GENERAL ASSEMBLY

INTRO TO DATA SCIENCE

LECTURE 14: TEXT MINING

January 26, 2015

DAT11-SF

LAST TIME:

- DIMENSIONALITY REDUCTION
- PCA
- SVD

I. TEXT MINING

TEXT MINING

Text mining (text data mining) is a process of deriving information from text (unstructured data).

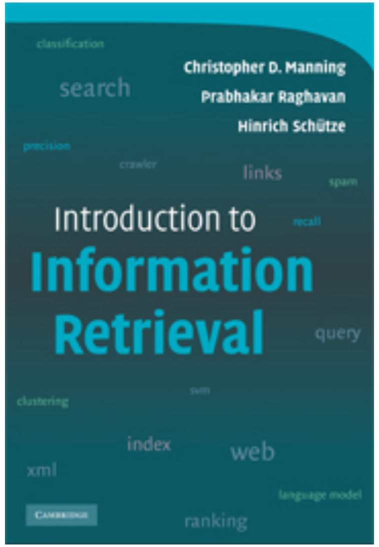
- Text categorization
- Sentiment analysis
- Document summarization
- Named entity recognition
- Information search and retrieval
- Question answering

Natural Language Processing (NLP) – is a field in computer science concerned with interaction between computer and human language, deriving meaning from text, natural language understanding

Information retrieval (IR) – finding relevant information from a collection of textual information sources.

- Dan Jurafsky and Christopher Manning, Stanford “Natural Language Processing” course

- Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, “Introduction to Information Retrieval”



II. BASIC TEXT PROCESSING

BASIC TEXT PROCESSING

1. Tokenization
2. Normalization
3. Lemmatization
4. Stemming (* IR)
5. Stop words removal (* IR)
6. Sentence segmentation

Word tokenization – splitting character sequence into tokens (units), sometimes removing certain characters- punctuation

How many tokens?

San Francisco

O’Neill

aren’t

Normalization – bringing terms to the same form, canonicalization

- Case folding: Windows -> windows, MIT-> mit (?)
- U.S.A. -> USA

Lemmatization – reducing inflectional form to a base form.
(done using vocabulary and morphological analysis of a word)

- Am, are, is -> be
- Car, cars, car’s -> car

Stemming – reducing terms to their stems (crude rule-based chopping off affixes). Often done in IR

- Automate, automatic, automation -> automat
- Ponies -> poni
- walking -> walk
- Activate -> activ

Stop words removal – dropping extremely common words, that appear to be a little value in particular IR task. Based on predefined stop word lists

A, an, and, are, as, at, be, by, for, from, has, he, in, is, it ….

Sentence segmentation – finding sentence boundary

- Period “.” can be ambiguous)
- Hand written rules
- Classifiers

III. NATURAL LANGUAGE PROCESSING

Language modeling (statistical language modeling) – assigning probabilities to a sentence (sequence of words)

$P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$

P(**high** winds tonite) > P(**large** winds tonite)

- Machine translation
- Spell correction
- Speech recognition
- Summarization
- Text classification etc

Chain rule: $P(w_1 w_2 \dots w_n) = \prod_i P(w_i \mid w_1 w_2 \dots w_{i-1})$

P(“its water is so transparent”) =

P(its) × P(water|its) × P(is|its water)

× P(so|its water is) × P(transparent|its water is so)

Markov assumption:

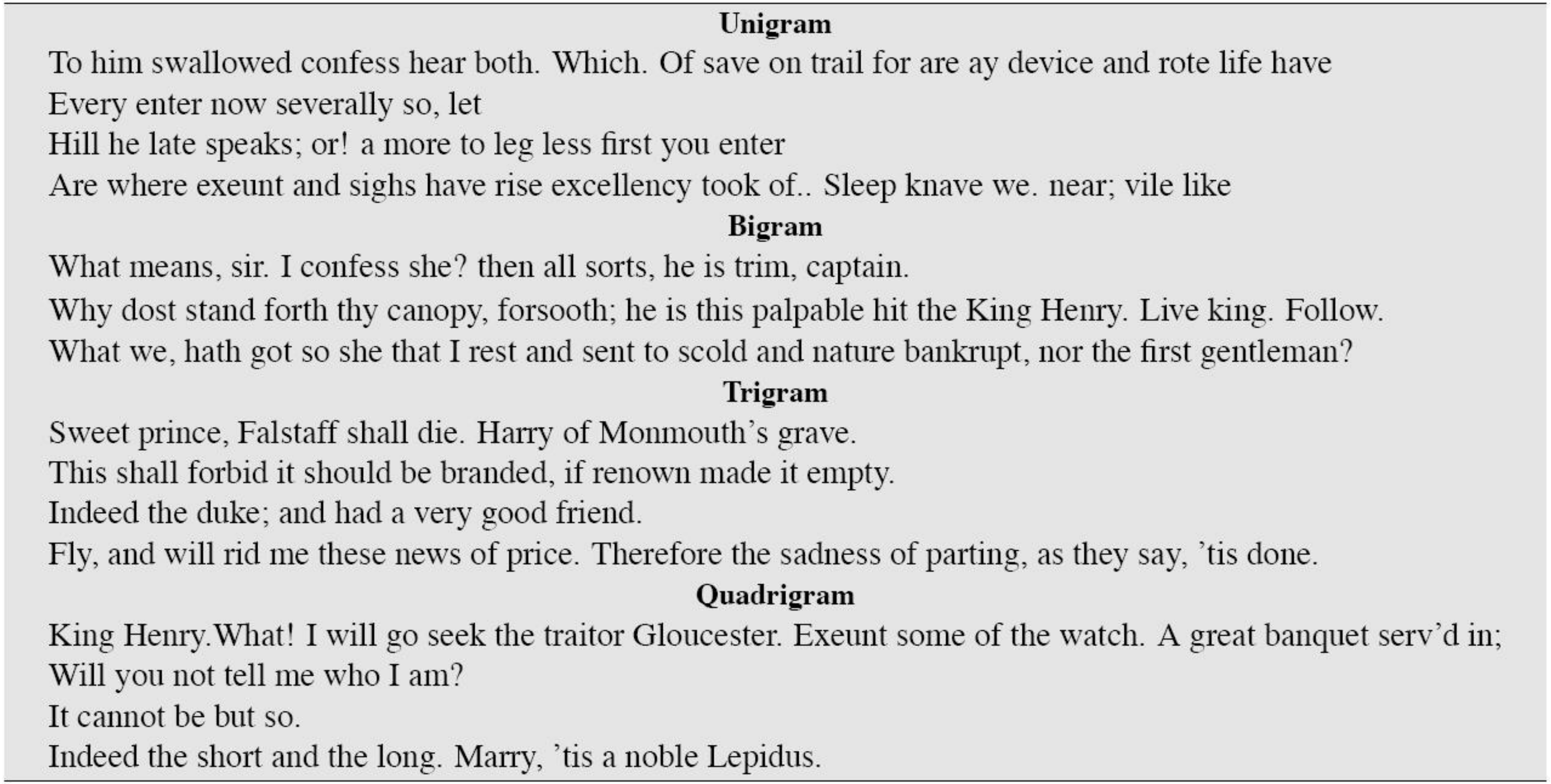
$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

- Unigram model: $P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i)$
- Bigram model:

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

- N-gram models

APPROXIMATING SHAKESPEARE



Unigram

To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have

Every enter now severally so, let

Hill he late speaks; or! a more to leg less first you enter

Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like

Bigram

What means, sir. I confess she? then all sorts, he is trim, captain.

Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.

What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?

Trigram

Sweet prince, Falstaff shall die. Harry of Monmouth's grave.

This shall forbid it should be branded, if renown made it empty.

Indeed the duke; and had a very good friend.

Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.

Quadrigram

King Henry.What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;

Will you not tell me who I am?

It cannot be but so.

Indeed the short and the long. Marry, 'tis a noble Lepidus.

