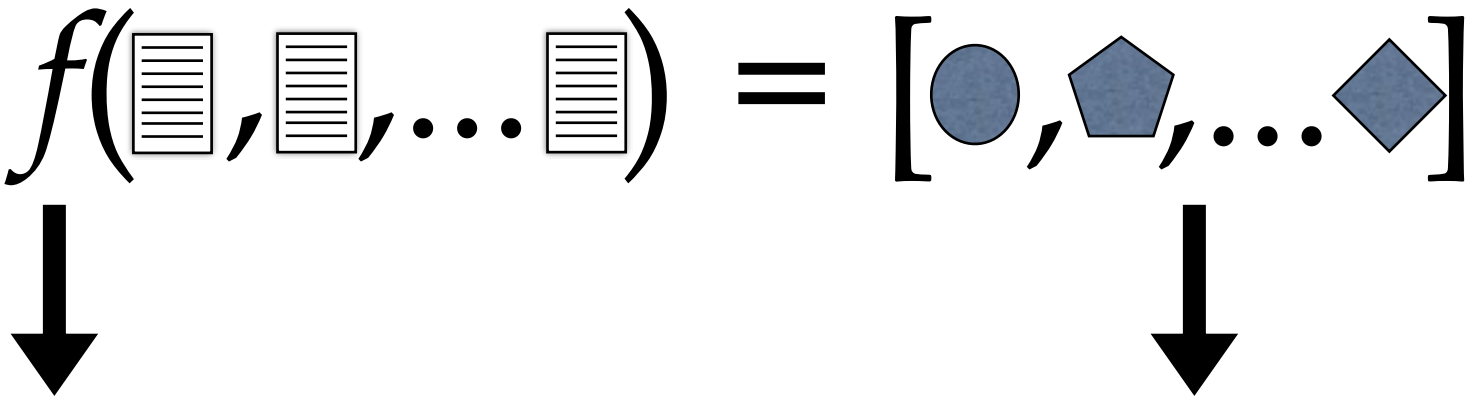
# Text classification

LIN 313: Language and Computers

Jason Baldridge
The University of Texas at Austin
(Many slides borrowed from Lillian Lee)

# Text analysis, in brief



Rules
Annotation & Learning
- annotated examples
- annotated knowledge
- interactive annotation and learning

Scalable human annotation

Sentiment labels
Parts-of-speech
Named Entities
Topic assignments
Geo-coordinates
Syntactic structures
Translations

# Document classification: automatically label some text

- **Language identification**: determine the language that a text is written in
- **Spam filtering**: label emails, tweets, blog comments as spam (undesired) or ham (desired)
- **Routing**: label emails to an organization based on which department should respond to them (e.g. complaints, tech support, order status)
- **Sentiment analysis**: label some text as being positive or negative (polarity classification)
- **Georeferencing**: identify the location (latitude and longitude) associated with a text

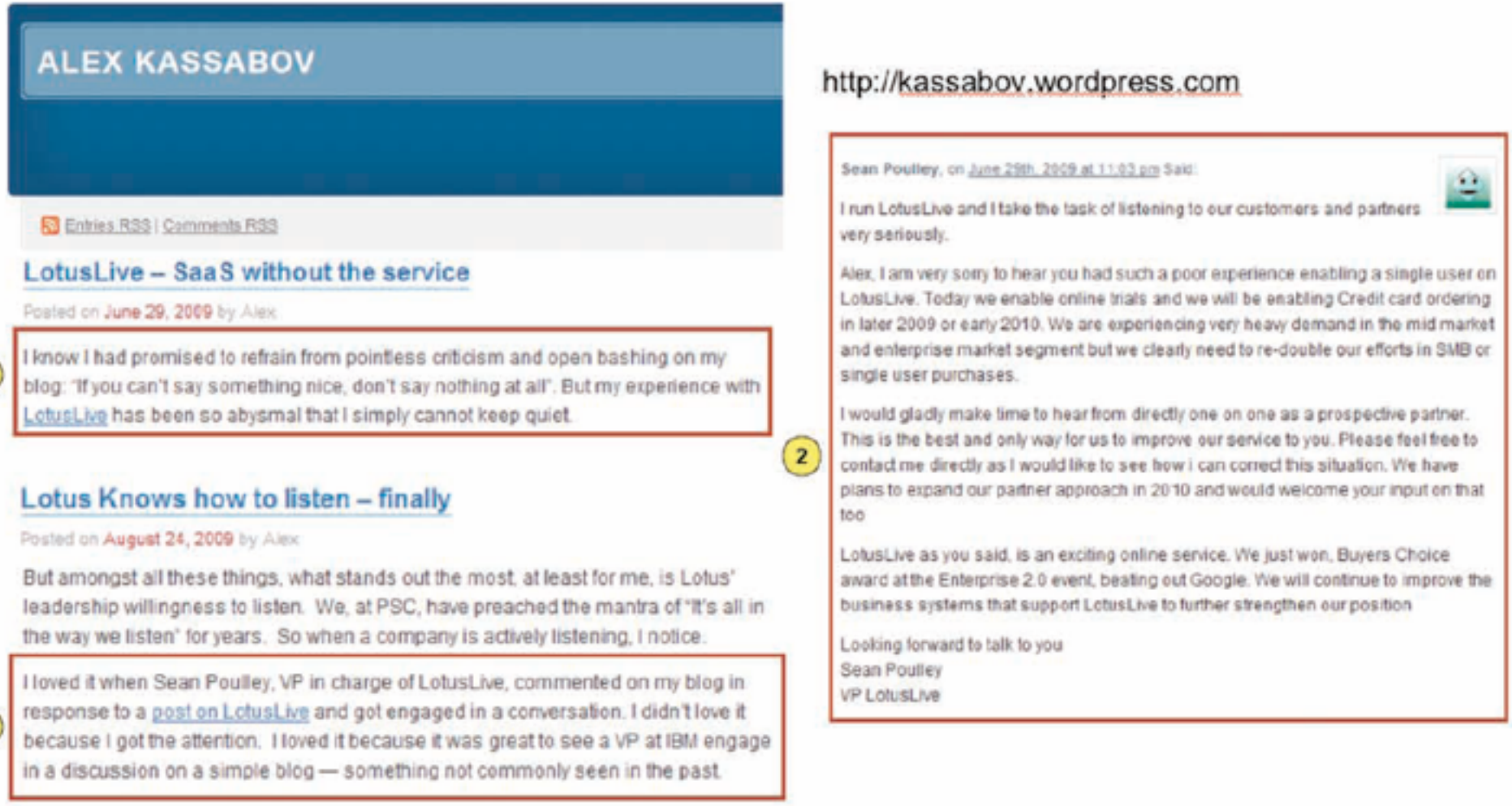
# Sentiment analysis: background [slide from Lillian Lee]

- People search for and **are affected by** online opinions.
  - TripAdvisor, Rotten Tomatoes, Yelp, Amazon, eBay, YouTube, blogs, Q&A and discussion sites
- According to a Comscore ’07 report and an ’08 Pew survey:
  - 60% of US residents have done online product research, and 15% do so on a typical day.
  - 73%-87% of US readers of online reviews of services say the reviews were significant influences. (more on economics later)
- But, 58% of US internet users report that online information was missing, impossible to find, confusing, and/or overwhelming.
- Creating technologies that find and analyze reviews would answer a tremendous information need.

