CSE 258 – Lecture 9

Data Mining and Predictive Analytics

Text Mining

Administrivia

- Midterms will be **in class** next Wednesday
- We'll do prep on Monday, and come back to the rest of text mining in week 7
- No class the Monday after the midterm (President's Day)

Prediction tasks involving text

What kind of quantities can we model, and what kind of prediction tasks can we solve using **text?**

Prediction tasks involving text

Does this article have a positive or negative sentiment about the subject being discussed?

What can stop US Postal Service trucks? The inexorable march of time

The ageing fleet of delivery vehicles is long past due an overhaul. Among the common-sense upgrades employees want: air conditioning and more workspace

Neither snow nor rain nor heat nor gloom of night stays these trucks - but time, it turns out, will. Photograph: Bill Sikes/AP

For the better part of the last 30 years, the flatulent buzz of the US Postal Service's boxy delivery vans - audible as they lighted from mailbox to mailbox - has been a familiar sound to most Americans. Neither snow nor rain nor heat nor gloom of night stays the USPS's mail trucks from the swift completion of their appointed

Prediction tasks involving text

What is the category/subject/topic of this article?

Apple Is Forming an Auto Team

By BRIAN X. CHEN and MIKE ISAAC FEB. 19, 2015

Email

Share

Tweet

Save

More

SAN FRANCISCO — While Apple has been preparing to release its first wearable computers, the company has also been busy assembling a team to work on an automobile.

The company has collected about 200 people over the last few years — both from inside Apple and potential competitors like Tesla — to develop technologies for an electric car, according to two people with knowledge of the company's plans, who asked not to be named because the plans were private.

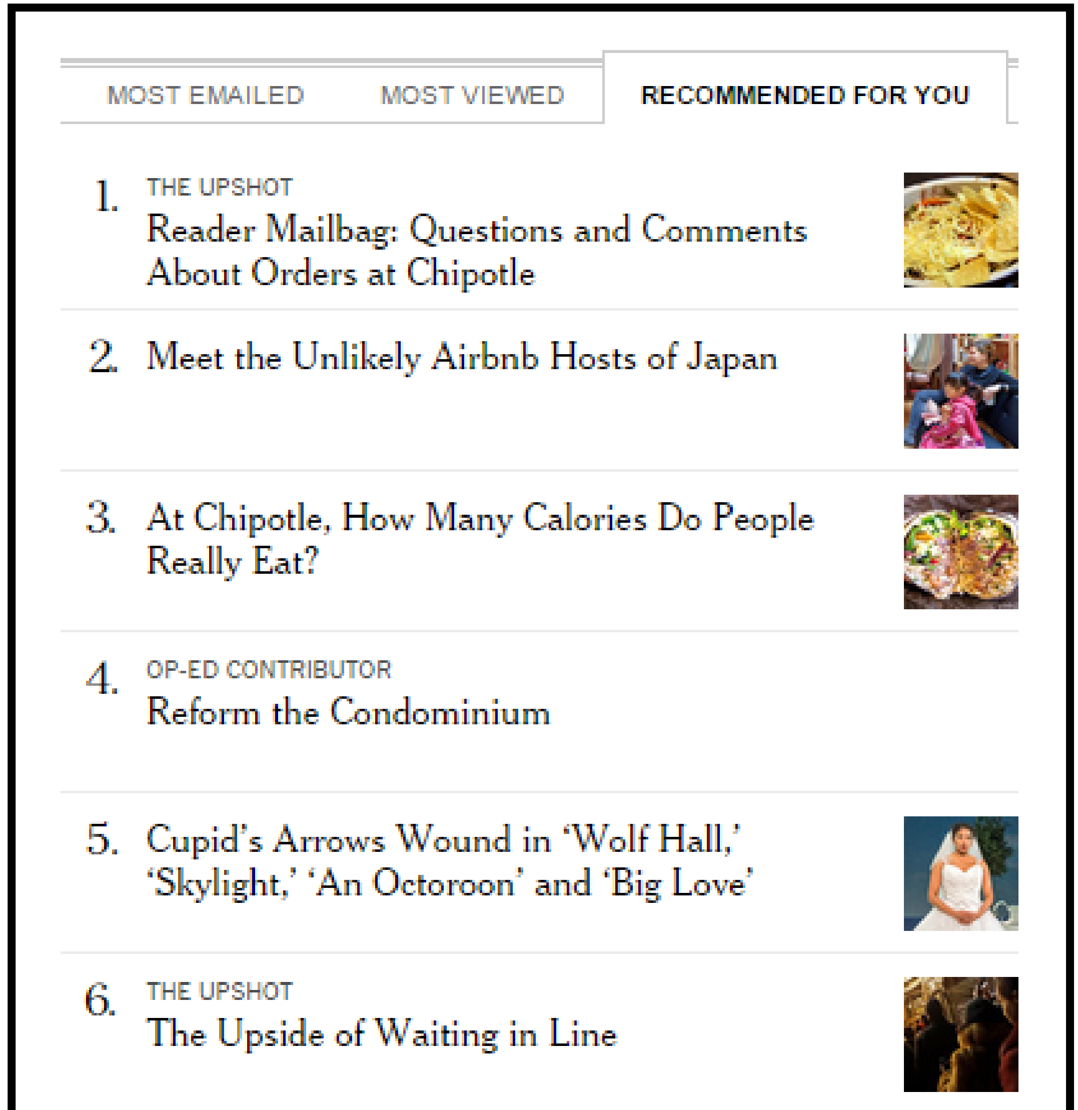
Electric car batteries being prepared for shipment at the A123 Systems plant in Livonia, Mich in 2012. Apple has hired engineers from A123 Systems. Stephen McGee for The New York Times

The car project is still in its prototype phase, one person said, meaning it is probably many years away from being a viable product and might never reach the mass market if the quality of the vehicle fails to impress Apple's executives.