THE WALL STREET JOURNAL

Unigram

Months the my and issue of year foreign new exchange’s september were recession exchange new endorsed a acquire to six executives

Bigram

Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her

Trigram

They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

MLE estimator:

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Add-1 estimate (Laplace smoothing):

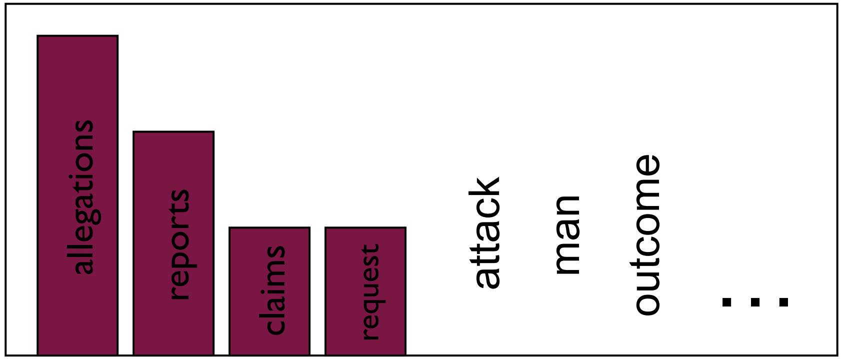
$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

PROBABILITY ESTIMATION

- When we have sparse statistics:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request

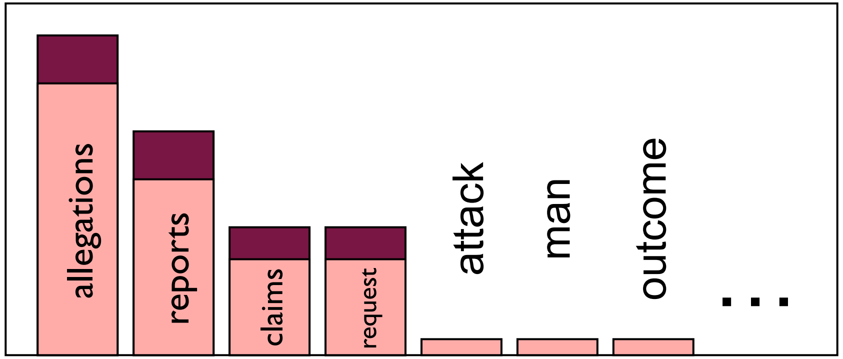
7 total



- Steal probability mass to generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other

7 total



TEXT CLASSIFICATION

1. Spam filters
2. Opinion mining/sentiment analysis
3. News aggregation
4. Scientific articles categorization
5. Text authorships identification
6. Language identification

- *Input:*
  - a document d
  - a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$

- *Output*:
  - a predicted class $c \in C$

- Rules based classification (“hand-written rules”)
- Machine learning methods (ML):
  - Naïve Bayes
  - Logistic regressions
  - SVM
  - kNN
- ML needs hand-labeled training set, (d,c) –pairs
- Extract features from text

BAG OF WORDS

```
I love this movie! It's sweet, but
with satirical humor. The dialogue is
great and the adventure scenes are
fun…  It manages to be whimsical and
romantic while laughing at the
conventions of the fairy tale genre. I
would recommend it to just about
anyone. I've seen it several times,
and I'm always happy to see it again
whenever I have a friend who hasn't
seen it yet.
```

great	2
love	2
recommend	1
laugh	1
happy	1
...	...

Bag of words representation: lose all information about order of the words in the documents, just the set (or subset) of words and their counts

NAÏVE BAYES CLASSIFIER

- For a document *d* and a class *c*

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(c \mid d) = \underset{c \in C}{\operatorname{argmax}} P(d \mid c)P(c)$$

$$= \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c)P(c)$$

NAÏVE BAYES CLASSIFIER

- Conditional independence

$$P(x_1,\ldots,x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

positions $\leftarrow$ all word positions in test document

- use the frequencies in the data

$$\hat{P}(c_j) = \frac{doccount(C = c_j)}{N_{doc}}$$

$$\hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)} \quad \Rightarrow \quad \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c)\right) + |V|}$$

MULTINOMIAL NAÏVE BAYES LEARNING

- From training corpus, extract *Vocabulary*

- Calculate $P(c_j)$ terms

  For each c_j in C do

  $docs_j \leftarrow$ all docs with class $= c_j$

$$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

- Calculate $P(w_k \mid c_j)$ terms
  - $Text_j \leftarrow$ single doc containing all $docs_j$
  - For each word w_k in *Vocabulary*