# Broader implications: economics [slide from Lillian Lee]

- Consumers *report* being willing to pay from 20% to 99% more for a 5-star-rated item than a 4-star-rated item. [comScore]
- But, does the polarity and/or volume of reviews have measurable, significant influence on actual consumer purchasing?
  - Implications for bang-for-the-buck, manipulation, etc.
- Sample quote (much debate in the literature):
  - ...on average, 3.46 percent of [eBay] sales is attributable to the seller's positive reputation stock. ... the average cost to sellers stemming from neutral or negative reputation scores is $2.28, or 0.93 percent of the final sales price. **If these percentages are applied to all of eBay's auctions [$1.6 billion in 2000 4Q]**, ... sellers' positive reputations added more than $55 million to ... sales, while non-positives reduced sales by about $15 million. [Houser and Wooders '06]

# Social media analytics: acting on sentiment




Richard Lawrence, Prem Melville, Claudia Perlich, Vikas Sindhwani, Estepan Meliksetian et al.
In *ORMS Today*, Volume 37, Number 1, February, 2010.

# Beyond consumption: politics [slide from Lillian Lee]

- In 2006, 31% of US residents used the internet for gathering or sharing political information (60M+ people).
- Major reason?
  - 28%: to get perspectives from within their community.
  - 34%: to get perspectives from outside it.
- The kind of sites they visit?
  - 28% said that most sites they use share their point of view.
  - 29% said that most challenge their point of view.
- From Rainie and Horrigan Pew survey, ’07

# Beyond individual interest [slide from Lillian Lee]

- Business intelligence systems could ...
  - search out, analyze, and summarize opinionated mentions of products, features, consumer desires, etc.
  - automatically process customer feedback
- Governmental eRulemaking initiatives (e.g., www.regulations.gov) directly solicit citizen comments on potential new rules
  - 400,000 received for a single rule on labeling organic food
- Many other applications exist, as well.

# Polarity classification [slide from Lillian Lee]

- Consider just classifying an avowedly subjective text unit as either positive or negative ("thumbs up or "thumbs down").
- One application: review summarization.
  - Elvis Mitchell, May 12, 2000: *It may be a bit early to make such judgments, but Battlefield Earth may well turn out to be the worst movie of this century*.
- Can't we just look for words like "great", "terrible", "worst"?
- Yes, but ... learning a sufficient set of such words or phrases is an active challenge.

# Using a lexicon [slide from Lillian Lee]

- From a small scale human study:

| | Proposed word lists | Accuracy |
|---|---|---|
| Subject 1 | Positive: dazzling, brilliant, phenomenal, excellent, fantastic<br>Negative: suck, terrible, awful, unwatchable, hideous | 58% |
| Subject 2 | Positive: gripping, mesmerizing, riveting, spectacular, cool, awesome, thrilling, badass, excellent, moving, exciting<br>Negative: bad, cliched, sucks, boring, stupid, slow | 64% |
| Automatically determined (from data) | Positive: love, wonderful, best, great, superb, beautiful, still<br>Negative: bad, worst, stupid, waste, boring, ?, ! | 69% |

# Polarity words are not enough [slide from Lillian Lee]

- Can't we just look for words like "great" or "terrible"?
- Yes, but ...
  - This laptop is a great deal.
  - A great deal of media attention surrounded the release of the new laptop.
  - This laptop is a great deal ... and I've got a nice bridge you might be interested in.

# Polarity words are not enough

- **Polarity flippers**: some words change positive expressions into negative ones and vice versa.
  - **Negation**: *America still needs to be focused on job creation. **Not** among Obama's **great** accomplishments since coming to office !!* [From a tweet in 2010]
  - **Contrastive discourse connectives**: *I used to **HATE** it. **But** this stuff is yummmmmy :)* [From a tweet in 2011 -- the tweeter had already bolded "HATE" and "But"!]
- **Multiword expressions**: other words in context can make a negative word positive:
  - That movie was shit. [negative]
  - That movie was the shit. [positive] (American slang from the 1990's)

# More subtle sentiment (from Pang and Lee)

- With many texts, no ostensibly negative words occur, yet they indicate strong negative polarity.
  - *"If you are reading this because it is your darling fragrance, please wear it at home exclusively, and tape the windows shut.*" (review by Luca Turin and Tania Sanchez of the Givenchy perfume Amarige, in *Perfumes: The Guide*, Viking 2008.)
  - "*She runs the gamut of emotions from A to B.*" (Dorothy Parker, speaking about Katharine Hepburn.)
  - *"Jane Austen's books madden me so that I can't conceal my frenzy from the reader. Every time I read 'Pride and Prejudice' I want to dig her up and beat her over the skull with her own shin-bone.*" (Mark Twain.)

# Thwarted expectations (from Pang and Lee)

- There are also highly negative texts that use lots of positive words, but ultimately are reversed by the final sentence. For example

  *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.*

- This is referred to as a **thwarted expectations narrative** because in the final sentence the author sets up a deliberate contrast to the preceding discourse, giving it more impact.

# Polarity classification

- It's not just positive or negative!
- Examples [http://www.edmunds.com/ford/focus/review.html]:
  - **Positive**: "As a used vehicle, the Ford Focus represents a solid pick."
  - **Negative**: "Still, the Focus' interior doesn't quite measure up to those offered by some of its competitors, both in terms of materials quality and design aesthetic."
  - **Neutral**: "The Ford Focus has been Ford's entry-level car since the start of the new millennium."
  - **Mixed**: "The current Focus has much to offer in the area of value, if not refinement."

# Other dimensions of sentiment analysis

- **Subjectivity**: is an opinion even being expressed? Many statements are simply factual.
- **Target**: what exactly is an opinion being expressed about?
  - Important for aggregating interesting and meaningful statistics about sentiment.
  - Also, it affects how the language use indicates polarity: e.g, *unpredictable* is usually positive for movie reviews, but is very negative for a car's steering
- **Ratings**: rather than a binary decision, it is often of interest to provide or interpret predictions about sentiment on a scale, such as a 5-star system.

