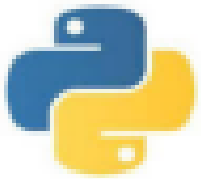
Text Mining with Deep Learning: Workshop

Raja

I^{2}R, A*STAR

Tools Needed for the Workshop



- Python 3.*
  - » install Anaconda with Spyder will do
- NLTK
  - » Required Packages: **reuters.corpus, punkt, stopwords**
- Gensim Module
  - » For Word2Vec
- Linux/Windows/MacOS

Workshop Structure

I. Learning Text Semantics with DL

a) Semantic Word Similarity with Word2Vec

b) Semantic Word Compositions with Word2Vec

II. Text Classification with DL

a) Data - Reuters corpus

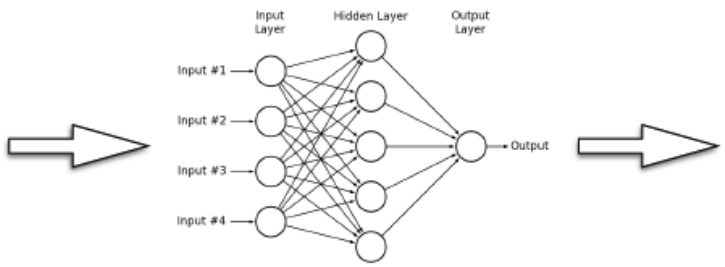
b) Classification with Word2Vec

c) Baseline Classification with SVM & Naïve Bayes

Word2Vec - Recap

word

The
Cardinals
will
win
the
world
series



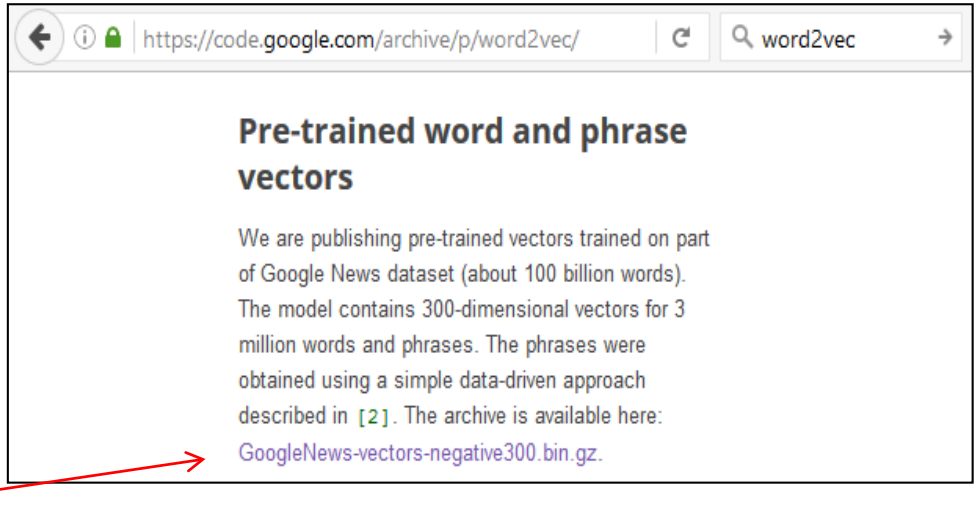
vector

(0.12, 0.23, 0.56)
(0.24, 0.65, 0.72)
(0.38, 0.42, 0.12)
(0.57, 0.01, 0.02)
(0.53, 0.68, 0.91)
(0.11, 0.27, 0.45)
(0.01, 0.05, 0.62)

- A specific DL approach to learn vector representations of words
- A w2v model should be learnt first by passing in a large text collection