    $n_k \leftarrow$ # of occurrences of w_k in $Text_j$

$$P(w_k \mid c_j) \leftarrow \frac{n_k + 1}{n + |Vocabulary|}$$

EXAMPLE

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

$$\hat{P}(w \mid c) = \frac{count(w,c)+1}{count(c)+|V|}$$

Priors:

$P(c) = \frac{3}{4}$

$P(j) = \frac{1}{4}$

$$\hat{P}(c) = \frac{N_c}{N}$$

Conditional Probabilities:

P(Chinese|*c*) = (5+1) / (8+6) = 6/14 = 3/7

P(Tokyo|*c*) = (0+1) / (8+6) = 1/14

P(Japan|*c*) = (0+1) / (8+6) = 1/14

P(Chinese|*j*) = (1+1) / (3+6) = 2/9

P(Tokyo|*j*) = (1+1) / (3+6) = 2/9

P(Japan|*j*) = (1+1) / (3+6) = 2/9

Choosing a class:

$P(c|d5) \propto 3/4 * (3/7)^3 * 1/14 * 1/14 \approx 0.0003$

$P(j|d5) \propto 1/4 * (2/9)^3 * 2/9 * 2/9 \approx 0.0001$

SENTIMENT ANALYSIS

- Movie reviews
- Product reviews
- Public opinions
- Twitter sentiment
- ……..

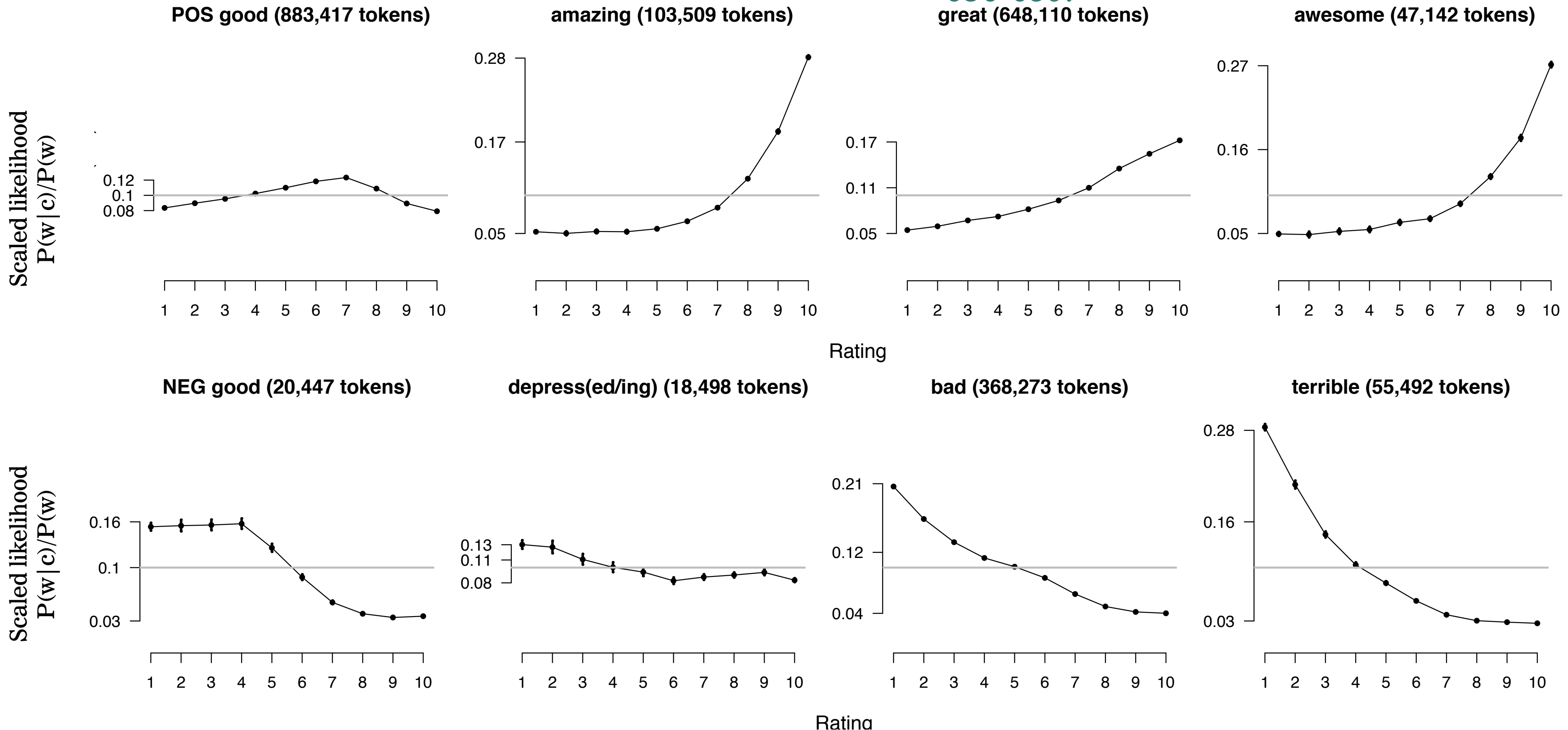
- Simple task: positive or negative
- More complex: rank the attitude 1-5
- Advanced: detect target, source, attitude