# Other dimensions of sentiment analysis

- **Perspective**: an opinion can be positive or negative depending on who is saying it
  - *entry-level* could be good or bad for different people
  - it also affects how an author describes a topic: e.g. *pro-choice* vs *pro-life*, *affordable health care* vs *obamacare*.
- **Authority:** was the text written by someone whose opinion matters more than others?
  - it is more important to identify and address negative sentiment expressed by a popular blogger than a one-off commenter or supplier of a product reviewer on a sales site
  - follower graphs (where applicable) are very useful in this regard
- **Spam**: is the text even valid or at least something of interest?
  - many tweets and blog post comments are just spammers trying to drive traffic to their sites

# Rule-based classification

- Identify words and patterns that are indicative of positive or negative sentiment:
  - *polarity words*: e.g. good, great, love; bad, terrible, hate
  - *polarity ngrams*: the shit (+), must buy (+), could care less (-)
  - *casing*: uppercase often indicates subjectivity
  - *punctuation*: lots of ! and ? indicates subjectivity (often negative)
  - *emoticons*: smiles like :) are generally positive, while frowns like :( are generally negative
- Use each pattern as a rule; if present in the text, the rule indicates whether the text is positive or negative.
- How to deal with conflicts? (E.g. multiple rules apply, but indicate both positive and negative?)

# Rule combination

- First try: order the rules according to their accuracy.
  - What is problematic with this?
- Second try: assign weights to the rules.
  - What is problematic with this?

# Problems with rule-based filters

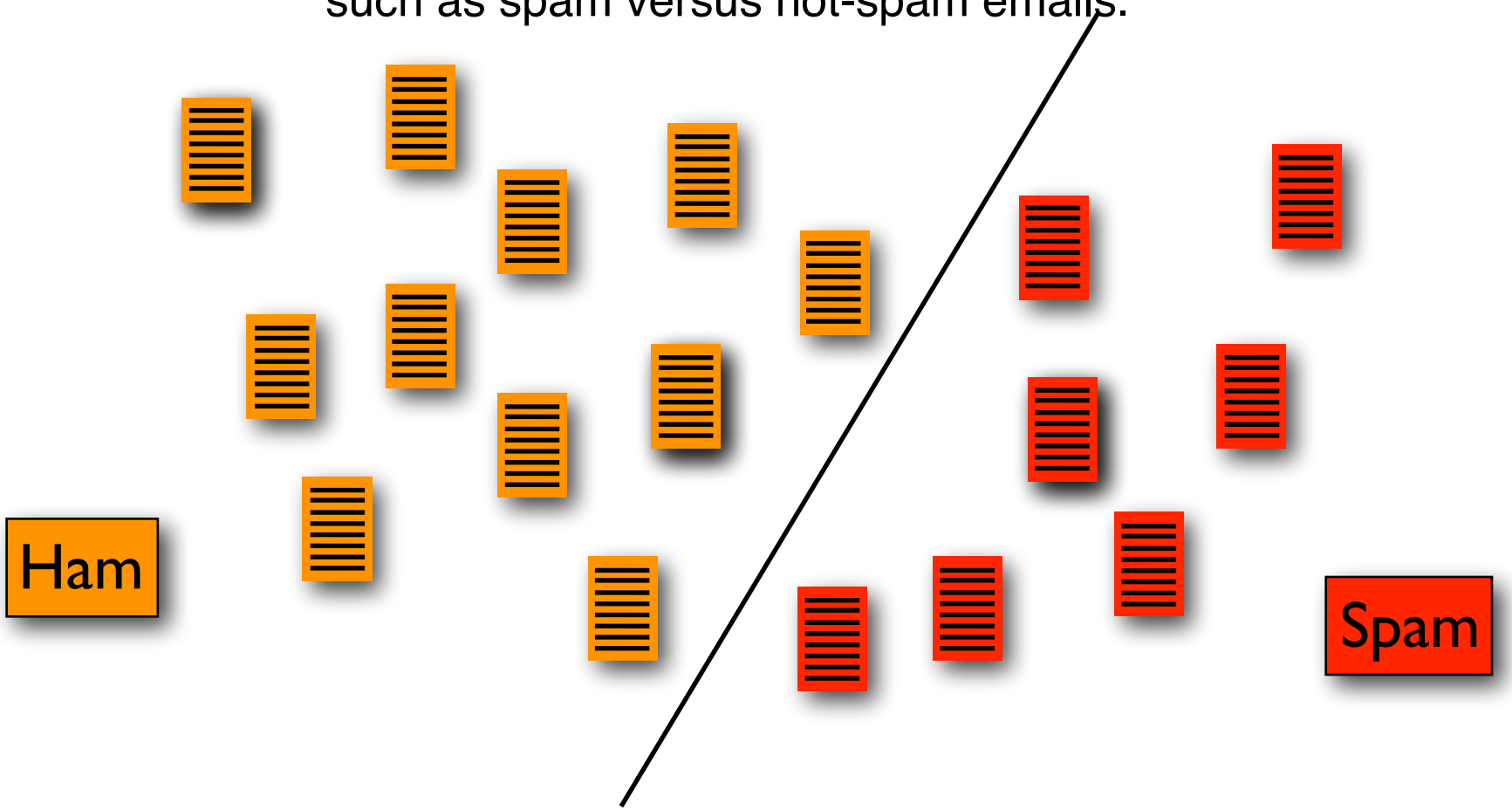
- Someone has to come up with the rules: this can take a long time and lots of effort.
  - This will probably result in high precision, but low recall. (More on these later.)
- The rules are designed for every dataset, but they don't always work universally.
  - E.g.: the word *unpredictable* is good for movies, bad for cars.
- Rules are designed and then deployed, so they don't evolve over time to adapt to changes in word use.
  - New expressions come into use all the time: e.g. "badass" and "bad ass" weren't used much until the last few years

# Machine learning for classification

- The rule-based approach requires defining a set of *ad hoc* rules and explicitly managing their interaction.
- If we instead have lots of examples of texts of different categories, we can learn a function that maps new texts to one category or the other.
  - These are often probabilistic, but need not be.
  - What were rules become features that are extracted from the input; their importance is extracted from statistics in a labeled training set.
  - These features are dimensions; their values for a given text plot it into space, just as we did with authorship attribution.

# Machine learning for classification

- Idea: software learns from examples it has seen.
- Find the boundary between different classes of things, such as spam versus not-spam emails.

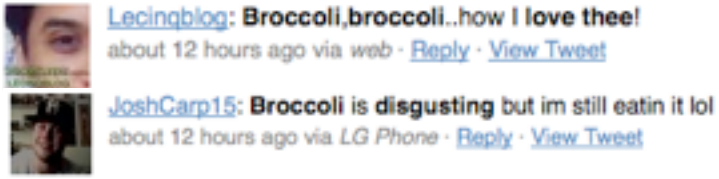


# Clustering versus classification

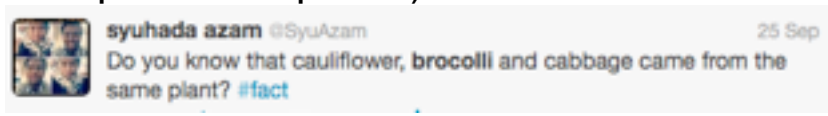
- As with k-means for finding clusters of authorship styles, we quantify features of the texts and are thus able to plot them into some space.
  - With N features, we have an N-dimensional space.
- Unlike k-means, we have a label associated with each text and can use such labels to directly model the classification task. This is **supervised machine learning**.
- Another way of saying this is that some reasonably knowledgeable human clustered the documents into categories that are meaningful.
- A classifier seeks to model these predefined clusters in a way that generalizes well to new documents, allowing them to be accurately labeled automatically.

