Word2Vec

Matt Olson

December 8, 2017

README.md

- `word2vec_tutorial.py`: playing around with Google's trained Word2Vec model
- `word2vec_preprocess.py`: preprocess news articles
- `word2vec.py`: Tensorflow implementation
- `doc2vec_preprocess.py`: preprocess news articles for classification
- `doc2vec.py`: train classifier to distinguish news sources

Word2Vec: High Level

- Embed words into vectors in $\mathbb{R}^d$
- Algebraic and geometric operations on these embeddings make sense!
  - king - man + woman $\approx$ queen
  - "nearest neighbor" to "Chicago" is "Windy City"
  - word2vec between languages amounts to orthogonal rotation
- Amounts to training a neural network with one hidden layer: learn to predict target words from context words
- Doc2Vec: embed documents into vectors (can then plug into any classifier)

Fun with Google News Word2Vec I

Google trained Word2Vec on hundreds of billions of words from Google News: can access trained model!

Can answer questions of the form: "a is to b as c is to d" by $d \approx a + b - c$

```
model.most_similar(positive=['France', 'Madrid'], negative=['Spain'], topn=2)
# [(u'Paris', 0.7502285242080688), (u'Marseille', 0.6038430333137512)]

model.most_similar(positive=['China', 'Madrid'], negative=['Spain'], topn=2)
# [(u'Beijing', 0.7582258582115173), (u'Shanghai', 0.6651272177696228)]

model.most_similar(positive=['Russia', 'Madrid'], negative=['Spain'], topn=2)
# [(u'Moscow', 0.8241983652114868), (u'Kremlin', 0.6693196296691895)]

model.most_similar(positive=['Germany', 'Madrid'], negative=['Spain'], topn=2)
# [(u'Munich', 0.7192912697792053), (u'Berlin', 0.7178487777709961)]

model.most_similar(positive=['Japan', 'Madrid'], negative=['Spain'], topn=2)
# [(u'Tokyo', 0.7938408851623535), (u'Nagoya', 0.6847245097160339)]
```