- Tokenization:
  - Html /xml markup, twitter hashtags
  - Capitalization (preserve for words in cap)
  - Phone numbers, dates
  - Emoticons :)
- Features
  - all words
  - adjectives
  - Sentiment lexicons

- Sentiment lexicons:
  - “The General Inquirer”, 1915 pos, 2291 neg words, strong/weak
  - “Linguistic Inquiry and Word Count”, 2300 words, 70 classes
  - “MPQA Subjectivity Cues Lexicon”, 2700 pos, 4912 neg
  - “Bing Liu Opinion Lexicon”, 2006 pos, 4783 neg

SENTIMENT ANALYSIS – WORD POLARITY IN IMDB

Potts, Christopher. 2011. On the negativity of negation. SALT 20, 636-659.



"This film should be brilliant. It sounds like a great plot, the actors are first grade, and the supporting cast is good as well, and Stallone is attempting to deliver a good performance. However, it **can't hold up**."

"Well as usual Keanu Reeves is nothing special, but surprisingly, the very talented Laurence Fishbourne is **not so good** either, I was surprised".

- Negation:

Add NOT_ to every word between negation and following punctuation:

```
didn’t like this movie , but I

didn’t NOT_like NOT_this NOT_movie but I
```

IV. INFORMATION RETRIEVAL

- Goal: Retrieve documents with information that is relevant to the user's information need and helps the user complete a task

- *Precision* : Fraction of retrieved docs that are relevant to the user's information need
- *Recall* : Fraction of relevant docs in collection that are retrieved

BOOLEAN INFORMATION RETRIEVAL

- Which plays of Shakespeare contain the words ***Brutus*** *AND* ***Caesar*** but *NOT* ***Calpurnia***?
- One could `grep` all of Shakespeare's plays for ***Brutus*** and ***Caesar,*** then strip out lines containing ***Calpurnia***?
- Why is that not the answer?

- Slow (for large corpora)
- *NOT* ***Calpurnia*** is non-trivial

TERM-DOCUMENT INCIDENCE MATRIX

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Each document is represented by a binary column vector $\in \{0,1\}^{|V|}$

RESULTS EVALUATION

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Each document is represented by a count vector with term freq

Bag of words model (no word position or ordering)

TF-IDF

- Rare terms in collection are more informative than frequent terms
- A document containing this term is very likely to be relevant to the query
- tf – term frequency in a document
- df – number of documents that contain term t