# Example scenario: Twitter sentiment classification

- You label all the tweets about broccoli on a given day.
  - There are 5000 tweets about broccoli.
  - 1500 of them are subjective (express an opinion)
    - 500 of these are positive (pos)
    - 1000 of these are negative (neg)




  - The other 3500 are objective (don't express an opinion)




- The word "hate" appears in 203 tweets
  - 200 of these are negative
  - 3 of these are positive.

# How likely are positive or negative tweets?

- We can determine several interesting probabilities straight away.
- Two of them are the probability of positive or negative tweets.
  - P(neg): the percentage of tweets which are negative
  - P(pos): the percentage of tweets which are positive
- These values are easy to calculate:
  - $P(\text{neg}) = \frac{\text{number of neg tweets}}{\text{number of tweets}} = 1000/1500 = 2/3 = .667$
  - $P(\text{pos}) = \frac{\text{number of pos tweets}}{\text{number of tweets}} = 500/1500 = 1/3 = .333$

# How likely is a word within positive or negative tweets?

- Other interesting probabilities are those of seeing the word "hate" in positive or negative tweets.
- We write these as
  - P(hate | neg): the percentage of neg tweets that contain the word "hate"
  - P(hate | pos): the percentage of pos tweets that contain "hate"
- These values can be directly estimated from the data:
  - $P(\text{hate}|\text{neg}) = \frac{\text{number of neg tweets with "hate"}}{\text{number of neg tweets}} = 200/1000 = .20$
  - $P(\text{hate}|\text{pos}) = \frac{\text{number of pos tweets with "hate"}}{\text{number of pos tweets}} = 3/500 = .006$

# Inverting conditioning: Bayes law

- But surely what we are most interested in is having a model which tells or how probable it is that a tweet is negative given that it contains the word *hate*...
  - We write this as P(neg | hate).
- Bayes law helps us out:

$$P(B|A) = \frac{P(A|B) \times P(B)}{P(A)}$$

- For our present example, this means we are looking at:

$$P(\text{neg} \mid \text{hate}) = \frac{P(\text{hate} \mid \text{neg}) \times P(\text{neg})}{P(\text{hate})}$$

- So, this gives us what we want, and we've already found P(hatelneg) and P(neg), leaving just P(hate) to be determined.

# How likely is a word overall?

- P(hate) is the probability of seeing the word "hate" in any tweet. Given our current scenario, we could compute this directly as:
  - $P(\text{hate}) = \frac{\text{number of tweets with "hate"}}{\text{number of tweets}} = 203/1500 = .135$
- Alternatively, we can compute it as the sum of the probability of *hate* in each type of tweet:
  - P(hate) = P(hate | neg) x P(neg) + P(hate | pos) x P(pos)
- Why do it this way? We'll see in a moment, but first let's start putting things together, and see if we obtain the value P(hate) = .135 when computed this way.

# How likely is a word overall? (2)

- What we know:
  - P(neg) = .667
  - P(hate | neg) = .20
  - P(pos) = .333
  - P(hate | pos) = .006
- So: P(hate | neg) x P(neg) = .2 x .667 = .133
- And: P(hate | pos) x P(pos) = .006 x .333 = .002
- Which means that:
  - P(hate) = P(hate|neg) x P(neg) + P(hate|pos) x P(pos)= .133 + .002 = .135

# Putting it all together

- Now we have all the components we need to calculate P(neg | hate):

$$P(neg|hate) = \frac{P(hate|neg) \times P(neg)}{P(hate)}$$

$$= \frac{P(hate|neg) \times P(neg)}{P(hate|neg) \times P(neg) + P(hate|pos) \times P(pos)}$$

$$= \frac{.2 \times .667}{.2 \times .667 + .006 \times .333} = \frac{.133}{.133 + .002} = .985$$

- So, based on the evidence of the word "hate" alone, the model thinks the a tweet with the word "hate" is 98.5% likely to be negative.

# Why did we go through all that work?

- Hang on... couldn't we have just directly calculated P(neg|hate)??!!
  - P(neg|hate) = $\frac{\text{number of neg tweets with "hate"}}{\text{number of tweets with "hate"}}$ = 200/203 = .985
- There are many deep and interesting reasons. For now, we'll consider just two:
  - **modularity**: use different models, or even guesses, for estimating the probabilities
  - **sparsity**: we'd like to use multiple words per tweet to determine whether its polarity, and we won't get sufficient counts using the above calculation.
- We'll consider both of these briefly.

# Modularity

- By estimating P(neg I hate) with Bayes law rather than direct calculation, we now have two different probability distributions of interest: P(hate I neg) and P(neg)
- P(neg) is called the **class prior.**
  - Think of it this way: prior to even seeing the words in an tweet, would I be likely to think of it as more or less likely to be negative?
- We estimated P(neg) from our training set. But, what if I have only a few tweets to train on? Perhaps then it would be better to use P(neg) based on some other source, like a poll about attitudes toward broccoli.
- This modularity allows us to combine a general P(neg) estimate with an estimate of P(hate I neg) from our data.

# Modularity, using a different prior

- Consider using a prior where P(neg) = .35 (rather than the value .667 from our data).
- So, P(pos) = 1-P(neg) = 1 - .35 = .65

$$P(\text{neg}|\text{hate}) = \frac{P(\text{hate}|\text{neg}) \times P(\text{neg})}{P(\text{hate}|\text{neg}) \times P(\text{neg}) + P(\text{hate}|\text{pos}) \times P(\text{pos})}$$

$$= \frac{.2 \times .35}{.2 \times .35 + .006 \times .65} = \frac{.07}{.07 + .004} = .946$$

- So, it still looks pretty negative, but this model doesn't think it is as negative as the previous one, which gave the value P(neg I hate) = .985.

# Sparsity

- We've been considering a single word example, but usually we want to use many or even most of the words in a tweet to determine its polarity.
- This means calculating things like:
  - P(neg | "hate", "disgusting", "broccoli", "eat", "aardvark"...).
  - If we want to determine the polarity directly, we need to count the number of tweets that contain **exactly** those words, in both the positive and negative groups.
- There would be very few tweets that contain the same set of words, even in a very large corpus! So, we wouldn't be able to get reliable counts.

$$P(\text{neg}|\text{hate, disgusting,..., aardvark}) = \frac{\text{number of neg tweets with "hate", "disgusting", ... "aardvark",..}}{\text{number of tweets with "hate", "disgusting", ... "aardvark",..}}$$

We might see just one message with these words = 1/1 = 100%!

