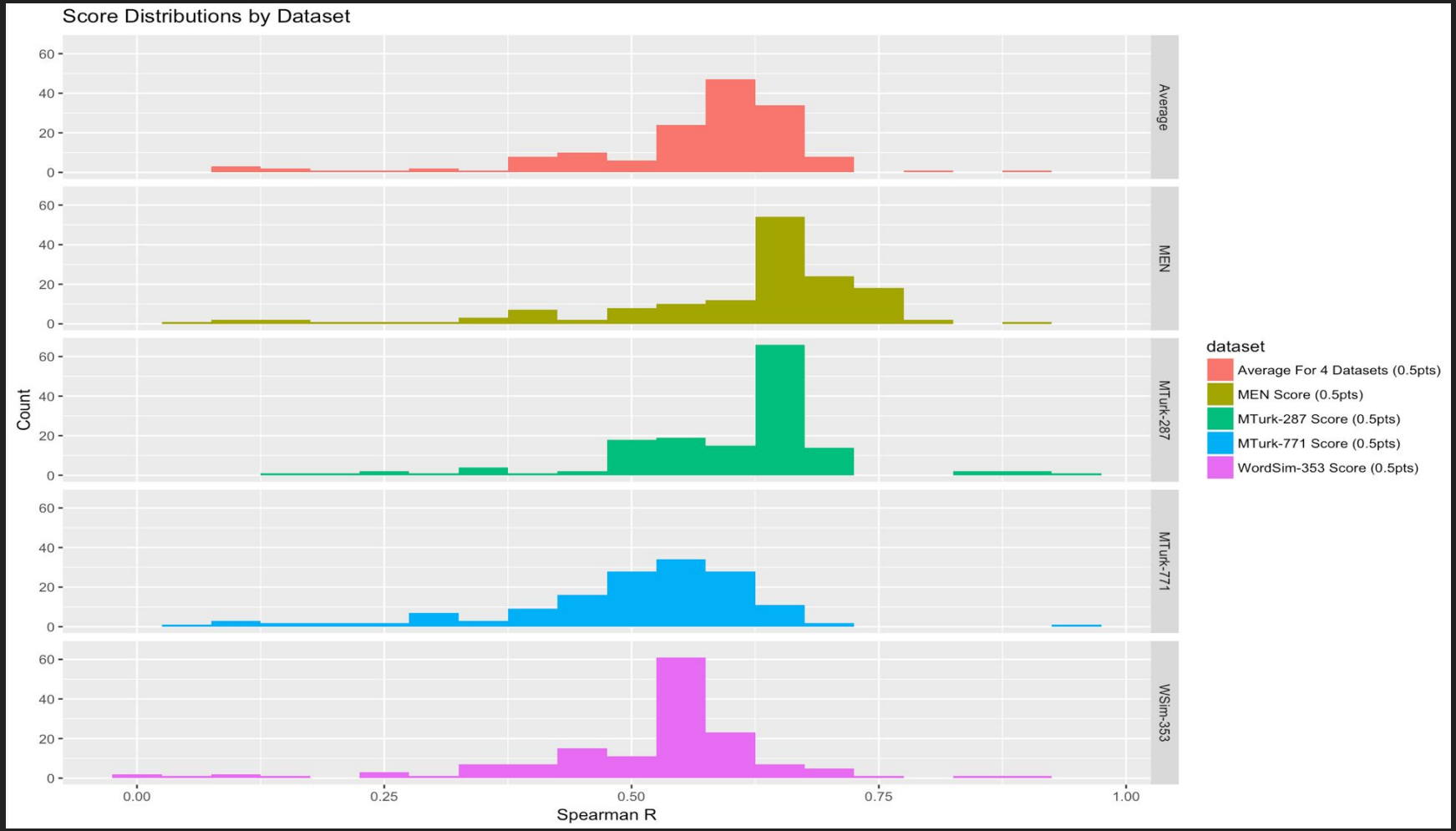
