In-class bake-offs

Bill MacCartney and Christopher Potts
CS224U, Stanford University
Background

• We've moved a lot of the material into videos and codelabs.
• A number of the classes will be devoted to hands-on work with the models and concepts.
• These meetings will be centered around challenge problems and low-stakes competitions.
• You get credit for these problems by working on them in class (and only there).
April 1 challenge problem

Working with `distributedwordreps.py`:

Do whatever you like to the count matrix $ww$ or $wd$ and run `word_similarity_evaluation` on it.

- 1 point: Any modification to $ww$ or $wd$ plus a successful run.
- 2 points: Highest scoring modification of all the teams.

The only requirements: no data beyond $ww$ or $wd$, and the work has to be done in the classroom.
April 6 challenge problem

**Working with** `distributedwordreps.py`:

Do whatever you like to the count matrix \(ww\) or \(wd\) to create a new VSM \(v\) and run

\[
\text{analogy\_evaluation(mat=v[0], rownames=v[1], src\_filename='distributedwordreps-data/question-data/gram7-past-tense.txt')}
\]

- 1 point: Any modification to \(ww\) or \(wd\) plus a successful run.
- 2 points: Highest mean reciprocal rank of all the teams.

As before, the only requirements: no data beyond \(ww\) or \(wd\), and the work has to be done in the classroom.
April 8 challenge problem

Problem: for two words $w_1$ and $w_2$, predict $w_1 \subset w_2$ or $w_2 \supset w_1$

$hippo \subset mammal \quad mammal \supset hippo$

Data:

- `vocabulary, items = pickle.load(file('wordentail_data.pickle'))`
- `items['train'] = {1.0: [[w_1, w_2], [w_6, w_7], ...], -1.0: [[w_4, w_3], [w_2, w_1], ...]}`
- `items['test'] = {1.0: [[w_1, w_7], [w_2, w_3], ...], -1.0: [[w_3, w_2], [w_9, w_8], ...]}`
- `items['disjoint_vocab_test'] = {1.0: [[w_21, w_72], ...], -1.0: [[w_97, w_121], ...]}`

The all three sets are disjoint. The test vocabulary is subset of the train vocabulary. The disjoint_vocab_test is disjoint from the others. All the words are in `glv`.

You should train only on `items['train']`

Starter code: `wordentail.py`

- `data_prep`: loads the data; you write `vector_func` and `vector_combo_func`
- `train_and_evaluate`: accepts the output of `data_prep` and handles evaluation; you set up and tune the network
April 8 challenge problem (continued)

You decide whether these two vectors come from by writing `vector_func`. They can be random or they can come from a VSM you like.

You decide how to put these vectors together by writing `vector_combo_func`. (Simplest is concatenation.)

You tune the network: hidden_dim, training iterations, learning date (activation function, optimizer, …). Tips: you'll know after 50 iterations whether things are good, and small hidden_dim seems powerful.