Fun with Google News Word2Vec II

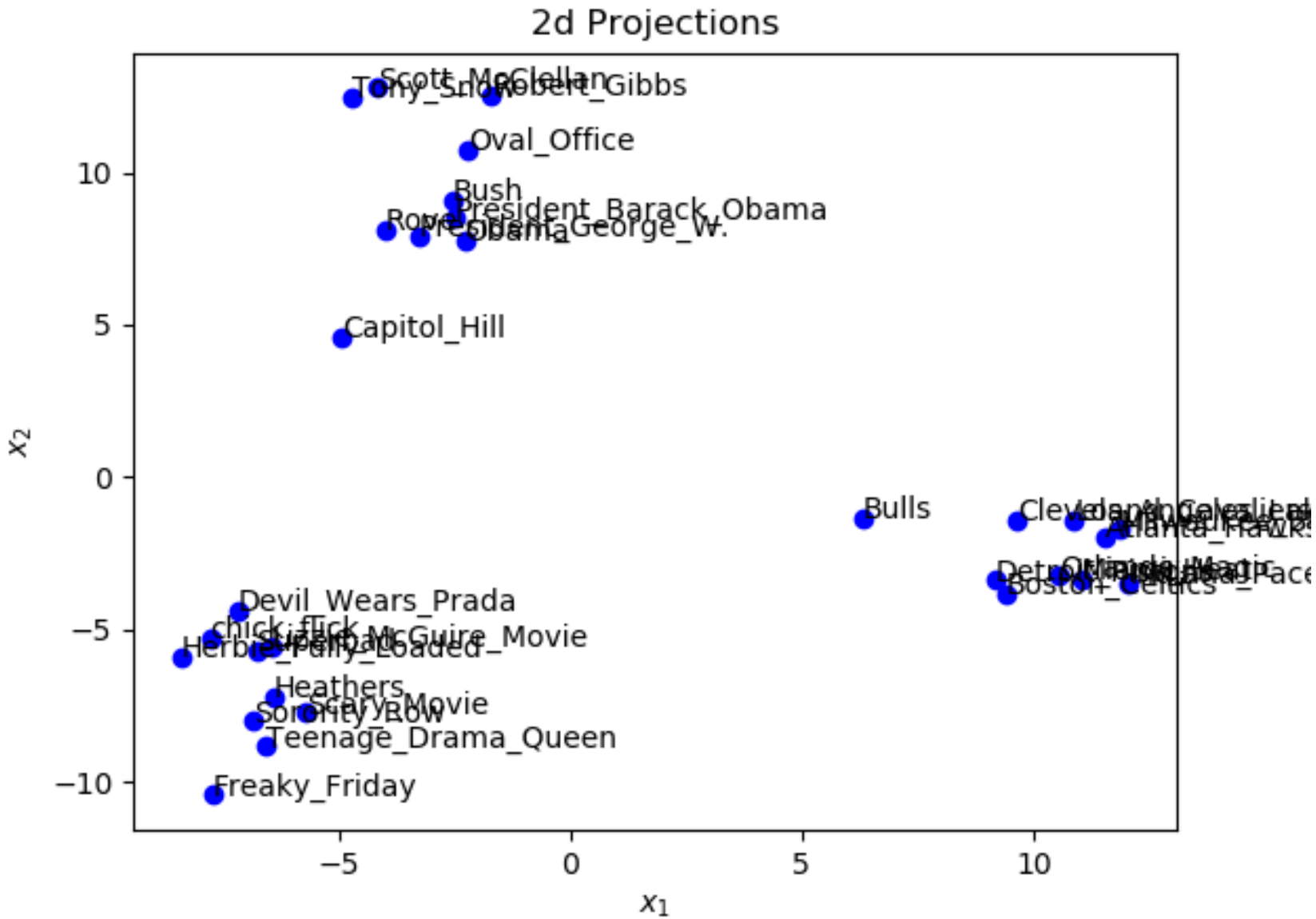
```
model.most_similar(positive=['Colts', 'Tom_Brady'], negative=['Patriots'], topn=2)
# [(u'Peyton_Manning', 0.767792284488678), (u'Drew_Brees', 0.6715065240859985)]

model.most_similar(positive=['Packers', 'Tom_Brady'], negative=['Patriots'], topn=2)
# [(u'Brett_Favre', 0.731001615524292), (u'Aaron_Rodgers', 0.6976802349090576)]

model.most_similar(positive=['Steelers', 'Tom_Brady'], negative=['Patriots'], topn=2)
# [(u'Ben_Roethlisberger', 0.7389534711837769),

model.most_similar(positive=['Saints', 'Tom_Brady'], negative=['Patriots'], topn=2)
# [(u'Drew_Brees', 0.6620509624481201),
```

Fun with Google News Word2Vec III



Skip-Gram Model and CBOW

The quick brown fox jumped over the lazy dog.

Context: neighboring words (red)

Target: words to predict (blue)

(context, target): ([quick], the), ([the, brown], quick), ...

Continuous Bag of Words: predict the target from context

Skip-Gram: predict the context from target work

Model Nuts and Bolts

sentence: $w_1, w_2, \ldots, w_3$

"one hot encode" each w_i to get vectors

use w_{t-1} and w_{t+1} to predict w_t using neural network with one hidden layer

$$p(w_t | w_{t-1}, w_{t+1}) \approx \sigma\left(W_2\sigma\left(W_1\begin{bmatrix} w_{t-1} \\ w_{t+1} \end{bmatrix} + b_1\right) + b_2\right)$$

The embeddings are precisely the matrix W_1

Fun fact: the Google model knows 3 million words, so our multinomial classifier has 3 million categories to predict!

Fake News! (Classification with Doc2Vec)

- Apply Doc2Vec to 36k articles from Fox, NYT, Atlantic, Reuters, National Review
- Embed articles in 150 dimensional space (unsupervised)
- Train logistic regression to differentiate article sources based on embedding (85% accuracy overall)

	Fox	NYT
Fox	923	77
NYT	40	960

Figure: Confusion Matrix