Latent Semantic Analysis

- Latent Semantic Analysis (LSA) is a framework for analyzing text using matrices
- Find relationships between documents and terms within documents
- Used for document classification, clustering, text search, and more
- Lots of experts here at CU Boulder!

sci-kit learn

- sci-kit learn is a Python library for doing machine learning, feature selection, etc.
- Integrates with numpy and scipy
- Great documentation and tutorials

Vectorizing text

- Most machine-learning and statistical algorithms only work with structured, tabular data
- A simple way to add structure to text is to use a document-term matrix

Document-term matrix

```python
In [97]:
import sklearn
# Import all of the scikit learn stuff
from __future__ import print_function
from sklearn.decomposition import TruncatedSVD
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import Normalizer
from sklearn import metrics
from sklearn.cluster import KMeans, MiniBatchKMeans

import pandas as pd
import warnings
# Suppress warnings from pandas library
warnings.filterwarnings("ignore", category=DeprecationWarning,
    module="pandas", lineno=570)
```
import numpy

In [155]: example = ["Machine learning is super fun",
               "Python is super, super cool",
               "Statistics is cool, too",
               "Data science is fun",
               "Python is great for machine learning",
               "I like football",
               "Football is great to watch"]
vectorizer = CountVectorizer(min_df = 1, stop_words = 'english')
dtm = vectorizer.fit_transform(example)
pd.DataFrame(dtm.toarray(),index=example,columns=vectorizer.get_feature_names ()).head(10)

Out[155]:

<table>
<thead>
<tr>
<th>cool</th>
<th>data</th>
<th>football</th>
<th>fun</th>
<th>great</th>
<th>learning</th>
<th>like</th>
<th>machine</th>
<th>python</th>
<th>science</th>
<th>statistics</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Each row represents a document. Each column represents a word. So each document is a 13-dim vector.
- Each entry equals the number of times the word appears in the document
In [115]:

    # Get words that correspond to each column
    vectorizer.get_feature_names()

Out[115]:

    ['cool',
     'data',
     'football',
     'fun',
     'great',
     'learning',
     'like',
     'machine',
     'python',
     'science',
     'statistics',
     'super',
     'watch']

Example: "machine" appears once in the first document, "super" appears twice in the second document, and "statistics" appears zero times in the third document.

**Singular value decomposition and LSA**

In [116]:

    # Fit LSA. Use algorithm = "randomized" for large datasets
    lsa = TruncatedSVD(2, algorithm = 'arpack')
    dtm_lsa = lsa.fit_transform(dtm)
    dtm_lsa = Normalizer(copy=False).fit_transform(dtm_lsa)

• Each LSA component is a linear combination of words

In [166]:

    pd.DataFrame(lsa.components_, index = ['component_1','component_2'], columns =
                 vectorizer.get_feature_names())

Out[166]:

<table>
<thead>
<tr>
<th></th>
<th>cool</th>
<th>data</th>
<th>football</th>
<th>fun</th>
<th>great</th>
<th>learning</th>
<th>like</th>
<th>mach</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_1</td>
<td>0.280004</td>
<td>0.035353</td>
<td>0.033417</td>
<td>0.223993</td>
<td>0.178307</td>
<td>0.338085</td>
<td>0.004555</td>
<td>0.338</td>
</tr>
<tr>
<td>component_2</td>
<td>0.365270</td>
<td>-0.064548</td>
<td>-0.298349</td>
<td>-0.168056</td>
<td>-0.478428</td>
<td>-0.366379</td>
<td>-0.082792</td>
<td>-0.361</td>
</tr>
</tbody>
</table>
In [157]: pd.DataFrame(dtm_lsa, index = example, columns = ["component_1","component_2" ])

Out[157]:

<table>
<thead>
<tr>
<th></th>
<th>component_1</th>
<th>component_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine learning is super fun</td>
<td>0.957024</td>
<td>-0.290007</td>
</tr>
<tr>
<td>Python is super, super cool</td>
<td>0.856484</td>
<td>0.516174</td>
</tr>
<tr>
<td>Statistics is cool, too</td>
<td>0.563355</td>
<td>0.826215</td>
</tr>
<tr>
<td>Data science is fun</td>
<td>0.704171</td>
<td>-0.710030</td>
</tr>
<tr>
<td>Python is great for machine learning</td>
<td>0.717284</td>
<td>-0.696781</td>
</tr>
<tr>
<td>I like football</td>
<td>0.099136</td>
<td>-0.995074</td>
</tr>
<tr>
<td>Football is great to watch</td>
<td>0.235618</td>
<td>-0.971846</td>
</tr>
</tbody>
</table>

In [132]: xs = [w[0] for w in dtm_lsa]
ys = [w[1] for w in dtm_lsa]
xs, ys

Out[132]: ([0.95702439393037975,
0.85648370973755439,
0.5635489155638729,
0.70417108799801642,
0.71728390345517212,
0.099136388426341523,
0.23561831047045331],
[-0.29000742994307793,
0.51617405490221679,
0.82621502416712989,
-0.7100303655449532,
-0.69678102861954516,
-0.99507385479158361,
-0.97184567281593082])