# Sparsity: how Bayes law helps

- Bayes law inverts the conditioning:

$$P(\text{neg}|\text{hate, disgusting,..., aardvark}) = \frac{P(\text{hate, disgusting,..., aardvark} \mid \text{neg}) \times P(\text{neg})}{P(\text{hate, disgusting,..., aardvark})}$$

- We may then assume independence between the words in order to simplify the computation of P(hate...aardvark I neg):

$$P(\text{hate, disgusting,..., aardvark} \mid \text{neg}) = P(\text{hate} \mid \text{neg}) \times P(\text{disgusting} \mid \text{neg}) \times .... \times P(\text{aardvark} \mid \text{neg})$$

- Bayes law plus the independence assumption allows us to break up the big calculation into a bunch of terms, each of which we can get reliable evidence (counts) for, as we did for P(hate I neg) before.

# Precision and Recall in sentiment analysis

- Very relevant for sentiment analysis because there are subjective and objective texts, and we only wish to assign polarity values (positive, negative) to subjective texts.
  - If we assign polarity to objective texts, it is by definition incorrect.
  - It will also produce poorer aggregate estimates of the relative positivity/negativity toward the target, because it is based on items that should be out of consideration.
- This means that we will usually perform subjectivity classification on a set of texts before doing polarity classification.
  - E.g. we started with 5,000, tweets about broccoli -- if a subjectivity classifier identifies 3,112 of them as objective, then we will only assign polarity to the remaining 1,888 (which the classifier thinks are subjective).
  - Errors: some of the tweets that the classifier said were objective could be subjective, and vice versa. In fact, for the Twitter broccoli dataset, *at least* 388 objective tweets were incorrectly identified as subjective (1888 identified as subjective - 1500 actually subjective tweets = 388 classified-as-subjective tweets too many.
  - Of course, probably some subjective tweets were classified as objective, leading to more errors. We'll measure the error with precision and recall on subjective and objective tweets.

# Precision and Recall: subjectivity classification

- We are interested in knowing the precision of a subjectivity classifier: out of all the tweets it thinks are subjective, how many actually are subjective?
- We'd also like to measure its recall: out of all the tweets that are in fact subjective, how many did it actually identify?
- We can ask the same questions about objective tweets.

# P/R for detecting subjective tweets

- Let's first consider the classifier's performance for identifying subjective tweets.
- Recall: there are 5000 broccoli tweets. The classifier identifies 1888 as subjective and 3112 (=5000-1888) as objective. Of the 1888, only 1355 are in fact subjective. Of the 3112, 145 are subjective (mistakenly identified as objective).

True Positives

False Negatives

| | Classifier:Subj | Classifier:Obj | |
|---|---|---|---|
| Truth:Subj | 1355 | 145 | 1500 |
| Truth:Obj | 533 | 2967 | 3500 |
| | 1888 | 3112 | 5000 |

True Negatives

False Positives

# Definition of precision and recall

- Precision: of how many you guessed, how many were correct?

$$\frac{\text{True Positives}}{\text{True Positives + False Positives}}$$

- Recall: of how many you should have found, how many did you identify?

$$\frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

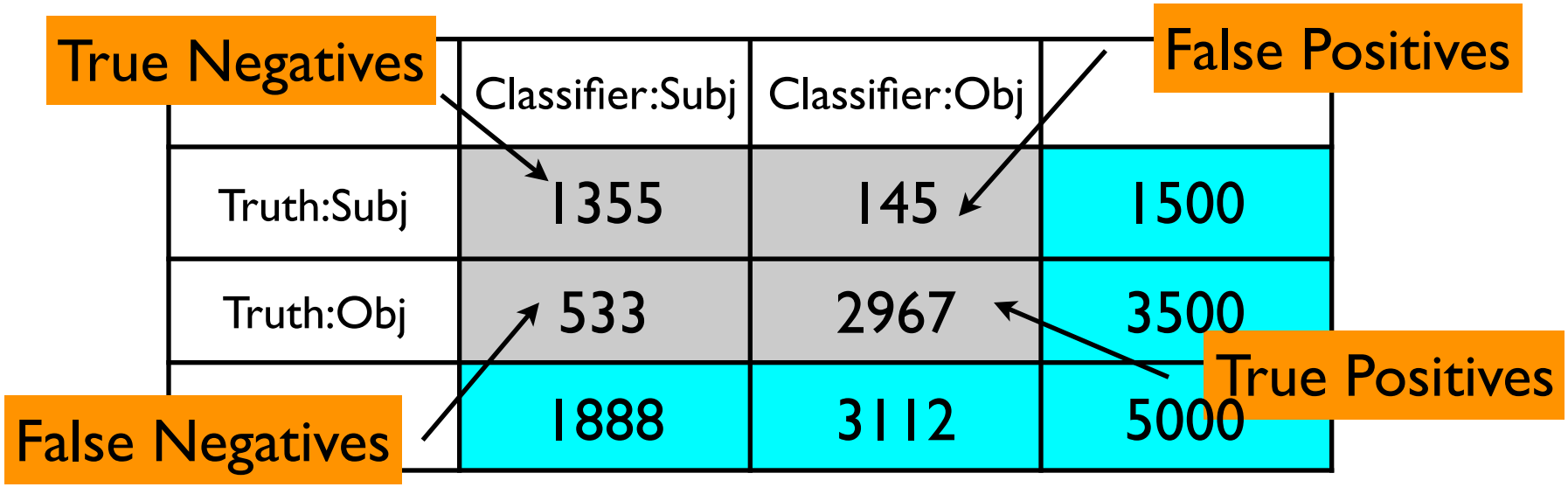
# Back to our subjectivity classification

| | Classifier:Subj | Classifier:Obj | |
|---|---|---|---|
| Truth:Subj | 1355 | 145 | 1500 |
| Truth:Obj | 533 | 2967 | 3500 |
| | 1888 | 3112 | 5000 |

- Precision = TP/(TP+FP) = 1355/(1355+533) = 1355/1888 = .7177 = 71.77%
- Recall = TP/(TP+FN) = 1355/(1355+145) = 1355/1500 = .9033 = 90.33%

# Precision and recall are relative

- What if we were interested in the classifier's performance in identifying objective tweets?
- Our positives and negatives are now defined differently.