Hands-on I: Text Semantics with Word2Vec

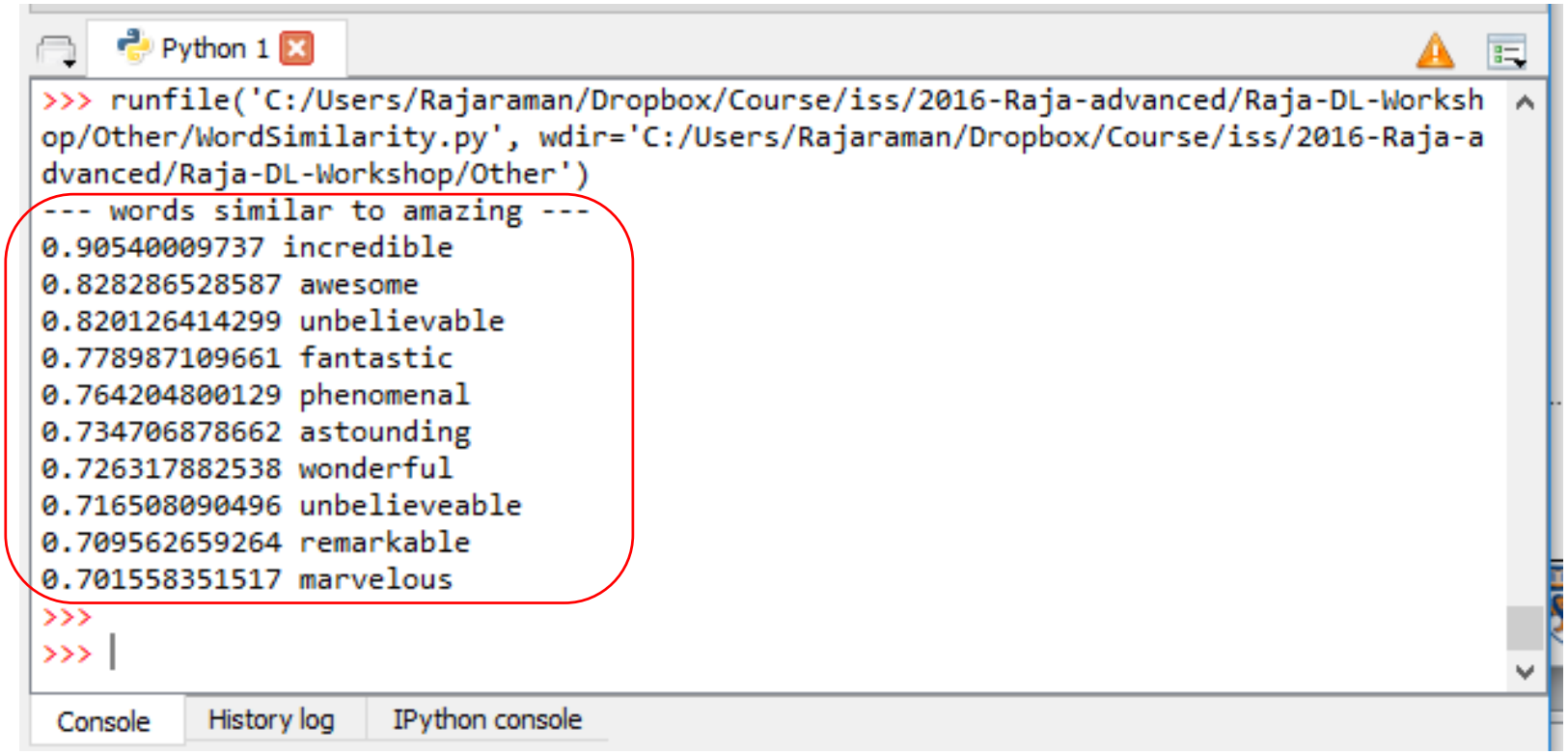
- Get Word2Vec Model
  - » Use pre-trained model from Google News
  - » https://code.google.com/archive/p/word2vec/
  - » Download, Unzip and Copy to your workshop folder



Text Semantics with Word2Vec

- Task A: Word Similarity
  - » Run "*python WordSimilarity.py*"
  - » Observe top ranked words returned as similar to 'amazing'