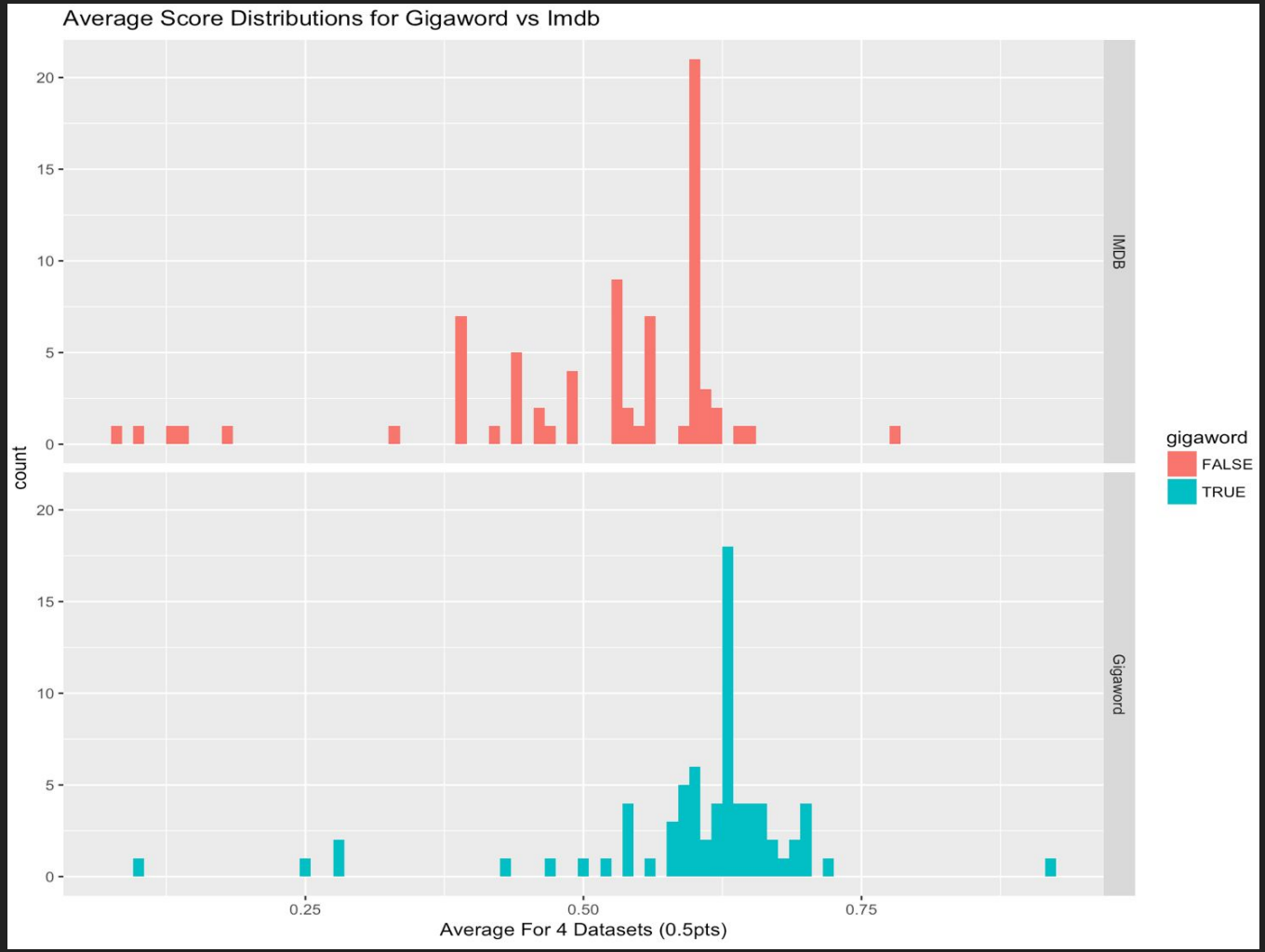