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

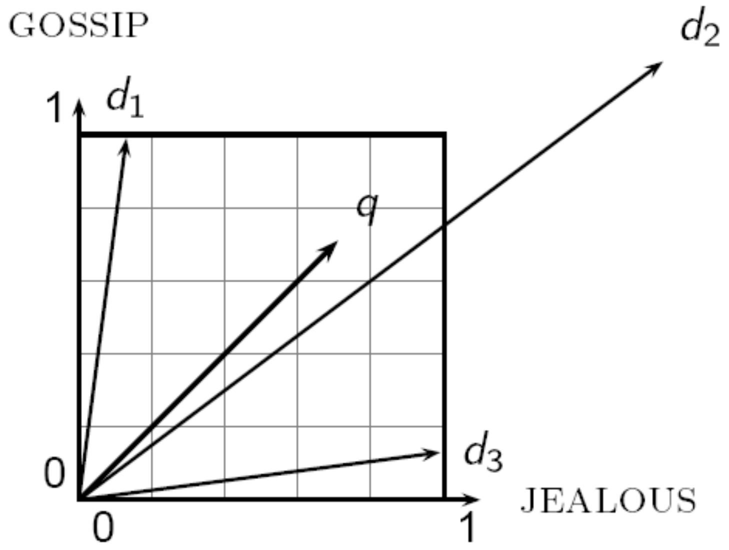
TERM-DOCUMENT WEIGHT MATRIX

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Each document is now represented by a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$

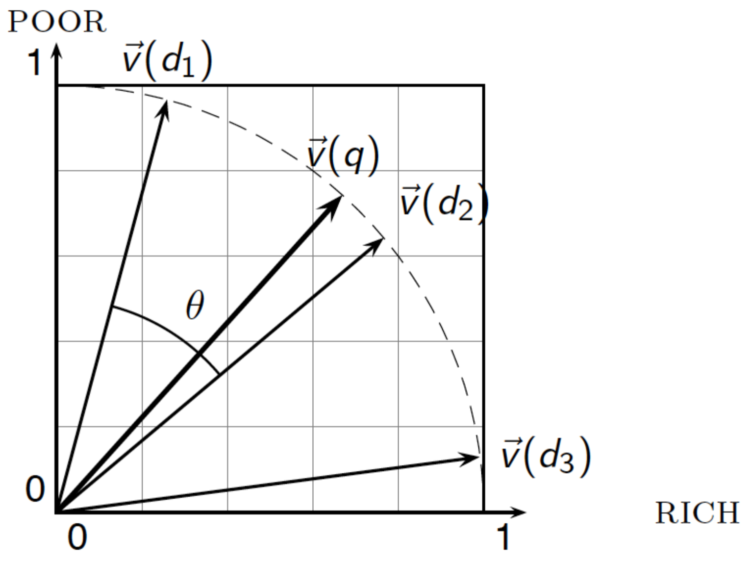
VECTOR-SPACE MODEL

- Now we have a |V|-dimensional vector space
- Terms are axes of the space
- Documents are points or vectors in this space
- Very high-dimensional
- These are very sparse vectors