Results

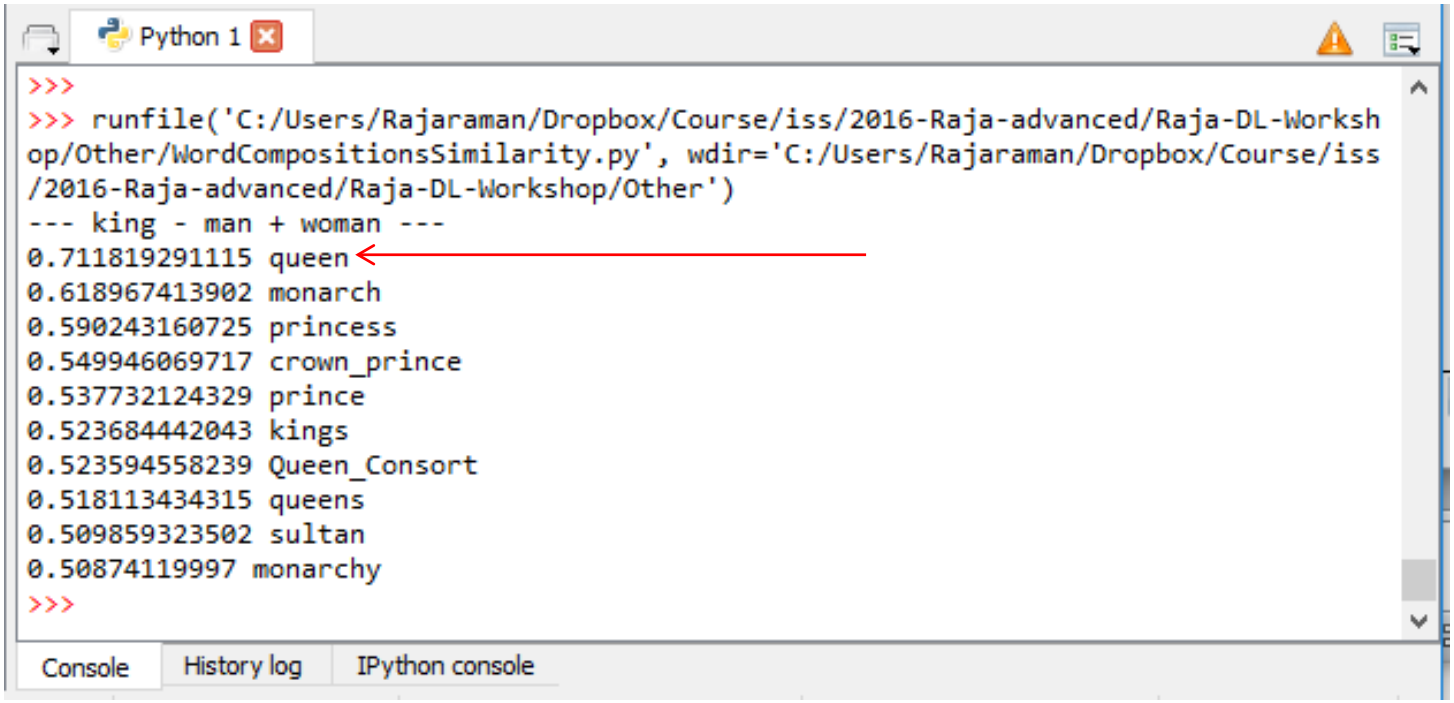


Exercise: Edit code and try setting **words=['king']** (Line 5), and run it

Text Semantics with Word2Vec

- Task B: Word Compositions
  - » Edit "*WordSimilarity.py*"
  - » Comment out Line 5, and uncomment Line 6
  - » Run it
  - » Observe top ranked word returned as similar to '*king – man + woman*'