# P/R for identifying objective tweets

| | Classifier:Subj | Classifier:Obj | |
|---|---|---|---|
| Truth:Subj | 1355 | 145 | 1500 |
| Truth:Obj | 533 | 2967 | 3500 |
| | 1888 | 3112 | 5000 |

- Now, TP=2967, FP=145, FN=533, and TN=1355
- Precision = TP/(TP+FP) = 2967/(2967+145) = 2967/3112 = .9534 = 95.34%
- Recall = TP/(TP+FN) = 2967/(2967+533) = 2967/3500 = .8477 = 84.77%

# Four sentiment datasets

| Dataset | Topic | Year | # Train | # Dev | #Test | Reference |
|---|---|---|---|---|---|---|
| Debate08 | Obama vs McCain debate | 2008 | 795 | 795 | 795 | Shamma, et al. (2009) "Tweet the Debates: Understanding Community Annotation of Uncollected Sources." |
| HCR | Health care reform | 2010 | 839 | 838 | 839 | Speriosu et al. (2011) "Twitter Polarity Classification with Label Propagation over Lexical Links and the Follower Graph." |
| STS | (Stanford) Twitter Sentiment | 2009 | - | 216 | - | Go et al. (2009) "Twitter sentiment classification using distant supervision" |
| IMDB | IMDB movie reviews | 2011 | 25,000 | 25,000 | - | Mas et al. (2011) "Learning Word Vectors for Sentiment Analysis" |

# Rule-based classification

- Identify words and patterns that are indicative of positive or negative sentiment:
  - *polarity words*: e.g. good, great, love; bad, terrible, hate
  - *polarity ngrams*: the shit (+), must buy (+), could care less (-)
  - *casing*: uppercase often indicates subjectivity
  - *punctuation*: lots of ! and ? indicates subjectivity (often negative)
  - *emoticons*: smiles like :) are generally positive, while frowns like :( are generally negative
- Use each pattern as a rule; if present in the text, the rule indicates whether the text is positive or negative.
- How to deal with conflicts? (E.g. multiple rules apply, but indicate both positive and negative?)
  - Simple: count number of matching rules and take the max.

# Simplest polarity classifier ever?

```
def polarity(document) =
  if (document contains "good")
     positive
  else if (document contains "bad")
     negative
  else
     neutral
```

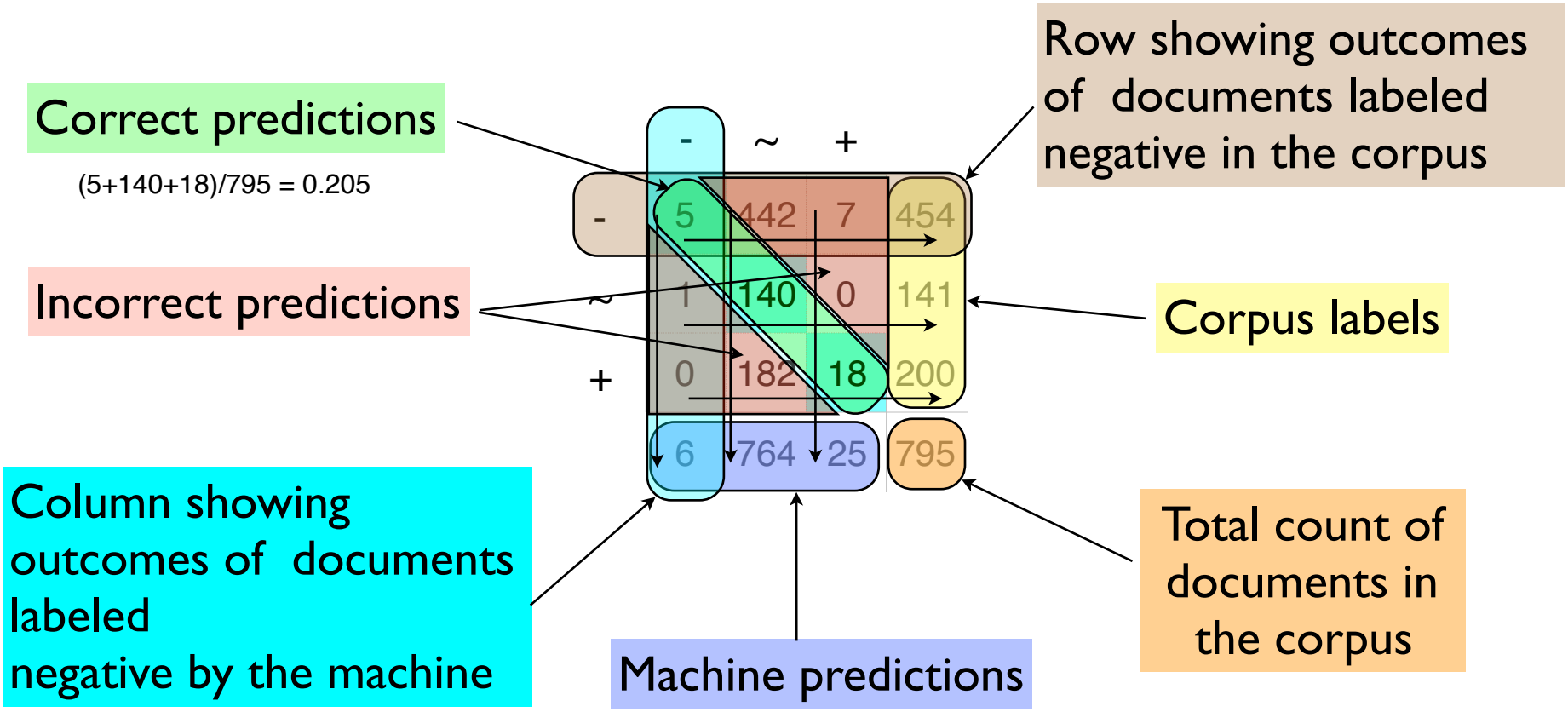
| Debate08 | HCR | STS | IMDB |
|---|---|---|---|
| 20.5 | 21.6 | 19.4 | 27.4 |

No better than flipping a (three-way) coin?

Code and data here: https://github.com/utcompling/sastut

# The confusion matrix

- We need to look at the confusion matrix and breakdowns for each label. For example, here it is for Debate08:




+ is positive, - is negative, ~ is neutral

# Precision, Recall, and F-score: per category scores

- Precision: the number of correct guesses (true positives) for the category divided by all guesses for it (true positives and false positives)

  P = TP/(TP+FP)

- Recall: the number of correct guesses (true positives) for the category divided by all the true documents in that category (true positives plus false negatives)

  R = TP/(TP+FN)

- F-score: derived measure combining precision and recall.

