Text Mining
### Assignment 1 – getting exciting!

<table>
<thead>
<tr>
<th>Competition</th>
<th>Teams/Entries</th>
<th>Quick Links</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CSE 255 (fa15) - Assignment 1 -- Task 2 -- Rating Prediction</strong></td>
<td>0 teams, 0 entries (0 this past week)</td>
<td>Leaderboard Forum, Edit in Wizard Invites</td>
</tr>
<tr>
<td><strong>CSE 190/255 (fa15) - Assignment 1 -- Task 1 -- Helpfulness Prediction</strong></td>
<td>0 teams, 0 entries (0 this past week)</td>
<td>Leaderboard Forum, Edit in Wizard Invites</td>
</tr>
<tr>
<td><strong>CSE 190 (fa15) - Assignment 1 -- Task 2 -- Purchase Prediction</strong></td>
<td>0 teams, 0 entries (0 this past week)</td>
<td>Leaderboard Forum, Edit in Wizard Invites</td>
</tr>
</tbody>
</table>

23 days to go
• Midterms will be **in class** next week
• We’ll do prep on Wednesday, and come back to the rest of text mining in week 7
• No class the Wednesday after the midterm
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.

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 rounds.
Apple Is Forming an Auto Team

By BRIAN X. CHEN and MIKE ISAAC   FEB 10, 2015

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.

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?

1. THE UPSHOT
   Reader Mailbag: Questions and Comments About Orders at Chipotle

2. Meet the Unlikely Airbnb Hosts of Japan

3. At Chipotle, How Many Calories Do People Really Eat?

4. OP-ED CONTRIBUTOR
   Reform the Condominium

5. Cupid’s Arrows Wound in ‘Wolf Hall,’ ‘Skylight,’ ‘An Octoroon’ and ‘Big Love’

6. THE UPSHOT
   The Upside of Waiting in Line
Prediction tasks involving text

Find me articles similar to this one

Meatloaf That Conquers the Mundane

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.

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; polpettone 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. Coated cheese and herbs are
Which of these reviews am I most likely to agree with or find helpful?
Prediction tasks involving text

Which of these sentences best summarizes people’s opinions?

Customer Reviews

- Easy to clean, beautiful color.
  - Howard R. Cohen

- I love my dutch oven, use it all the time.....so I bought one for my mother, and she is really enjoying it too!
  - Juli Scott

- Have made spaghetti sauce, beef stew, chicken stew, vegetable soup, pot roast.....all kinds of things.
  - J. L. Knox

See all 2,939 customer reviews »
‘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?
Bag-of-words models
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

\[ F_{\text{text}} = [150, 0, 0, 0, 0, 0, 0, \ldots, 0] \]
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/cse190/data/beer/beer_50000.json
(see course webpage, from week 1)

Code on:
http://jmcauley.ucsd.edu/cse190/code/week5.py
Q1: How many words are there?

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

print(len(wordCount))
```

36,225 - too many
2: What if we remove capitalization/punctuation?

```python
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)
```

19,426
3: What if we merge different inflections of words?

- drinks $\rightarrow$ drink
- drinking $\rightarrow$ drink
- drinker $\rightarrow$ drink
- argue $\rightarrow$ argu
- arguing $\rightarrow$ argu
- argues $\rightarrow$ argu
- arguing $\rightarrow$ argu
- argus $\rightarrow$ 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
Feature vectors from text

3: What if we merge different inflections of words?
The algorithm is (fairly) simple but depends on a huge number of rules

3: What if we merge different inflections of words?

```python
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)
```

14,847
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
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!
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:

\[ \text{rating} \sim \alpha + \sum_{w \in \text{text}} \text{count}(w) \cdot \theta_w \]

Code: [http://jmcauley.ucsd.edu/cse190/code/week5.py](http://jmcauley.ucsd.edu/cse190/code/week5.py)
What do the parameters look like?

\[ \theta_{\text{fantastic}} = 0.143 \]

\[ \theta_{\text{watery}} = -0.163 \]

\[ \theta_{\text{and}} = -0.008 \]

\[ \theta_{\text{me}} = -0.037 \]
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:

```python
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']
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!
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.
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)
We can build a richer predictor by using \textit{n-grams}

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

\[ f(\text{text}) \rightarrow \text{class label} \]

(or even \( f(\text{text}) \rightarrow \{1 \text{ star, 2 star, 3 star, 4 star, 5 star}\} \))

using a probabilistic classifier:

\[ p(\text{class} = c | \text{text}) \]
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:

$$p(\text{label} | x) = \sigma(\langle \phi(x), \theta \rangle) p(\neg \text{label} | x) = \frac{e^{-\langle \phi(x), \theta \rangle}}{1 + e^{-\langle \phi(x), \theta \rangle}}$$
Recall: multinomial distributions

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

\[ p(\text{class} = c|x) = \frac{1}{Z} \exp \langle \theta_c, x \rangle = \frac{e^{\langle \theta_c, x \rangle}}{\sum_c e^{\langle \theta_c, x \rangle}} \]

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?

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

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

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:** $\theta_{5\text{-star},'great'}, \theta_{5\text{-star},'fantastic'}, \theta_{1\text{-star},'terrible'}$

**Low weights:** $\theta_{1\text{-star},'great'}, \theta_{1\text{-star},'fantastic'}, \theta_{5\text{-star},'terrible'}$
So far...

Bags-of-words representations of text

- Stemming & stopwords
- Unigrams & N-grams
- Sentiment analysis & text classification
Further reading:

• Original stemming paper
  “Development of a stemming algorithm” (Lovins, 1968):

• Porter’s paper on stemming
  “An algorithm for suffix stripping” (Porter, 1980):
CSE 190 – Lecture 9
Data Mining and Predictive Analytics

Case study: inferring aspects from multi-dimensional reviews
A (very quick) case study

(I know it’s not that part of the lecture yet)

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:

‘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

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
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:

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

\[ \frac{1}{Z} \exp \sum_{w \in s} \left\{ \theta_{k,w} + \phi_{k,v_k,w} \right\} \]

- 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:

\[
P(\text{aspect}(s) = k | \text{sentence } s, \text{ rating } v) = \frac{1}{Z} \exp \sum_{w \in s} \left\{ \theta_{k,w} + \phi_{k,v_k,w} \right\}
\]

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)
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

30% agreement
90%
30%

me

me

me

me

me

me

me

me

me

me

me

me

me

me

me
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 $\theta_k$
Feel
- body
- beery
- clear
- smooth
- carbonation
- bottle
- little
- medium

Look
- head
- finish
- nice
- like
- dark
- lacing
- beer
- brown
- carbonation
- lacing
- bright
- water
- yellow
- macro

Smell
- nice
- caramel
-注册
- chocolate
- malt
- hops
- nose
- notes
- fruit

Taste
- nice
- malt
- sweetness
- finish
- aftertaste
- bitterness
- dark
- flavor
- good
- chocolate
- beer
- water
- bland
- corn
- water
- macro

Overall
impression
- glass
- one
- bottle
- good
- well
- drain
- worst
- disappointment
- water
- water
- bland
- beer
- more
- avoid
- amazing
- perfect
- exceptional
- perfectly
- kill
- great
- perfect
Moral of the story:

• We can obtain fairly accurate results just using a bag-of-words approach
• People use very different language if they 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...

Further reading:

- Linguistics of food
  “The language of Food: A Linguist Reads the Menu”
  http://www.amazon.com/The-Language-Food-Linguist-Reads/dp/0393240835