It could also go nowhere if Apple struggles to find a compelling business opportunity in automobiles, a business that typically has much lower sales margins than

Prediction tasks involving text

Which of these articles are relevant to my interests?



Prediction tasks involving text

Find me articles similar to this one

Meatloaf That Conquers the Mundane

FEB. 13, 2015

City Kitchen

By DAVID TANIS

Email

Share

Tweet

Pin

Save

More

I was raised on Midwestern meatloaf. My mother's dependable recipe did not vary: Ground beef, grated onion and carrot and a little oatmeal were the main ingredients, along with a dash of "seasoned salt." A ribbon of bottled chili sauce ran down a gully in the center.

Served hot, accompanied by Tater Tots, it was dinner. Served cold for lunch, it was always a sandwich on white bread, with potato chips on the side. It was usually moist and tasty but never remarkable, and there was no way you could call it anything but meatloaf.

Do I harbor a kind of nostalgia for it? Yes. But would I use that recipe now? I think not.

Evan Sung for The New York Times

I have a friend from Brussels who loves to entertain. Of his dinner party repertoire, one dish is most requested and admired. It is pain de veau, served with a vermouth-splashed mushroom sauce. In French, it sounds elegant. Translated into English — veal loaf — it sounds dull.

The Italian word for meatloaf is polpettone. (Polpette are Italian meatballs; polpettine are meatballs, too, but more diminutive.) This substantial family-size meatball, whether ovoid or elongated, plain or fancy, served with tomato sauce or not, is beloved both in Italy and in Italian communities throughout the world. Aside from its melodic, polysyllabic name, polpettone is always well seasoned, prepared with care and served with gusto.

It is usually a combination of different kinds of ground meat, typically beef, pork and veal in equal parts. Grated cheese and herbs are

RELATED COVERAGE

City Kitchen: How to Make Polpettone, Step by Step FEB. 13, 2015

RECIPES FROM COOKING

Polpettone with Spinach and Provolone

By David Tanis

related articles

Prediction tasks involving text

Which of these reviews am I most likely to agree with or find helpful?

Most Helpful Customer Reviews

1,900 of 1,928 people found the following review helpful

Le Creuset on a budget

By N. Lafond on October 24, 2007

Color Name: Caribbean Blue | Size Name: 6 qt | **Verified Purchase**

Enamel on cast iron cookware like this, was, until recently, only available from makers like Le Creuset. Lately, several lower cost makers have come on the scene, like Target and Innova. The new budget priced Lodge cookware is in the same price range as the low cost alternatives but completely out performs them.

I have all of the brands I have mentioned. The Lodge is the same weight as the Le Creuset which is much heavier than the other budget models. The ridge where the lid and sides meet is a matt black porcelain on the Lodge and Le Creuset but is just exposed cast iron for the other budget models (which leads to rusting if you are not careful). The porcelain resists staining (even tomato sauces) in the Lodge and Le Creuset but the other budget models stain very easily. And finally, the Lodge and Le Creuset maintain a very polished interior finish that resists sticking which others do not. So, I see no performance differences at all between the Le Creuset and the Lodge whereas the comparably priced budget models are certainly inferior.

If you plan of using these pots very heavily (every day for example) you might want to upgrade to the higher priced Lodge product. It has 4 coatings of enamel as opposed to 2 in this model. But if you use them once or twice a week I dont think you will need the added wear resistance.

47 Comments | Was this review helpful to you? Yes No

1,105 of 1,164 people found the following review helpful

OK pot, Great Price. Some flaws.

By J. G. Pavlovich on March 2, 2008

Color Name: Island Spice Red | Size Name: 6 qt | **Verified Purchase**

This is a terrific value. The quality and performance match my Le Creuset pieces at a fraction of the price. The only slight design flaw I have found is that the rounded bottom makes browning large pieces of meat awkward. Other than that I have no complaints. Even heating. Easy clean up. I use it several times a week.
UPDATE: I found a second minor problem. The inside rim of the lid has a couple of raised spots which prevent the lid from seating tightly. This causes steam to escape much faster than I would like during a long braise or stew.

Update 2: Three years in I am dropping my rating to three stars. It's still a decent pot at a bargain price, but it will not be an heirloom piece like my Le Creuset. The loose fitting lid turns

Prediction tasks involving text

Which of these sentences best summarizes people's opinions?

Prediction tasks involving text

Which sentences refer to which aspect of the product?

'Partridge in a Pear Tree', brewed by 'The Bruery'

Dark brown with a light tan head, minimal lace and low retention. Excellent aroma of dark fruit, plum, raisin and red grape with light vanilla, oak, caramel and toffee. Medium thick body with low carbonation. Flavor has strong brown sugar and molasses from the start over bready yeast and a dark fruit and plum finish. Minimal alcohol presence. Actually, this is a nice quad.

Feel: 4.5 Look: 4 Smell: 4.5 Taste: 4 Overall: 4

Today

Using **text** to solve predictive tasks

- How to represent **documents** using **features?**
- Is text **structured** or **unstructured?**
- Does structure actually help us?
- How to account for the fact that most words may not convey much information?
- How can we find **low-dimensional** structure in text?