  F = 2PR/(P+R)

| | - | ~ | + | |
|---|---|---|---|---|
| - | 5 | 442 | 7 | 454 |
| ~ | 1 | 140 | 0 | 141 |
| + | 0 | 182 | 18 | 200 |
| | 6 | 764 | 25 | 795 |

$$P_{\sim} = \frac{140}{140+442+182} = .183$$

$$R_{-} = \frac{5}{5+442+7} = .011$$

$$F_{+} = \frac{2 \times .72 \times .09}{.72+.09} = .16$$

| | P | R | F |
|---|---|---|---|
| - | 83.3 | 1.1 | 2.2 |
| ~ | 18.3 | 99.3 | 30.1 |
| + | 72 | 9 | 16 |
| Avg | 57.9 | 36.5 | 16.4 |

# What does it tell us?

- Overall accuracy is low, because the model overpredicts neutral.
- Precision is pretty good for negative, and okay for positive. This means the simple rules “has the word ‘good’” and “has the word ‘bad’” are good predictors.

| | - | ~ | + | |
|---|---|---|---|---|
| - | 5 | 442 | 7 | 454 |
| ~ | 1 | 140 | 0 | 141 |
| + | 0 | 182 | 18 | 200 |
| | 6 | 764 | 25 | 795 |

| | P | R | F |
|---|---|---|---|
| - | 83.3 | 1.1 | 2.2 |
| ~ | 18.3 | 99.3 | 30.1 |
| + | 72 | 9 | 16 |
| Avg | 57.9 | 36.5 | 16.4 |

# Where do the rules go wrong?

- Confusion matrix for STS:

| | - | ~ | + | |
|---|---|---|---|---|
| - | 0 | 73 | 2 | 75 |
| ~ | 0 | 31 | 2 | 33 |
| + | 1 | 96 | 11 | 108 |
| | 1 | 200 | 15 | 216 |

- The one negative-labeled tweet that is actually positive, using the very positive expression “bad ass” (thus matching “bad”).

> Booz Allen Hamilton has a bad ass homegrown social collaboration platform. Way cool! #ttiv

# A bigger lexicon (rule set) and a better rule

```
pos_words = {"good","awesome","great","fantastic","wonderful"}
neg_words = {"bad","terrible","worst","sucks","awful","dumb"}
def polarity(document) =
   num_pos = count of words in document also in pos_words
   num_neg = count of words in document also in neg_words
  if (num_pos == 0 and num_neg == 0)
     neutral
  else if (num_pos > num_neg)
     positive
  else
     negative
```

| | Debate08 | HCR | STS | IMDB |
|---|---|---|---|---|
| Super simple | 20.5 | 21.6 | 19.4 | 27.4 |
| Small lexicon | 21.5 | 22.1 | 25.5 | 51.4 |

- Good improvements for five minutes of effort!
- Why such a large improvement for IMDB?

# IMDB: no neutrals!

- Data is from 10 star movie ratings (>=7 are pos, <= 4 are neg)
- Compare the confusion matrices!

## "Good/Bad" rule

| | - | ~ | + | |
|---|---|---|---|---|
| - | 2324 | 5476 | 4700 | 12500 |
| ~ | 0 | 0 | 0 | 0 |
| + | 651 | 7325 | 4524 | 12500 |
| | 2975 | 12801 | 9224 | 25000 |

Accuracy: 27.4

## Small lexicon with counting rule

| | - | ~ | + | |
|---|---|---|---|---|
| - | 5744 | 3316 | 3440 | 12500 |
| ~ | 0 | 0 | 0 | 0 |
| + | 1147 | 4247 | 7106 | 12500 |
| | 6891 | 7563 | 10546 | 25000 |

Accuracy: 51.4

# Sentiment lexicons: Bing Liu's opinion lexicon

- Bing Liu maintains and freely distributes a sentiment lexicon consisting of lists of strings.
  - Distribution page (direct link to rar archive)
  - Positive words: 2006
  - Negative words: 4783
  - Useful properties: includes mis-spellings, morphological variants, slang, and social-media mark-up
  - Note: may be used for academic and commercial purposes.

# Sentiment lexicons: MPQA

- The MPQA (Multi-Perspective Question Answering) Subjectivity Lexicon is maintained by Theresa Wilson, Janyce Wiebe, and Paul Hoffmann (Wiebe, Wilson, and Cardie 2005).
- Note: distributed under a GNU Public License (not suitable for most commercial uses).

| | Strength | Length | Word | Part-of-speech | Stemmed | Polarity |
|---|---|---|---|---|---|---|
| 1. | type=weaksubj | len=1 | word1=abandoned | pos1=adj | stemmed1=n | priorpolarity=negative |
| 2. | type=weaksubj | len=1 | word1=abandonment | pos1=noun | stemmed1=n | priorpolarity=negative |
| 3. | type=weaksubj | len=1 | word1=abandon | pos1=verb | stemmed1=y | priorpolarity=negative |
| 4. | type=strongsubj | len=1 | word1=abase | pos1=verb | stemmed1=y | priorpolarity=negative |
| 5. | type=strongsubj | len=1 | word1=abasement | pos1=anypos | stemmed1=y | priorpolarity=negative |
| 6. | type=strongsubj | len=1 | word1=abash | pos1=verb | stemmed1=y | priorpolarity=negative |
| 7. | type=weaksubj | len=1 | word1=abate | pos1=verb | stemmed1=y | priorpolarity=negative |
| 8. | type=weaksubj | len=1 | word1=abdicate | pos1=verb | stemmed1=y | priorpolarity=negative |
| 9. | type=strongsubj | len=1 | word1=aberration | pos1=adj | stemmed1=n | priorpolarity=negative |
| 10. | type=strongsubj | len=1 | word1=aberration | pos1=noun | stemmed1=n | priorpolarity=negative |
| ... | | | | | | |
| 8221. | type=strongsubj | len=1 | word1=zest | pos1=noun | stemmed1=n | priorpolarity=positive |

# Other sentiment lexicons

- SentiWordNet (Baccianella, Esuli, and Sebastiani 2010) attaches positive and negative real-valued sentiment scores to WordNet synsets (Fellbaum1998).
  - Note: recently changed license to permissive, commercial-friendly terms.
- Harvard General Inquirer is a lexicon attaching syntactic, semantic, and pragmatic information to part-of-speech tagged words (Stone, Dunphry, Smith, and Ogilvie 1966).
- Linguistic Inquiry and Word Counts (LIWC) is a proprietary database consisting of a lot of categorized regular expressions. Its classifications are highly correlated with those of the Harvard General Inquirer.

# When you have a big lexicon, use it!

- Using Bing Liu's Opinion Lexicon, scores across all datasets go up dramatically.

| | Debate08 | HCR | STS | IMDB |
|---|---|---|---|---|
| Super simple | 20.5 | 21.6 | 19.4 | 27.4 |
| Small lexicon | 21.5 | 22.1 | 25.5 | 51.4 |
| Opinion lexicon | 47.8 | 42.3 | 62 | 73.6 |

Well above (three-way) coin-flipping!

# If you don't have a big lexicon, bootstrap one

- There is a reasonably large literature on creating sentiment lexicons, using various sources such as WordNet (knowledge source) and review data (domain-specific data source).
- Advantage of review data: often able to obtain easily for many languages.
- See Chris Potts' 2011 SAS tutorial for more details:
  - http://sentiment.christopherpotts.net/lexicons.html
- A simple, intuitive measure is the log-likelihood ratio, which I'll show for IMDB data.