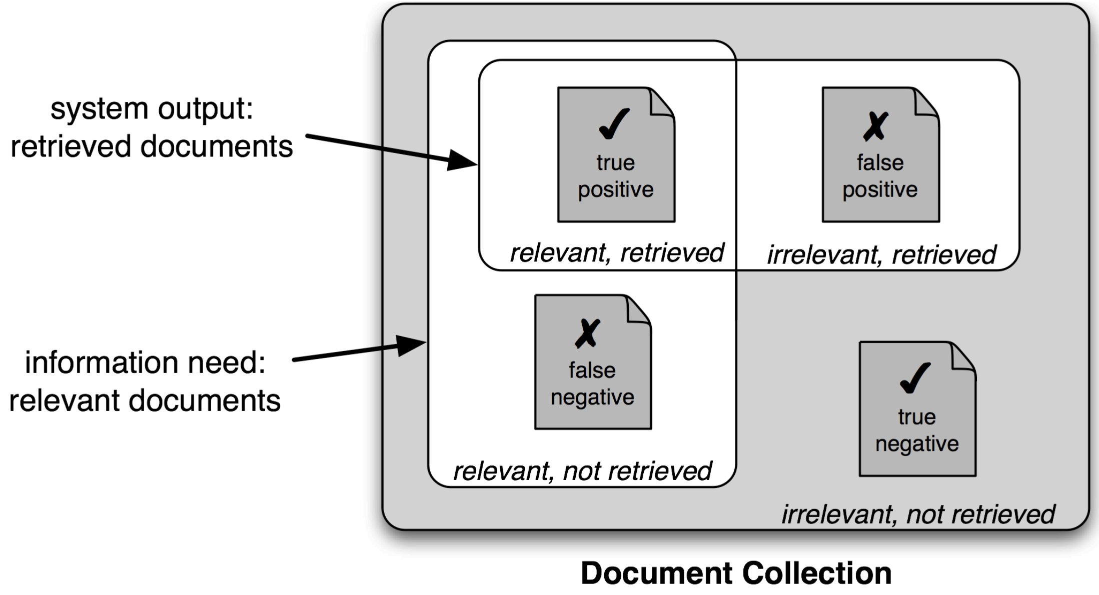
RANKED RETRIEVAL

- Represent the query as a weighted tf-idf vector
- Represent each document as a weighted tf-idf vector
- Compute the cosine similarity score for the query vector and each document vector
- Rank documents with respect to the query by score
- Return the top *K* (e.g., *K* = 10) to the user



$$\cos(\vec{q},\vec{d}) = \frac{\vec{q}\bullet\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|}\bullet\frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

PRECISION-RECALL

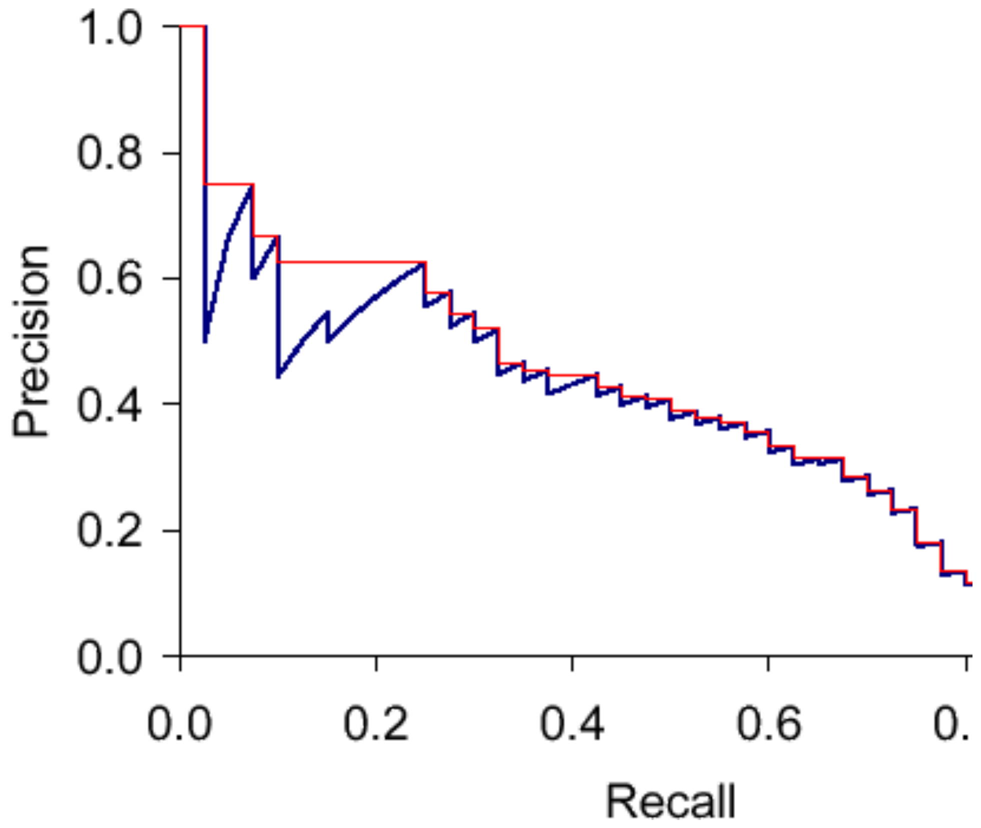


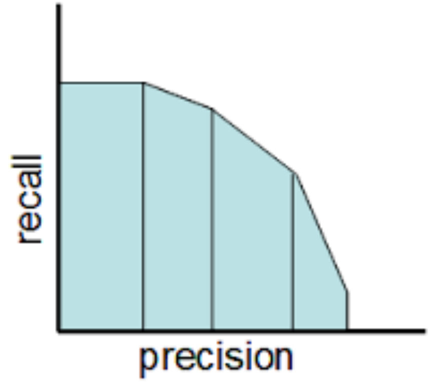
$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

PRECISION-RECALL CURVE





$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

V. LAB: TEXT PROCESSING IN NLTK