CSE 258 – Lecture 9

Web Mining and Recommender Systems

Bag-of-words models

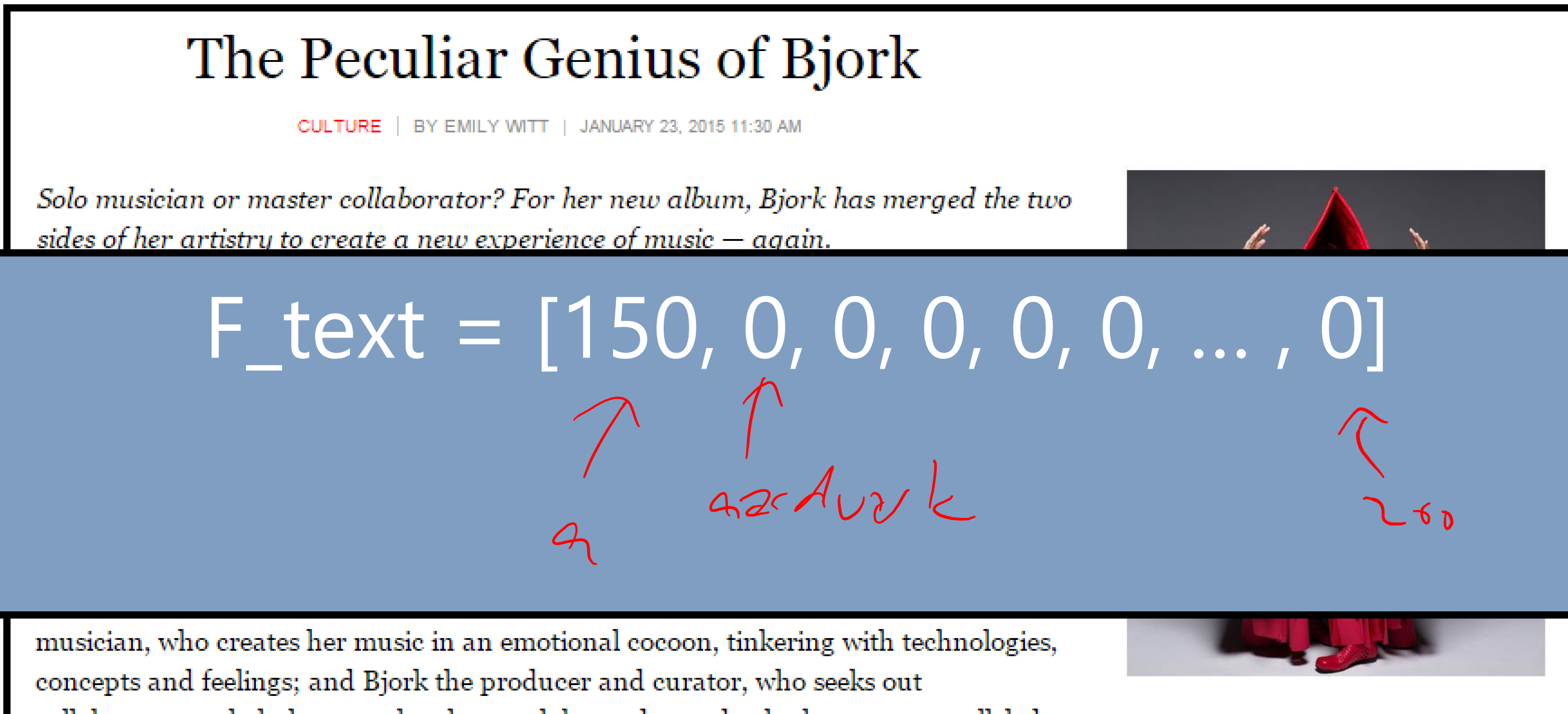
Feature vectors from text

We'd like a fixed-dimensional representation of documents, i.e., we'd like to describe them using **feature vectors**

This will allow us to compare documents, and associate weights with particular features to solve predictive tasks etc. (i.e., the kind of things we've been doing every week)

Feature vectors from text

Option 1: just count how many times each word appears in each document



The Peculiar Genius of Bjork

CULTURE | BY EMILY WITT | JANUARY 23, 2015 11:30 AM

Solo musician or master collaborator? For her new album, Bjork has merged the two sides of her artistry to create a new experience of music — again.

musician, who creates her music in an emotional cocoon, tinkering with technologies, concepts and feelings; and Bjork the producer and curator, who seeks out

F_text = [150, 0, 0, 0, 0, 0, ... , 0]

a

aardvark

zoo

Feature vectors from text

Option 1: just count how many times each word appears in each document

Dark brown with a light tan head, minimal lace and low retention. Excellent aroma of dark fruit, plum, raisin and red grape with light vanilla, oak, caramel and toffee. Medium thick body with low carbonation. Flavor has strong brown sugar and molasses from the start over bready yeast and a dark fruit and plum finish. Minimal alcohol presence. Actually, this is a nice quad.	yeast and minimal red body thick light a Flavor sugar strong quad. grape over is molasses lace the low and caramel fruit Minimal start and toffee. dark plum, dark brown Actually, alcohol Dark oak, nice vanilla, has brown of a with presence. light carbonation. bready from retention. with finish. with and this and plum and head, fruit, low a Excellent raisin aroma Medium tan

These two documents have **exactly** the same representation in this model, i.e., we're completely **ignoring** syntax.
This is called a "bag-of-words" model.

Feature vectors from text

Option 1: just count how many times each word appears in each document

We've already seen some (potential) problems with this type of representation in week 3 (dimensionality reduction), but let's see what we can do to get it working

Feature vectors from text

50,000 reviews are available on :
http://jmcauley.ucsd.edu/cse258/data/beer/beer_50000.json
(see course webpage, from week 1)

Code on:
http://jmcauley.ucsd.edu/cse258/code/week5.py

Feature vectors from text

Q1: How many words are there?

```
wordCount = defaultdict(int)
for d in data:
  for w in d['review/text'].split():
    wordCount[w] += 1

print len(wordCount)
```

Feature vectors from text

2: What if we remove capitalization/punctuation?

```
wordCount = defaultdict(int)
punctuation = set(string.punctuation)
for d in data:
  for w in d['review/text'].split():
    w = ''.join([c for c in w.lower() if not c in punctuation])
    wordCount[w] += 1

print len(wordCount)
```

Feature vectors from text

3: What if we merge different inflections of words?

drinks → drink
drinking → drink
drinker → drink

argue → argu
arguing → argu
argues → argu
arguing → argu
argus → argu

3: What if we merge different inflections of words?