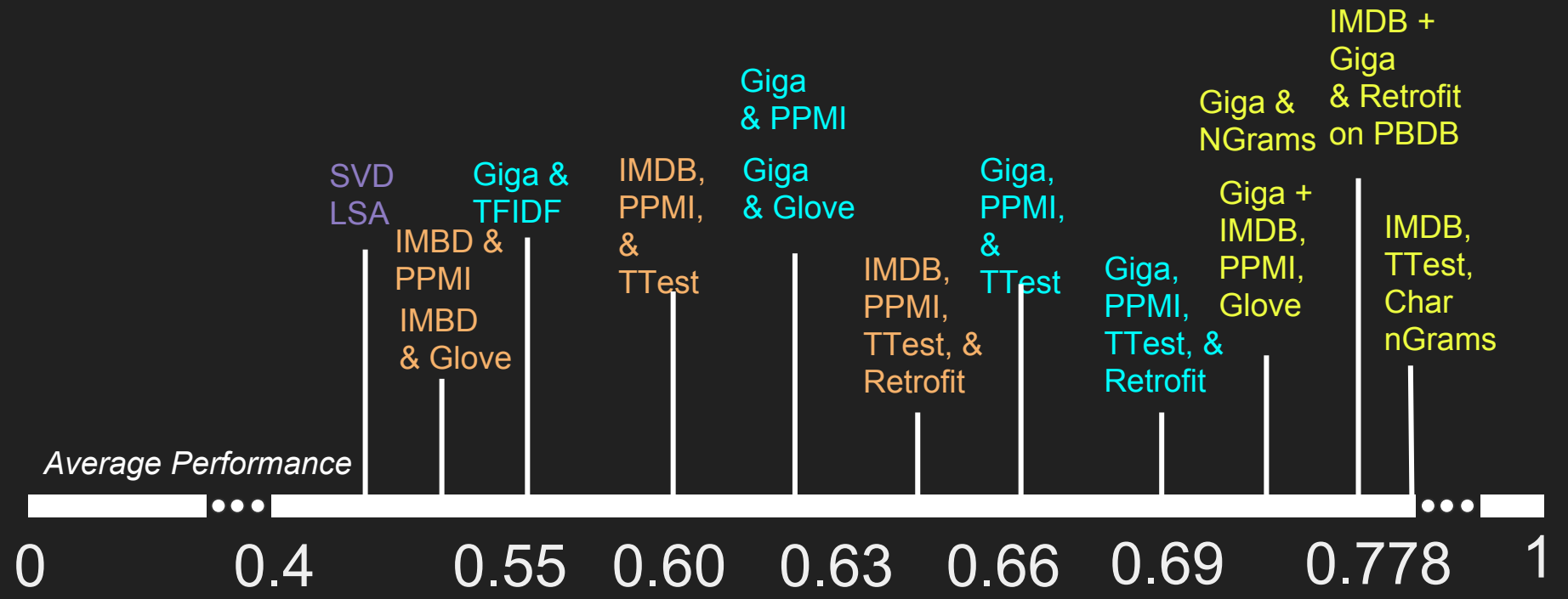
Bake-off 1 Discussion

