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

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]:

	cool	data	football	fun	great	learning	like	machine	python	science	statistics	s
Machine learning is super fun	0	0	0	1	0	1	0	1	0	0	0	1
Python is super, super cool	1	0	0	0	0	0	0	0	1	0	0	2
Statistics is cool, too	1	0	0	0	0	0	0	0	0	0	1	0
Data science is fun	0	1	0	1	0	0	0	0	0	1	0	0
Python is great for machine learning	0	0	0	0	1	1	0	1	1	0	0	0
I like football	0	0	1	0	0	0	1	0	0	0	0	0
Football is great to watch	0	0	1	0	1	0	0	0	0	0	0	0

- 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

- Note: order and proximity of words in documents is NOT accounted for. Called a "bag of words" representation.

In [115]:
```
# Get words that correspond to each column
vectorizer.get_feature_names()
```

Out[115]:
```
[u'cool',
 u'data',
 u'football',
 u'fun',
 u'great',
 u'learning',
 u'like',
 u'machine',
 u'python',
 u'science',
 u'statistics',
 u'super',
 u'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]:

	cool	data	football	fun	great	learning	like	mach
component_1	0.280004	0.035353	0.033417	0.223993	0.178307	0.338085	0.004555	0.338
component_2	0.365270	-0.064548	-0.298349	-0.168056	-0.478428	-0.366379	-0.082792	-0.36

- Each document is a linear combination of the LSA components

In [157]:
```
pd.DataFrame(dtm_lsa, index = example, columns = ["component_1","component_2"
])
```

Out[157]:

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

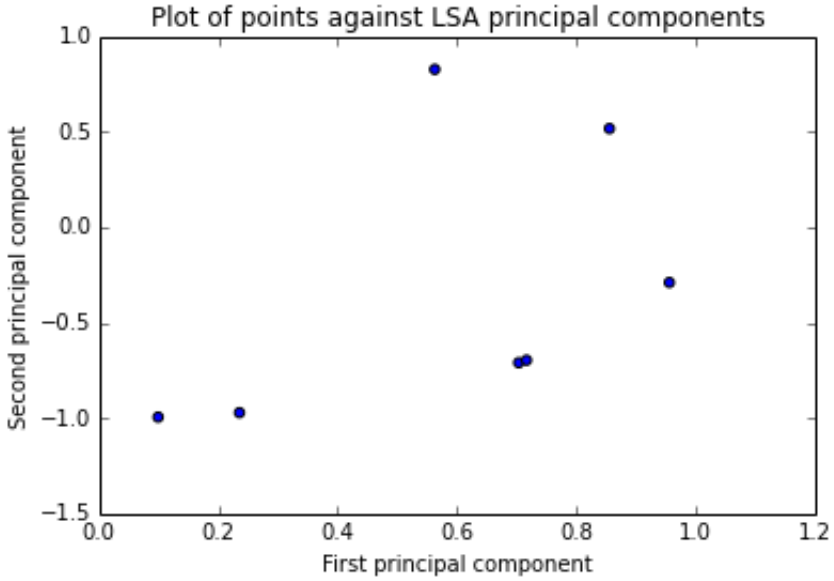
In [132]:
```
xs = [w[0] for w in dtm_lsa]
ys = [w[1] for w in dtm_lsa]
xs, ys
```

Out[132]:
```
([0.95702439393037975,
  0.85648370973755439,
  0.56335489155638729,
  0.70417108799801642,
  0.71728390345517212,
  0.099136388426341523,
  0.23561831047045331],
 [-0.29000742994307793,
  0.51617405490221679,
  0.82621502416712989,
  -0.71003033655449532,
  -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()
```

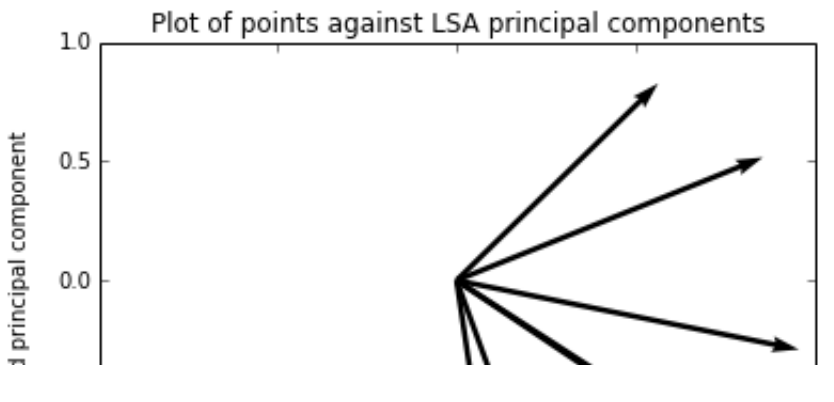
```
Populating the interactive namespace from numpy and matplotlib
```

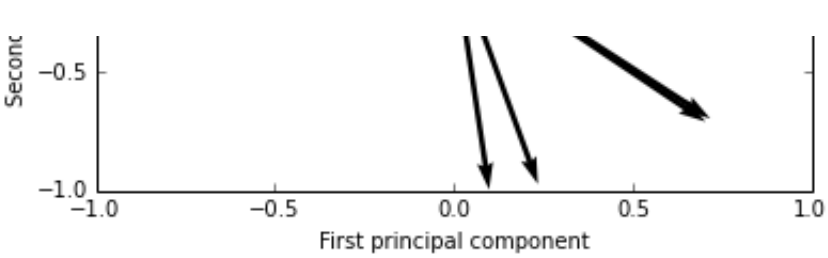


Geometric picture

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]:

	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
Machine learning is super fun	1.000000	0.669981	0.299536	0.879823	0.888530	0.383455	0.507335
Python is super, super cool	0.669981	1.000000	0.908975	0.236612	0.254682	-0.428723	-0.299838
Statistics is cool, too	0.299536	0.908975	1.000000	-0.189940	-0.171606	-0.766296	-0.670217
Data science is fun	0.879823	0.236612	-0.189940	1.000000	0.999826	0.776342	0.855956
Python is great for machine learning	0.888530	0.254682	-0.171606	0.999826	1.000000	0.764458	0.846169
I like football	0.383455	-0.428723	-0.766296	0.776342	0.764458	1.000000	0.990417
Football is great to	0.507335	-0.299838	-0.670217	0.855956	0.846169	0.990417	1.000000

watch							

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?