This process is called “stemming”

- The first stemmer was created by Julie Beth Lovins (in 1968!!)
- The most popular stemmer was created by Martin Porter in 1980

3: What if we merge different inflections of words?

The algorithm is (fairly) simple but depends on a huge number of rules

Step 1a

```
SSES -> SS                  caresses  ->  caress
IES  -> I                   ponies    ->  poni
                            ties      ->  ti
SS   -> SS                  caress    ->  caress
S    ->                     cats      ->  cat
```

Step 1b

```
(m>0) EED -> EE             feed      ->  feed
                            agreed    ->  agree
(*v*) ED  ->                plastered ->  plaster
                            bled      ->  bled
(*v*) ING ->                motoring  ->  motor
                            sing      ->  sing
```

If the second or third of the rules in Step 1b is successful, the following is done:

```
AT -> ATE                   conflat(ed)  ->  conflate
BL -> BLE                   troubl(ed)   ->  trouble
IZ -> IZE                   siz(ed)      ->  size
(*d and not (*L or *S or *Z))
   -> single letter
                            hopp(ing)    ->  hop
                            tann(ed)     ->  tan
                            fall(ing)    ->  fall
                            hiss(ing)    ->  hiss
                            fizz(ed)     ->  fizz
(m=1 and *o) -> E           fail(ing)    ->  fail
                            fil(ing)     ->  file
```

The rule to map to a single letter causes the removal of one of the double letter pair. The -E is put back on … recognised la…

Step 1c

```
(*v*) Y -> I                happy        ->  happi
```

Step 2

```
(m>0) ATIONAL ->  ATE       relational      ->  relate
(m>0) TIONAL  ->  TION      conditional     ->  condition
                            rational        ->  rational
(m>0) ENCI    ->  ENCE      valenci         ->  valence
(m>0) ANCI    ->  ANCE      hesitanci       ->  hesitance
(m>0) IZER    ->  IZE       digitizer       ->  digitize
(m>0) ABLI    ->  ABLE      conformabli     ->  conformable
(m>0) ALLI    ->  AL        radicalli       ->  radical
(m>0) ENTLI   ->  ENT       differentli     ->  different
(m>0) ELI     ->  E         vileli         - >  vile
(m>0) OUSLI   ->  OUS       analogousli     ->  analogous
(m>0) IZATION ->  IZE       vietnamization  ->  vietnamize
(m>0) ATION   ->  ATE       predication     ->  predicate
(m>0) ATOR    ->  ATE       operator        ->  operate
(m>0) ALISM   ->  AL        feudalism       ->  feudal
(m>0) IVENESS ->  IVE       decisiveness    ->  decisive
(m>0) FULNESS ->  FUL       hopefulness     ->  hopeful
(m>0) OUSNESS ->  OUS       callousness     ->  callous
(m>0) ALITI   ->  AL        formaliti       ->  formal
(m>0) IVITI   ->  IVE       sensitiviti     ->  sensitive
(m>0) BILITI  ->  BLE       sensibiliti     ->  sensible
```

The test for the string S1 can be made fast by doing a program switch on the penultimate letter of the word being tested. This gives a fairly even breakdown of the possible values of the string S1. It will be seen in fact that the S1-strings in step 2 are presented here in the alphabetical order of their penultimate letter. Similar techniques may be applied in the other steps.

Step 3

```
(m>0) NESS  ->              goodness        ->  good
```

Step 4

```
(m>1) AL    ->              revival         ->  reviv
(m>1) ANCE  ->              allowance       ->  allow
(m>1) ENCE  ->              inference       ->  infer
(m>1) ER    ->              airliner        ->  airlin
(m>1) IC    ->              gyroscopic      ->  gyroscop
(m>1) ABLE  ->              adjustable      ->  adjust
(m>1) IBLE  ->              defensible      ->  defens
(m>1) ANT   ->              irritant        ->  irrit
(m>1) EMENT ->              replacement     ->  replac
(m>1) MENT  ->              adjustment      ->  adjust
(m>1) ENT   ->              dependent       ->  depend
(m>1 and (*S or *T)) ION -> adoption        ->  adopt
(m>1) OU    ->              homologou       ->  homolog
(m>1) ISM   ->              communism       ->  commun
(m>1) ATE   ->              activate        ->  activ
(m>1) ITI   ->              angulariti      ->  angular
(m>1) OUS   ->              homologous      ->  homolog
(m>1) IVE   ->              effective       ->  effect
(m>1) IZE   ->              bowdlerize      ->  bowdler
```

The suffixes are now removed. All that remains is a little tidying up.

Step 5a

```
(m>1) E     ->              probate         ->  probat
                            rate            ->  rate
(m=1 and not *o) E ->       cease           ->  ceas
```

Step 5b

```
ontrol
oll
```

http://telemat.det.unifi.it/book/2001/wchange/download/stem_porter.html

Feature vectors from text

3: What if we merge different inflections of words?

```
wordCount = defaultdict(int)
punctuation = set(string.punctuation)
stemmer = nltk.stem.porter.PorterStemmer()
for d in data:
  for w in d['review/text'].split():
    w = ''.join([c for c in w.lower() if not c in punctuation])
    w = stemmer.stem(w)
    wordCount[w] += 1

print len(wordCount)
```

Feature vectors from text

3: What if we merge different inflections of words?

- Stemming is **critical** for retrieval-type applications (e.g. we want Google to return pages with the word "cat" when we search for "cats")
- Personally I tend not to use it for predictive tasks. Words like "waste" and "wasted" may have different meanings (in beer reviews), and we're throwing that away by stemming

Feature vectors from text

4: Just discard extremely rare words…

```
counts = [(wordCount[w], w) for w in wordCount]
counts.sort()
counts.reverse()

words = [x[1] for x in counts[:1000]]
```

- Pretty unsatisfying but at least we can get to some inference now!