cs224u
Zachary Maurer


Score Distributions by Dataset
Count
Spearman R
0.00
0.25
0.50
0.75
1.00
0
20
40
60
Average
MEN
MTurk-287
MTurk-771
WSim-353
dataset
Average For 4 Datasets (0.5pts)
MEN Score (0.5pts)
MTurk-287 Score (0.5pts)
MTurk-771 Score (0.5pts)
WordSim-353 Score (0.5pts)


Average Score Distributions for Gigaword vs Imdb
count
20
15
10
5
0
IMDB
Gigaword
0.25
0.50
0.75
Average For 4 Datasets (0.5pts)
gigaword
FALSE
TRUE

Performance of Common Combinations

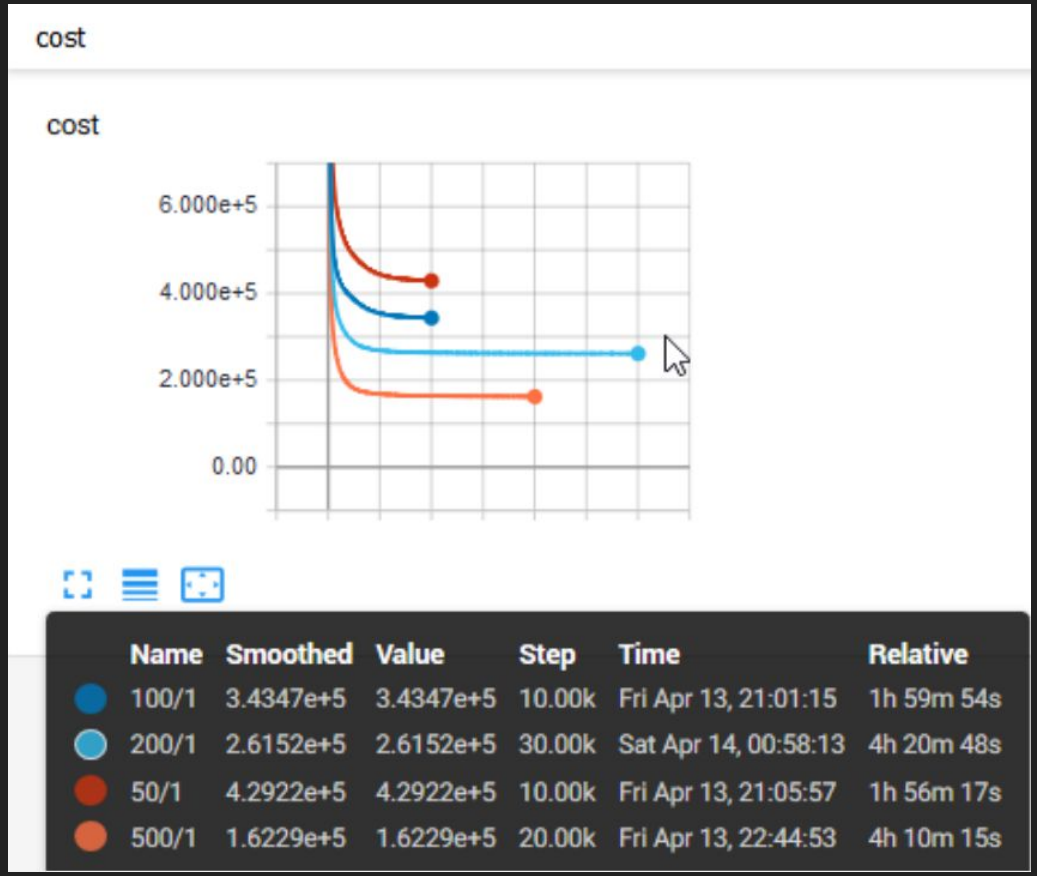


Some Observations

- TTest reweighting (and possibly PPMI) performed consistently well, > 0.6 avg score
- SVD/LSA was particularly poor across the board, Autoencoders were only slightly better
- Glove dimensions needed to be > 100 for good performance
- Gigaword tended to crush IMDB
- Ensembling the datasets by adding the matrices or averaging word vectors worked well, > 0.6 avg score consistently
- PPMI was much better than PMI

Training Glove

- It’s interesting that Glove was able to adjust to arbitrary re-weightings/scalings of the co-occurence matrix
- Guilherme Reis tried plotted the loss curves for different dimensionalities of vectors
- The loss seems to plateau around 10k iterations