In [140]: # Plot scatter plot of points
   %pylab inline
   import matplotlib.pyplot as plt
   figure()
   plt.scatter(xs,ys)
xlabel('First principal component')
ylabel('Second principal component')
title('Plot of points against LSA principal components')
show()

Populating the interactive namespace from numpy and matplotlib

**Geometric picture**

```python
In [176]:
    # Plot scatter plot of points with vectors
    %pylab inline
    import matplotlib.pyplot as plt
    plt.figure()
    ax = plt.gca()
    ax.quiver(0,0,xs,ys,angles='xy',scale_units='xy',scale=1, linewidth = .01)
    ax.set_xlim([-1,1])
    ax.set_ylim([-1,1])
    xlabel('First principal component')
    ylabel('Second principal component')
    title('Plot of points against LSA principal components')
    plt.draw()
    plt.show()
```

Populating the interactive namespace from numpy and matplotlib
• We have reduced dimension from 13-dim to 2-dim (and have lost some info)
• Similar docs point in similar directions. Dissimilar docs have perpendicular (orthogonal) vectors. "Cosine similarity"
• Can use cosine similarity for search: which doc has the smallest angle with search term?

Document similarity using LSA

In [154]:
# Compute document similarity using LSA components
similarity = np.asarray(numpy.asmatrix(dtm_lsa) * numpy.asmatrix(dtm_lsa).T)
pd.DataFrame(similarity,index=example, columns=example).head(10)

Out[154]:

<table>
<thead>
<tr>
<th></th>
<th>Machine learning is super fun</th>
<th>Python is super, super cool</th>
<th>Statistics is cool, too</th>
<th>Data science is fun</th>
<th>Python is great for machine learning</th>
<th>I like football</th>
<th>Football is great to watch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine learning is</td>
<td>1.000000</td>
<td>0.669981</td>
<td>0.299536</td>
<td>0.879823</td>
<td>0.888530</td>
<td>0.383455</td>
<td>0.507335</td>
</tr>
<tr>
<td>super fun</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Python is super,</td>
<td>0.669981</td>
<td>1.000000</td>
<td>0.908975</td>
<td>0.236612</td>
<td>0.254682</td>
<td>-0.428723</td>
<td>-0.299838</td>
</tr>
<tr>
<td>super cool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Statistics is cool,</td>
<td>0.299536</td>
<td>0.908975</td>
<td>1.000000</td>
<td>-0.189940</td>
<td>-0.171606</td>
<td>-0.766296</td>
<td>-0.670217</td>
</tr>
<tr>
<td>too</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data science is fun</td>
<td>0.879823</td>
<td>0.236612</td>
<td>-0.189940</td>
<td>1.000000</td>
<td>0.999826</td>
<td>0.776342</td>
<td>0.855956</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Python is great for</td>
<td>0.888530</td>
<td>0.254682</td>
<td>-0.171606</td>
<td>0.999826</td>
<td>1.000000</td>
<td>0.764458</td>
<td>0.846169</td>
</tr>
<tr>
<td>machine learning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I like football</td>
<td>0.383455</td>
<td>-0.428723</td>
<td>-0.766296</td>
<td>0.776342</td>
<td>0.764458</td>
<td>1.000000</td>
<td>0.990417</td>
</tr>
<tr>
<td>Football is great to</td>
<td>0.507335</td>
<td>-0.299838</td>
<td>-0.670217</td>
<td>0.855956</td>
<td>0.846169</td>
<td>0.990417</td>
<td>1.000000</td>
</tr>
<tr>
<td>watch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Improvements and next steps:

- Vectorize with TFIDF (term-frequency inverse document-frequency: uses overall frequency of words to weight document-term matrix)
- Use LSA components as features in machine learning algorithm: clustering, classification, regression
- Alternative dimensionality reduction: Isomap, Random Matrix Methods, Laplacian Eigenmaps, Kernel PCA (cool names!)

Try it on your own

- List of 140k StackOverflow posts taken from a Kaggle competition
- Sample code on my personal website (www.williamgstanton.com): LSA (with 3 components instead of 2), document similarity, clustering

In []: # Import pandas for data frame functionality, CSV import, etc.
import pandas as pd

In []: # Import data as csv. Change directory to file location on your computer.
dat = pd.read_csv("/Users/wstanton/Desktop/stack_overflow/train-sample.csv")

In []: # Extract titles column from DataFrame
titles = dat.Title.values

In []: # Extract bodies column from DataFrame
bodies = dat.BodyMarkdown.values

Thanks for listening!

- Thanks to Data Science Assoc. and Meetup for inviting me to talk
- Find me on LinkedIn: www.linkedin.com/in/willstanton
- Personal website: www.williamgstanton.com
- Any further questions?