Feature vectors from text

Let's do some inference!

Problem 1: Sentiment analysis

Let's build a predictor of the form:

$$f(\text{text}) \rightarrow \text{rating}$$

using a model based on linear regression:

$$\text{rating} \simeq \alpha + \sum_{w \in \text{text}} \text{count}(w) \cdot \theta_w$$

[X, 1, 0, 6, 0, 0, 2, 0, 0, 3, 0, 0, 0]

Code: http://jmcauley.ucsd.edu/cse258/code/week5.py

Feature vectors from text

What do the parameters look like?

$$\theta_{\text{fantastic}} = 0.143$$

$$\theta_{\text{watery}} = -0.163$$

$$\theta_{\text{and}} = -0.008$$

$$\theta_{\text{me}} = -0.037$$

Feature vectors from text

Why might parameters associated with "and", "of", etc. have non-zero values?

- Maybe they have meaning, in that they might frequently appear slightly more often in positive/negative phrases
- Or maybe we're just measuring the length of the review…

How to fix this (and is it a problem)?

1) Add the length of the review to our feature vector
2) Remove stopwords

Feature vectors from text

Removing stopwords:

```
from nltk.corpus import stopwords
stopwords.words(“english”)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',
'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself',
'she', 'her', 'hers', 'herself', 'it', 'its', 'itself', 'they', 'them',
'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this',
'that', 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been',
'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing',
'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until',
'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between',
'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to',
'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again',
'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why',
'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other',
'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than',
'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now']
```

Feature vectors from text

Why remove stopwords?

some (potentially inconsistent) reasons:

- They convey little information, but are a substantial fraction of the corpus, so we can reduce our corpus size by ignoring them
- They **do** convey information, but only by being correlated by a feature that we don't want in our model
- They make it more difficult to reason about which features are informative (e.g. they might make a model harder to visualize)
- We're confounding their importance with that of phrases they appear in (e.g. words like "The Matrix", "The Dark Night", "The Hobbit" might predict that an article is about movies)

so use n-grams!

Feature vectors from text

We can build a richer predictor by using **n-grams**

e.g. “Medium thick body with low carbonation.“

unigrams: `[“medium”, “thick”, “body”, “with”, “low”, “carbonation”]`

bigrams: `[“medium thick”, “thick body”, “body with”, “with low”, “low carbonation”]`

trigrams: `[“medium thick body”, “thick body with”, “body with low”, “with low carbonation”]`

etc.

Feature vectors from text

We can build a richer predictor by using **n-grams**

- Fixes some of the issues associated with using a bag-of-words model – namely we recover some basic **syntax** – e.g. "good" and "not good" will have different weights associated with them in a sentiment model
- Increases the **dictionary size** by a lot, and increases the sparsity in the dictionary even further
- We might end up double (or triple-)-counting some features (e.g. we'll predict that "Adam Sandler", "Adam", and "Sandler" are associated with negative ratings, even though they're all referring to the same concept)

Feature vectors from text

We can build a richer predictor by using **n-grams**

- This last problem (that of double counting) is bigger than it seems: We're **massively** increasing the number of features, but possibly increasing the number of **informative** features only slightly
- So, for a **fixed-length** representation (e.g. 1000 most-common words vs. 1000 most-common words+bigrams) the bigram model will quite possibly perform **worse** than the unigram model

(homework exercise?)

Feature vectors from text

Other prediction tasks:

Problem 2: Multiclass classification

Let's build a predictor of the form:

$$f(\text{text}) \to \text{class label}$$

(or even *f*(text) → {1 star, 2 star, 3 star, 4 star, 5 star})
using a probabilistic classifier:

$$p(\text{class} = c | \text{text})$$

Feature vectors from text

Recall: multinomial distributions

Want:

$$\sum_c p(\text{class} = c|\text{text}) = 1$$

When there were **two** classes, we used a sigmoid function to ensure that probabilities would sum to 1:

$$p(\text{label}|x) = \sigma(X_i \cdot \theta) \qquad p(\neg\text{label}|x) = 1 - \sigma(X_i \cdot \theta)$$

$$= \frac{1}{1+e^{-X_i \cdot \theta}} \qquad \frac{e^{-X_i \cdot \theta}}{1+e^{-X_i \cdot \theta}}$$

Feature vectors from text

Recall: multinomial distributions

With **many** classes, we can use the same idea, by exponentiating linear predictors and normalizing:

$$p(\text{class} = c|x) = \frac{1}{Z}\exp\langle \theta_c, x\rangle = \frac{e^{X_i \cdot \theta_c}}{\sum_{c'} e^{X_i \cdot \theta_{c'}}}$$

Each class has its own set of parameters

We can optimize this model exactly as we did for logistic regression, i.e., by computing the (log) likelihood and fitting parameters to maximize it

Feature vectors from text

How to apply this to text classification?

$$p(\text{class} = c|x) = \frac{1}{Z} \exp\langle\theta_c, x\rangle = \frac{\exp\langle\theta_c, x\rangle}{\sum_{c'} \exp\langle\theta_{c'}, x\rangle}$$

$$\langle\theta_c, x\rangle = \theta_{c,0} + \sum_{w \in \text{text}} \text{count}(w) \cdot \theta_{c,w}$$

Background probability of this class

Score associated with the word *w* appearing in the class *c*

Feature vectors from text

$\theta_{c,w}$ is now a “descriptor” of each category, with high weights for words that are likely to appear in the category

high weights: $\theta_{\text{5-star},\text{‘great’}}, \theta_{\text{5-star},\text{‘fantastic’}}, \theta_{\text{1-star},\text{‘terrible’}}$

low weights: $\theta_{\text{1-star},\text{‘great’}}, \theta_{\text{1-star},\text{‘fantastic’}}, \theta_{\text{5-star},\text{‘terrible’}}$

So far...

