Bake-off 1

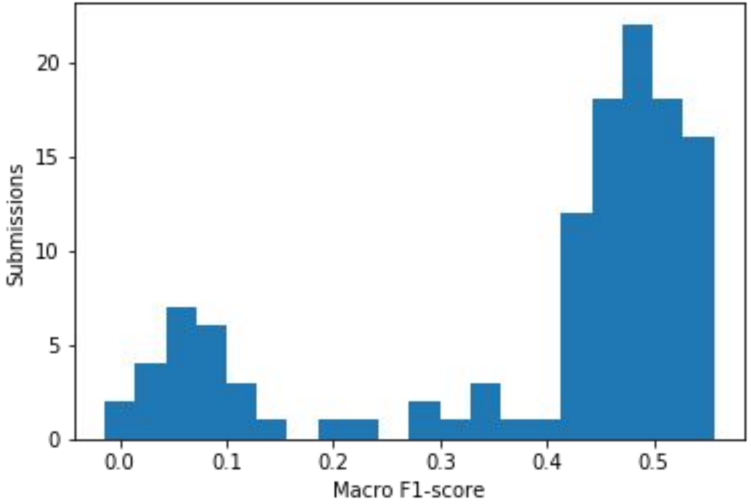
Lucy Li and Ashkon Farhangi
CS 224u

Task

Evaluate distributed representations using word similarity datasets.

- Similarity datasets have word pairs with an associated human-annotated similarity score
- Evaluation measures the distance between the word pairs in your chosen VSM

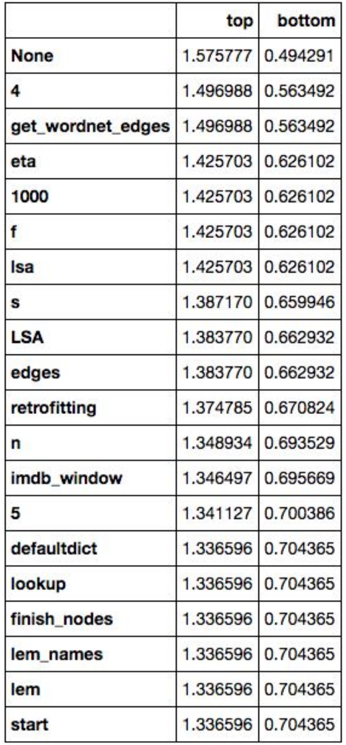
Histogram of scores



Thanks Chris for the idea!

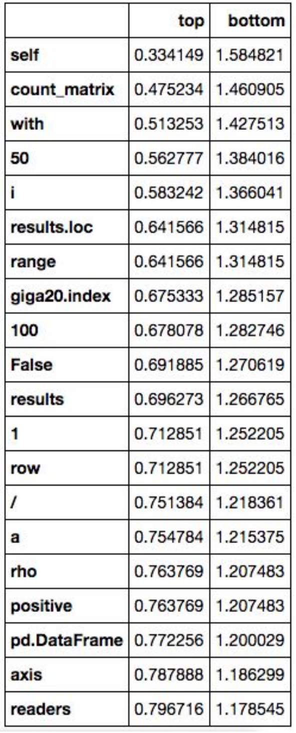
What distinguishes the high and low scorers?

High o/e for top scorers



	top	bottom
None	1.575777	0.494291
4	1.496988	0.563492
get_wordnet_edges	1.496988	0.563492
eta	1.425703	0.626102
1000	1.425703	0.626102
f	1.425703	0.626102
lsa	1.425703	0.626102
s	1.387170	0.659946
LSA	1.383770	0.662932
edges	1.383770	0.662932
retrofitting	1.374785	0.670824
n	1.348934	0.693529
imdb_window	1.346497	0.695669
5	1.341127	0.700386
defaultdict	1.336596	0.704365
lookup	1.336596	0.704365
finish_nodes	1.336596	0.704365
lem_names	1.336596	0.704365
lem	1.336596	0.704365
start	1.336596	0.704365

High o/e for low scorers

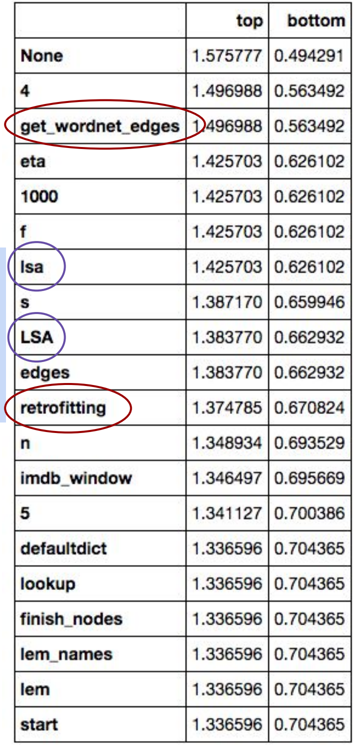


	top	bottom
self	0.334149	1.584821
count_matrix	0.475234	1.460905
with	0.513253	1.427513
50	0.562777	1.384016
i	0.583242	1.366041
results.loc	0.641566	1.314815
range	0.641566	1.314815
giga20.index	0.675333	1.285157
100	0.678078	1.282746
False	0.691885	1.270619
results	0.696273	1.266765
1	0.712851	1.252205
row	0.712851	1.252205
/	0.751384	1.218361
a	0.754784	1.215375
rho	0.763769	1.207483
positive	0.763769	1.207483
pd.DataFrame	0.772256	1.200029
axis	0.787888	1.186299
readers	0.796716	1.178545

Thanks Chris!