Results



Exercise: Edit code and try this composition *prince – man + woman*

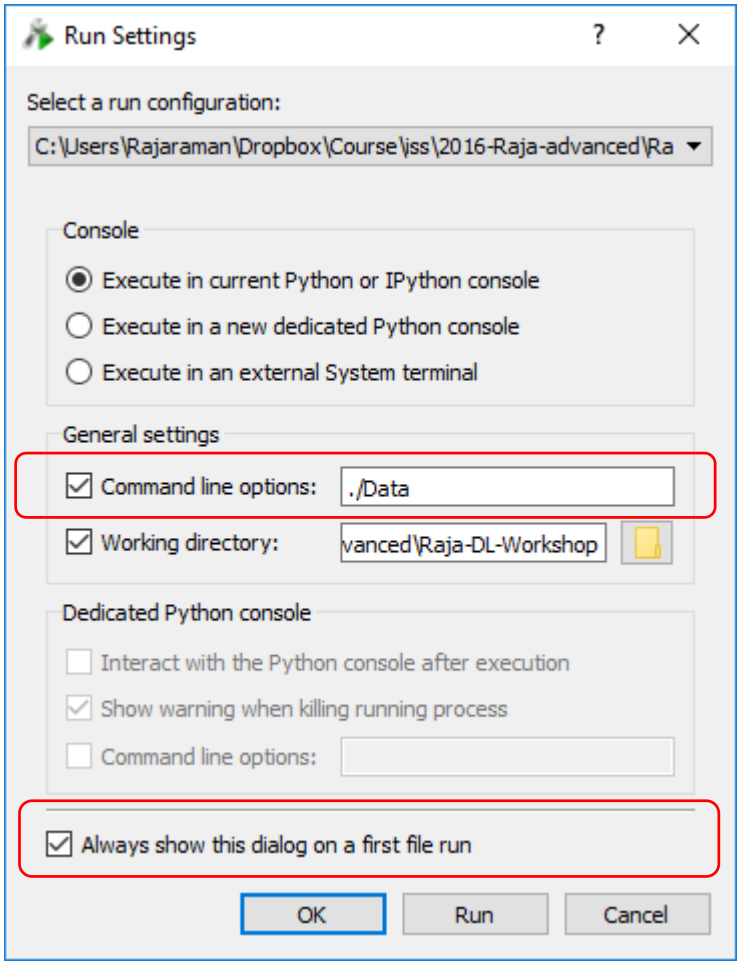
Hands-on II: Text Classification with Word2Vec

- Problem
  - » Given a bunch of documents divided into training and test sets
  - » Build a classifier using the training set
  - » Predict class label on test set
- Data
  - » Reuters corpus – popular with research community
  - » Comes with NLTK
  - » 10,788 documents (1.3 million words)

    classified into 90 topics
  - » We will consider one topic: **money-fx**

Hands-on II: Text Classification with Word2Vec

- Task A: Build Word2Vec Model
  - » Run "*python BuildW2VModel.py ./Data*"
  - » Will create **word2vec_model** file (about 17MB size)



Spyder: Run -> Configure

Text Classification with Word2Vec

- Task B: Run Word2Vec Classifier
  - » Run "*python RunW2V.py ./Data word2vec_model money-fx*"
  - » And wait…
  - » Prints % accuracy to the console (5 runs)

Text Classification with Word2Vec

- Task C: Run Baseline Classifier (SVM & NBayes)
  - » Run "*python RunBaseline.py money-fx*"
  - » Prints % accuracy to the console (5 runs)
- Study average accuracy of all 3 classifiers and compare
  - » Does SVM improve over NBayes? By how much (on average)?
  - » Does Word2Vec improve over them? By how much (on average)?

Note: Don't compare individual runs. Both approaches use different subsets of data.

Summary

- Word2vec can nicely learn and represent text semantics
  - » Can naturally capture word compositions
  - » NO labelled data is required!
- Word2vec significantly improves classification performance compared to all previous models
  - » Can get even better with more data thrown at it!