Bags-of-words representations of text

- Stemming & stopwords
- Unigrams & N-grams
- Sentiment analysis & text classification

Questions?

Further reading:

- Original stemming paper

"Development of a stemming algorithm" (Lovins, 1968):
http://mt-archive.info/MT-1968-Lovins.pdf

- Porter's paper on stemming

"An algorithm for suffix stripping" (Porter, 1980):
http://telemat.det.unifi.it/book/2001/wchange/download/stem_porter.html

CSE 258 – Lecture 9

Web Mining and Recommender Systems

Case study: inferring aspects from multi-dimensional reviews

A (very quick) case study

How can we estimate which words in a review refer to which sensory aspects?

'Partridge in a Pear Tree', brewed by 'The Bruery'

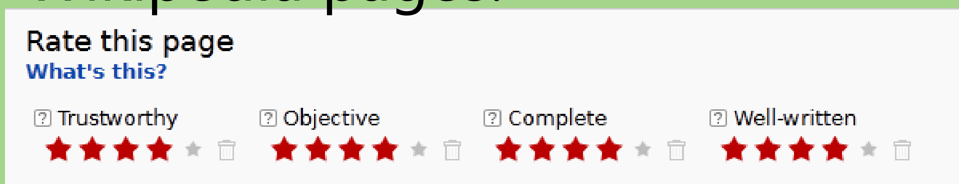
Dark brown with a light tan head, minimal lace and low retention. Excellent aroma of dark fruit, plum, raisin and red grape with light vanilla, oak, caramel and toffee. Medium thick body with low carbonation. Flavor has strong brown sugar and molasses from the start over bready yeast and a dark fruit and plum finish. Minimal alcohol presence. Actually, this is a nice quad.

Feel: 4.5 Look: 4 Smell: 4.5 Taste: 4 Overall: 4

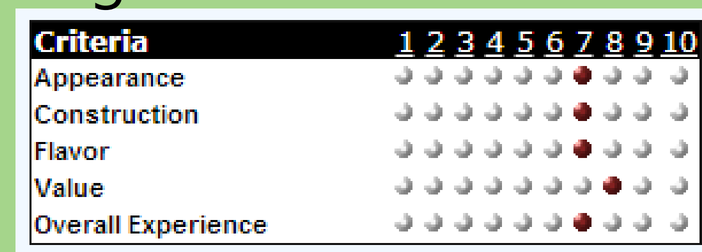
Aspects of opinions

There are lots of settings in which people's opinions cover many dimensions:

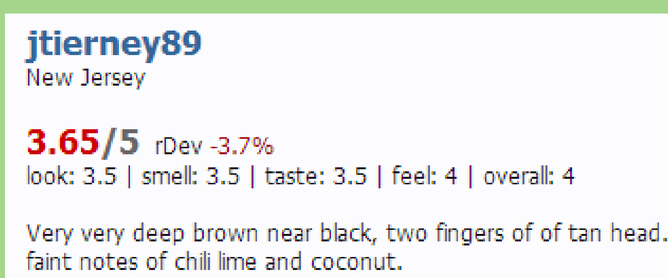
Wikipedia pages:



Cigars:



Beers:



Audiobooks:

Hotels:

Aspects of opinions

Further reading on this problem:

- Brody & Elhadad
  "An unsupervised aspect-sentiment model for online reviews"
- Gupta, Di Fabbrizio, & Haffner
  "Capturing the stars: predicting ratings for service and product reviews"
- Ganu, Elhadad, & Marian
  "Beyond the stars: Improving rating predictions using review text content"
- Lu, Ott, Cardie, & Tsou
  "Multi-aspect sentiment analysis with topic models"
- Rao & Ravichandran
  "Semi-supervised polarity lexicon induction"
- Titov & McDonald
  "A joint model of text and aspect ratings for sentiment summarization"

Aspects of opinions

If we can uncover these dimensions, we might be able to:

- Build sentiment models for each of the different aspects
- Summarize opinions according to each of the sensory aspects
- Predict the multiple dimensions of ratings from the text alone

- But also: **understand** the types of positive and negative language that people use

Aspects of opinions

Task: given (multidimensional) ratings and plain-text reviews, predict which sentences in the review refer to which aspect

Input:	Output:
'Partridge in a Pear Tree', brewed by 'The Bruery' Dark brown with a light tan head, minimal lace and low retention. Excellent aroma of dark fruit, plum, raisin and red grape with light vanilla, oak, caramel and toffee. Medium thick body with low carbonation. Flavor has strong brown sugar and molasses from the start over bready yeast and a dark fruit and plum finish. Minimal alcohol presence. Actually, this is a nice quad. Feel: 4.5 Look: 4 Smell: 4.5 Taste: 4 Overall: 4	'Partridge in a Pear Tree', brewed by 'The Bruery' Dark brown with a light tan head, minimal lace and low retention. Excellent aroma of dark fruit, plum, raisin and red grape with light vanilla, oak, caramel and toffee. Medium thick body with low carbonation. Flavor has strong brown sugar and molasses from the start over bready yeast and a dark fruit and plum finish. Minimal alcohol presence. Actually, this is a nice quad. Feel: 4.5 Look: 4 Smell: 4.5 Taste: 4 Overall: 4

Aspects of opinions

Solving this problem depends on solving the following two sub-problems:

1. Labeling the sentences is **easy** if we have a good model of the words used to describe each aspect
2. Building a model of the different aspects is **easy** if we have labels for each sentence

- **Challenge:** each of these subproblems depends on having a good solution to the other one
- So (as usual) start the model somewhere and alternately solve the subproblems until convergence