Name	Smoothed	Value	Step	Time	Relative
100/1	3.4347e+5	3.4347e+5	10.00k	Fri Apr 13, 21:01:15	1h 59m 54s
200/1	2.6152e+5	2.6152e+5	30.00k	Sat Apr 14, 00:58:13	4h 20m 48s
50/1	4.2922e+5	4.2922e+5	10.00k	Fri Apr 13, 21:05:57	1h 56m 17s
500/1	1.6229e+5	1.6229e+5	20.00k	Fri Apr 13, 22:44:53	4h 10m 15s

Top 3 Systems

1. **Avg Score = 0.778**

“I had not had any experience with NLU before this class and decided to use this as an opportunity to explore some of the functions we wrote in HW1. For this bakeoff I used imdb20. First, I reweighted my VSM using ttest-reweighting. I then put this reweighted VSM through an n-gram character-level VSM. The major hyper parameter in this problem was n, so I explored possible values for n from 3-9 and found that 6 produced the best results.”

Top 3 Systems

2. Avg Score = 0.721

"1. We normalized and combined the four co-occurrence matrices by adding them all together (keep common words only)

2. We applied PPMI followed by the t-test (that already gives us average pearson's r = 0.666)

3. We then applied retrofitting using PBDB (PPDB: The Paraphrase Database), which eventually gives us average pearson's r = 0.780"

Top 3 Systems

3.1. Avg Score = 0.699

"I changed to the "gigaword_window20-flat.csv.gz" data set since it is much richer given that it is not restricted to movies but to much broader newspaper articles and information / words. This led to an increase. Furthermore, I also deduced 6-level ngrams from the giga20 dataset before I extracted the PMI. After some hyperparameter-search, 6 ended up being the best average value for the 4 datasets.

Top 3 Systems

3.2. Avg Score = 0.698

"Take the PPMI of the imdb5, imdb20, giga5, and giga20 datasets. Reindex the matrices so they each all 6021 words, filling in with 0s. Sum the matrices element wise. Run GloVe with n=250 and max_iter=30. [Last submission, use this one, thanks] r = 0.780"

Top 3 Systems

3.3. Avg Score = 0.691

"1. Merge together all four starting datasets with equal weighting on each. 2. Apply PPMI 3. Reduce dimensions to 400 with GloVE. To arrive at this, we did a hyper-parameter grid search over three different layers: 1. weighting of corpora (merging {1-4} corpora with different weights) 2. reweighting (PMI, PPMI, TFIDF) 3. dimension reduction (LSA{50,100,200,400}; GloVe{100, 200, 400}) We did a sort of beam search, keeping the 5 best candidates from each layer."

Other Interesting Approaches

- “Then we used downloaded human-scored dimensions of identities, actions, and situations on how good/bad (evaluation, E), powerful/weak (power, P), and loud/quiet (active, A) each concept is (EPA ratings)...we created a weighted knowledge graph where the weight of the edge was negative, mean-centered euclidean distance between the two concepts in EPA-space. We retrofitted our count matrix on this weighted knowledge graph (i.e. we set the beta of the retrofit function to the weight of each tie in the knowledge graph) with 10 iterations. Our spearman-r decreased after retrofitting on the EPA information.”

- “This alone gave some values moderately higher than the baseline, but wasn't particularly interesting. From there, I tried some matrix multiplication. In doing so, I found that, as potentially expected, multiplying the glove_test output vector by itself would nuke the Spearman values, but weirdly enough, multiplying that again would give a reasonable value. This pattern continued over all n multiplied that I tested (out to 20) - for any odd n of glove_out multiples, I'd get reasonable outputs (though never quite as good as n=1), while for even multiples, I'd get awful values. This is n=3, that is, out*out*out, which I selected both as a representative sample of the larger pattern as well as of one of the better examples. While not as good as the original, I thought it was pretty cool nonetheless.”

- I changed the distance function from using a cosine distance function to using a euclidean distance function. As we can see from the scores this actually led to a less accurate result, as euclidean distance is often not as good a measure of similarity due to the frequency of words. Cosine uses the angles between words which is probably why it was much more accurate.

Thank you!