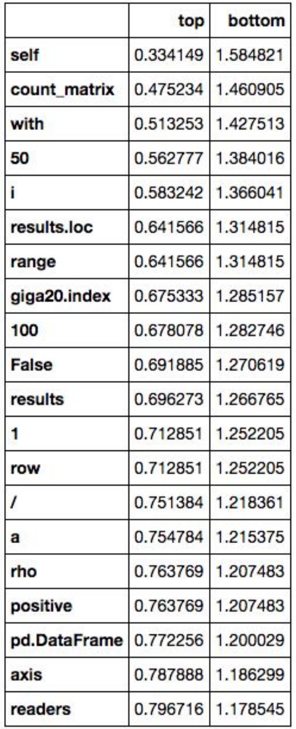
What distinguishes the high and low scorers?

High o/e for top scorers



	top	bottom
None	1.575777	0.494291
4	1.496988	0.563492
get_wordnet_edges	1.496988	0.563492
eta	1.425703	0.626102
1000	1.425703	0.626102
f	1.425703	0.626102
lsa	1.425703	0.626102
s	1.387170	0.659946
LSA	1.383770	0.662932
edges	1.383770	0.662932
retrofitting	1.374785	0.670824
n	1.348934	0.693529
imdb_window	1.346497	0.695669
5	1.341127	0.700386
defaultdict	1.336596	0.704365
lookup	1.336596	0.704365
finish_nodes	1.336596	0.704365
lem_names	1.336596	0.704365
lem	1.336596	0.704365
start	1.336596	0.704365

High o/e for low scorers



	top	bottom
self	0.334149	1.584821
count_matrix	0.475234	1.460905
with	0.513253	1.427513
50	0.562777	1.384016
i	0.583242	1.366041
results.loc	0.641566	1.314815
range	0.641566	1.314815
giga20.index	0.675333	1.285157
100	0.678078	1.282746
False	0.691885	1.270619
results	0.696273	1.266765
1	0.712851	1.252205
row	0.712851	1.252205
/	0.751384	1.218361
a	0.754784	1.215375
rho	0.763769	1.207483
positive	0.763769	1.207483
pd.DataFrame	0.772256	1.200029
axis	0.787888	1.186299
readers	0.796716	1.178545

It seems like retrofitting on WordNet and LSA are common ways to build better models.

Thanks Chris!

First place

1. PMI
2. T-test
3. Retrofitting on WordNet
4. Autoencoder

Group 80’s score: **0.556024**
Emma, Santosh, Mel

```
def myVSM11(X:pd.DataFrame):
    h1 = _ppmi(X)
    h2 = _ttest(h1)
    h3 = _retro(h2)
    h4 = _autoEncoder(h3, 1000)
    return h4

imdb5_myVSM = myVSM11(imdb5)
```

```
normalized,
max_iter=100,
hidden_dim=1000,
eta=0.001
```

Second place

PMI
+
Retrofitting w/
subword model

Jingying’s score: **0.552842**

```
def custom_model(df, readers=READERS):
    pmi_model = vsm.pmi(df)
    ngrams_model = vsm.ngram_vsm(pmi_model, n=4)
    ngrams_words = get_ngrams_matrix(pmi_model, ngrams_model, n=4)
    ngrams_norm = ngrams_words.apply(vsm.length_norm, axis = 1)
    wn_edges = get_wordnet_edges()
    wn_retro = Retrofitter()
    subword_model = ttest(ngrams_norm * 0.4 + pmi_model)
    wn_index_edges = convert_edges_to_indices(wn_edges, subword_model)
    X_retro = wn_retro.fit(subword_model, wn_index_edges)
    return X_retro
custom_model(imdb5)
```

Third place

Chaining:

1. PMI
2. T-test
3. LSA
4. Retrofitting

Aditya’s score: **0.552457**

```
def original_model(dataset_loc, k=5150):
    dataset = pd.read_csv(os.path.join(VSM_HOME, dataset_loc), index_col=0)
    dataset_pmi = vsm.pmi(dataset)
    dataset_ttest = ttest_reweight(dataset_pmi)
    dataset_pmi_ttest_lsa = vsm.lsa(dataset_ttest, k)
    results = full_word_similarity_evaluation(dataset_pmi_ttest_lsa)
    wn_edges = get_wordnet_edges()
    wn_index_edges = convert_edges_to_indices(wn_edges, dataset_pmi_ttest_lsa)
    wn_retro = Retrofitter(verbose=True)
    dataset_retro = wn_retro.fit(dataset_pmi_ttest_lsa, wn_index_edges)
    return dataset_retro
original_model('imdb_window5-scaled.csv.gz')
```

Last place

1. T-test
2. LSA
3. Jaccard distfunc

Score: **0.009926**

```
def my_model(data):
    data_result = reweighting(data)
    data_result = vsm.lsa(data_result, k = 200)
    return data_result
giga20_mymodel = my_model(giga20)
full_word_similarity_evaluation(giga20_mymodel, readers=BAKEOFF,
distfunc=vsm.jaccard)
```