Aspects of opinions

Model:

$$P(\text{aspect}(s) = k | \text{sentence } s, \text{rating } v) =$$

$$\frac{1}{Z} \exp \sum_{w \in s} \Big\{ \underbrace{\theta_{k,w}}_{\text{aspect weights}} + \underbrace{\phi_{k,v_k,w}}_{\text{sentiment weights}} \Big\}$$

normalization
over all aspects

Sum over words
in the sentence

Weight for a word
(w) appearing in a
particular aspect (k)

Weight for a word
(w) appearing in a
particular aspect
(k), when the rating
is v_k

Aspects of opinions

Intuition:

$$P(\text{aspect}(s) = k | \text{sentence } s, \text{rating } v) =$$

$$\frac{1}{Z} \exp \sum_{w \in s} \Big\{ \underbrace{\theta_{k,w}}_{\text{aspect weights}} + \underbrace{\phi_{k,v_k,w}}_{\text{sentiment weights}} \Big\}$$

Nouns should have high weights, since they describe an aspect but are independent of the sentiment

Adjectives should have high weights, since they describe specific sentiments

Aspects of opinions

Procedure:

1. Given the current model (theta and phi), choose the most likely aspect labels for each sentence

$$\max_{\text{aspect labels for each sentence}} P_{\theta,\phi}(\text{aspect}(s) = k | \text{sentence } s, \text{rating } v)$$

2. Given the current aspect labels, estimate the parameters theta and phi (convex problem)

$$\max_{\theta,\phi} P_{\theta,\phi}(\text{aspect}(s) = k | \text{sentence } s, \text{rating } v)$$

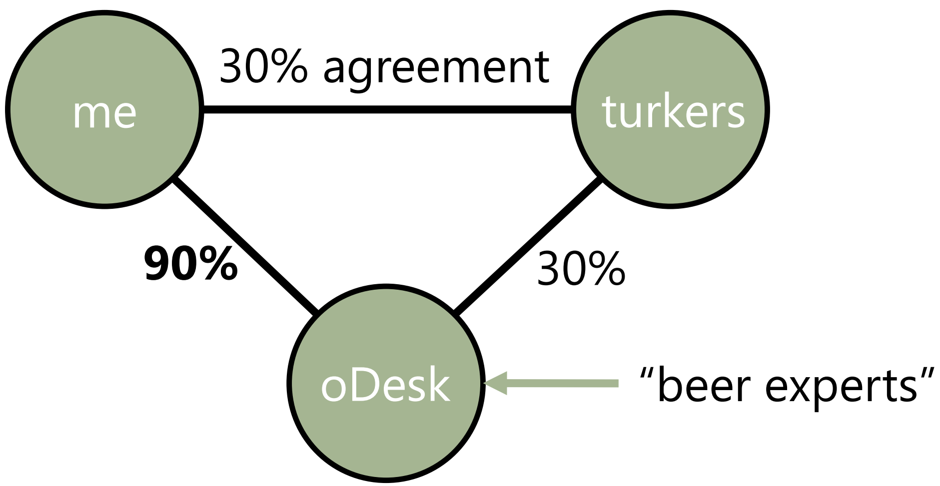
3. Iterate until convergence (i.e., until aspect labels don't change)

Aspects of opinions

Evaluation:

In order to tell if this is working, we need to get some humans to label some sentences

- I labeled 100 sentences for validation, and sent 10,000 sentences to Amazon's "mechanical turk"
  - These were next-to-useless
- So we hired some "experts" to label beer sentences



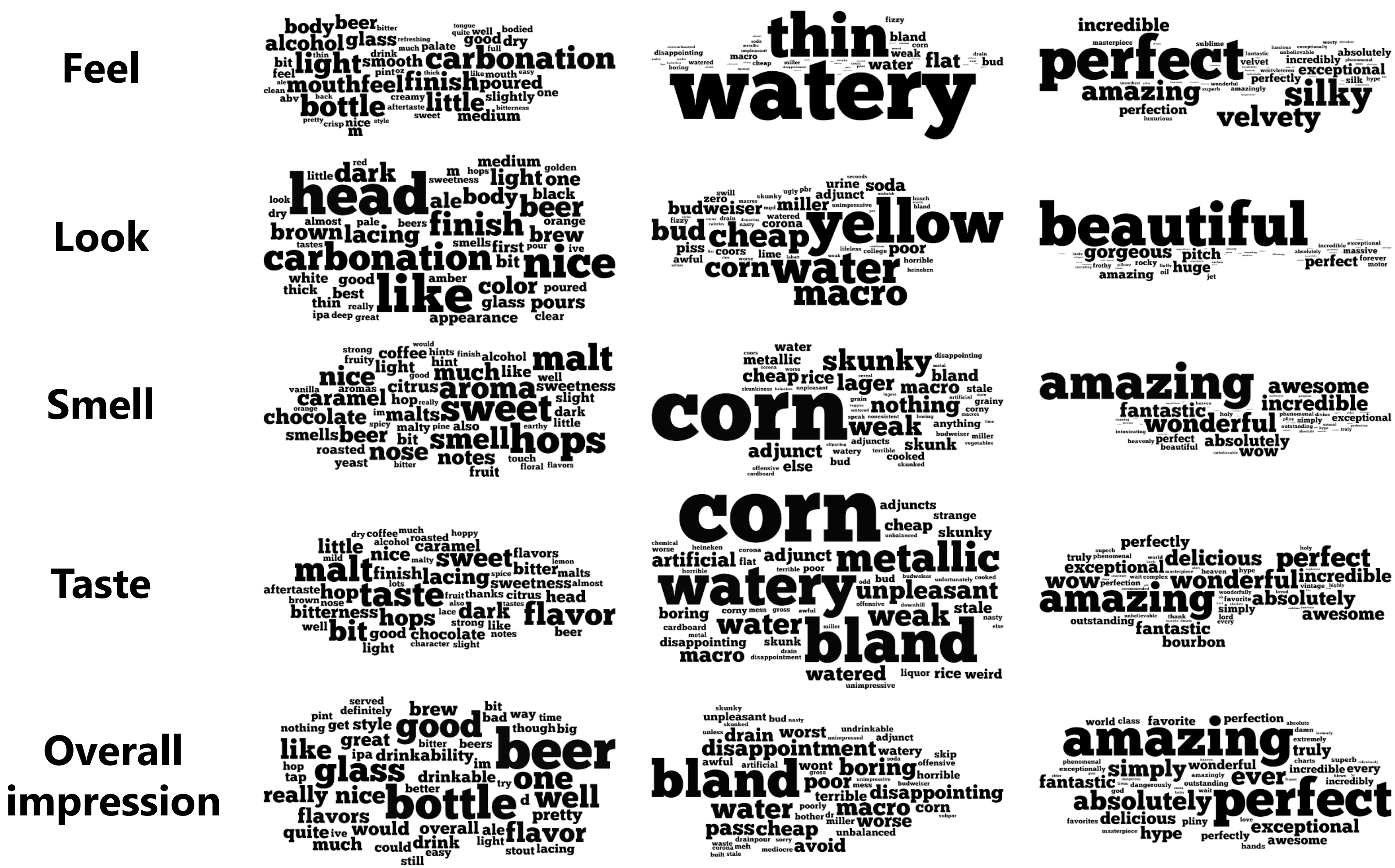
Aspects of opinions

Evaluation:

- 70-80% accurate at labeling beer sentences (somewhat less accurate for other review datasets)
- A few other tasks too, e.g. summarization (selecting sentences that describe different opinions on a particular aspect), and missing rating completion

Aspects of opinions

Aspect words θ_k | **Sentiment words (2-star)** $\phi_{k,2}$ | **Sentiment words (5-star)** $\phi_{k,5}$



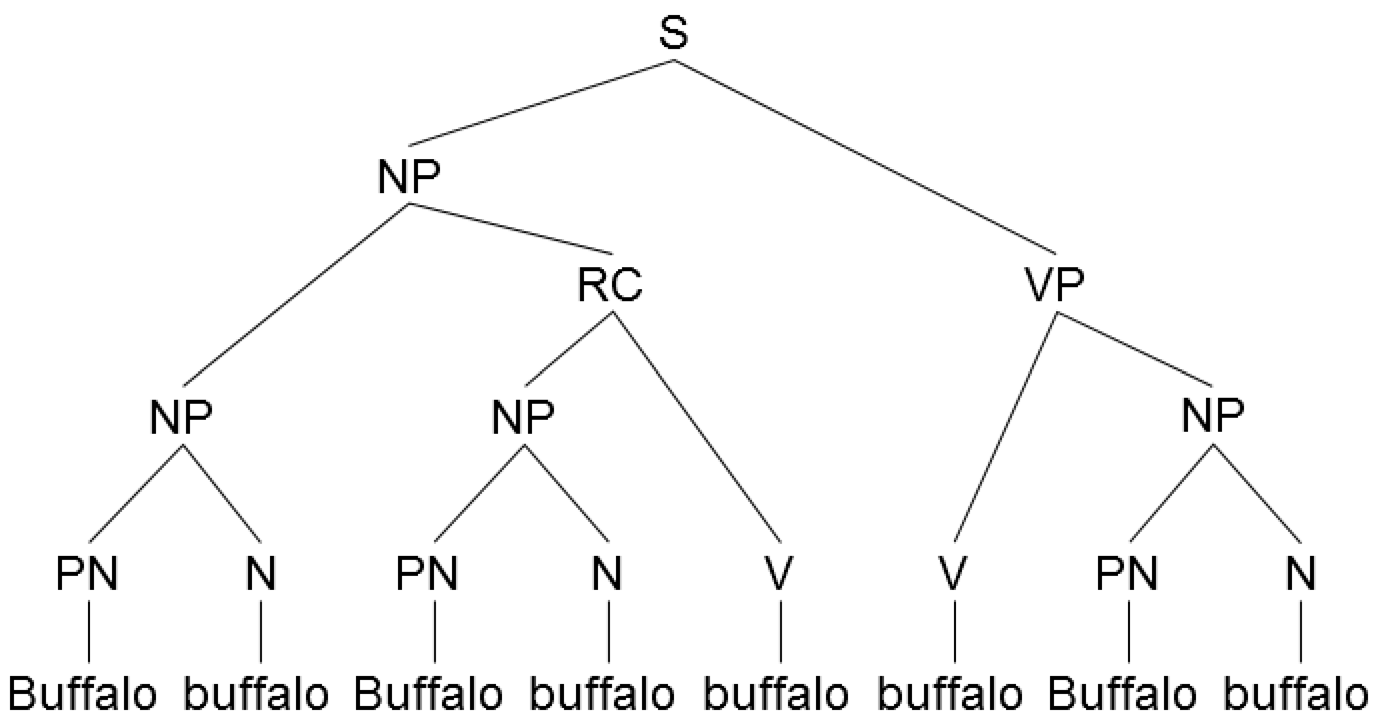
Aspects of opinions

Moral of the story:

- We can obtain fairly accurate results just using a bag-of-words approach
- People use very different language if the have positive vs. negative opinions
- In particular, people don't just take positive language and negate it, so modeling syntax (presumably?) wouldn't help that much

Aspects of opinions

Not today…



See Michael Collins & Regina Barzilay's NLP mooc if you're interested: http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-864-advanced-natural-language-processing-fall-2005/index.htm

Questions?

Further reading:

- Latent Dirichlet Allocation:

http://machinelearning.wustl.edu/mlpapers/paper_files/BleiNJ03.pdf

- Linguistics of food

"The language of Food: A Linguist Reads the Menu"

http://www.amazon.com/The-Language-Food-Linguist-Reads/dp/0393240835