# Log-likelihood ratio: basic recipe

- Given: a corpus of positive texts, negative texts, and a held out corpus
- For each word in the vocabulary, calculate its probability in each corpus. E.g. for the positive corpus:

$$P_+(good) = \frac{count_+(good)}{\sum_{w_i \in vocab} count_+(w_i)}$$

- Compute its log-liklihood ratio for positive vs negative documents:

$$LLR(good) = log \frac{P_+(good)}{P_-(good)}$$

- Rank all words from highest LLR to lowest.

# LLR examples computed from IMDB reviews

\+

| | |
|---|---|
| edie | 16.069394855429 |
| antwone | 15.8553838121324 |
| din | 15.7474948645816 |
| goldsworthy | 15.5524343124634 |
| gunga | 15.5369301286125 |

⋮

\-

| | |
|---|---|
| kornbluth | -15.0901061313017 |
| kareena | -15.1154239321259 |
| tashan | -15.233206936755 |
| hobgoblins | -15.233206936755 |
| slater | -15.3183647248326 |

**Filter:**

$$P(w_i) \geq \frac{2}{|vocab|}$$

| | |
|---|---|
| perfection | 2.20422774489731 |
| captures | 2.05519247042604 |
| wonderfully | 2.02082497132301 |
| powell | 1.99331708656209 |
| refreshing | 1.8672999245198 |

⋮

| | |
|---|---|
| pointless | -2.47740636027027 |
| blah | -2.57814744950696 |
| waste | -2.67366867254484 |
| unfunny | -2.70848760424055 |
| seagal | -3.6618321047833 |

# Top 25 filtered positive and negative words using LLR on IMDB

- Some obvious film domain dependence, but also lots of generally good valence determinations.

\+ perfection captures wonderfully powell refreshing flynn delightful gripping beautifully underrated superb delight welles unforgettable touching favorites extraordinary stewart brilliantly friendship wonderful magnificent finest marie jackie

\- horrible unconvincing uninteresting insult uninspired sucks miserably boredom cannibal godzilla lame wasting remotely awful poorly laughable worst lousy redeeming atrocious pointless blah waste unfunny seagal

# Using the learned lexicon

- There are various ways to use the LLR ranks:
  - Take the top N of positive and negative and use them as the positive and negative sets.
  - Combine the top N with another lexicon (e.g. the super small one or the Opinion Lexicon).
  - Take the top N and manually prune words that are not generally applicable.
  - Use the LLR values as the input to a more complex (and presumably more capable) algorithm.
- Here we'll try three things:
  - IMDB100: the top 100 positive and 100 negative filtered words
  - IMDB1000: the top 1000 positive and 1000 negative filtered words
  - Opinion Lexicon + IMDB1000: take the union of positive terms in Opinion Lexicon and IMDB1000, and same for the negative terms.

# Better lexicons can get pretty big improvements!

| | Debate08 | HCR | STS | IMDB |
|---|---|---|---|---|
| Super simple | 20.5 | 21.6 | 19.4 | 27.4 |
| Small lexicon | 21.5 | 22.1 | 25.5 | 51.4 |
| Opinion lexicon | 47.8 | 42.3 | 62 | 73.6 |
| IMDB100 | 24.1 | 22.6 | 35.7 | **77.9** |
| IMDB1000 | 58 | 45.6 | 50.5 | 66 |
| Opinion Lexicon + IMDB1000 | **62.4** | **49.1** | **56** | 66.1 |

Nonetheless: for the reasons mentioned previously, this strategy eventually runs out of steam. It is a **starting** point.

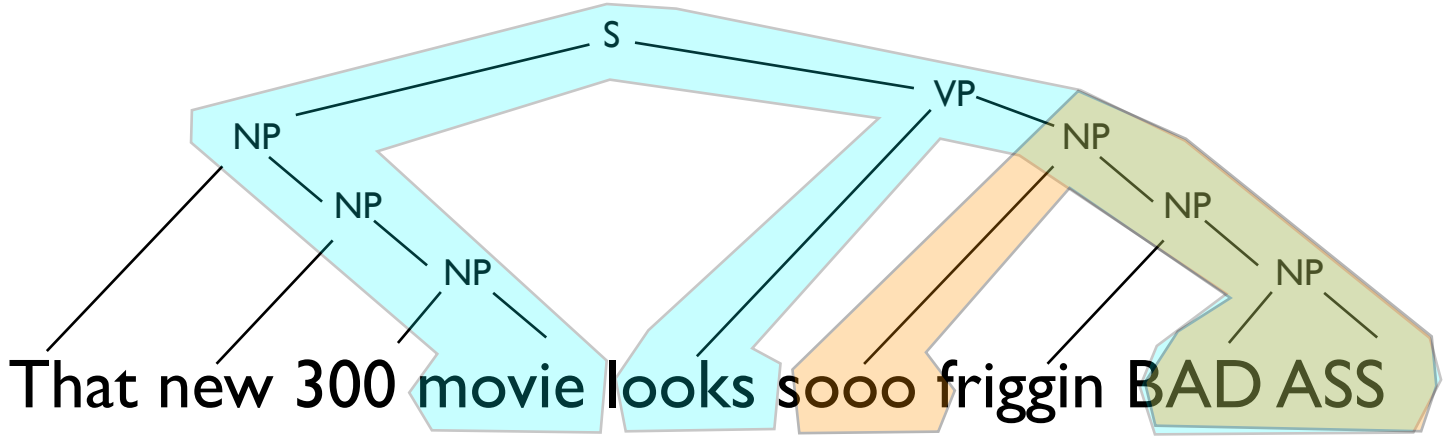
# Machine learning for classification

- Given a set of labeled points, there are many standard methods for learning linear classifiers. Some popular ones are:
  - Naive Bayes
  - Logistic Regression / Maximum Entropy
  - Perceptrons
  - Support Vector Machines (SVMs)
- The properties of these classifier types are widely covered in tutorials, code, and homework problems.
- There are various reasons to prefer one or the other of these, depending on amount of training material, tolerance for longer training times, and the complexity of features used.

# Features for document classification

- All of the linear classifiers require documents to be represented as points in some n-dimensional space.
- Each dimension corresponds to a feature, or observation about a subpart of a document.
- A feature's value is typically the number of times it occurs.
  - Ex: Consider the document "That new 300 movie looks sooo friggin bad ass. Totally BAD ASS!" The feature "the lowercase form of the word 'bad'" has a value of 2, and the feature "is_negative_word" would be 4 ("bad","ass","BAD","ASS").
- For many documentation classification tasks (e.g. spam classification), bag-of-words features are unreasonably effective.
- However, for more subtle tasks, including polarity classification, we usually employ more interesting features.

# Features for classification




```
w=that
w=new
w=300
w=movie
w=looks
w=sooo
w=friggin
w=bad
w=ass
```

```
w=so
```

```
bi=<START>_that
bi=that_new
bi=new_300
bi=300_movie
bi=movie_looks
bi=looks_sooo
bi=sooo_friggin
bi=friggin_bad
bi=bad_ass
bi=ass_.
bi=._<END>
```

```
wt=that_art
wt=new_adj
wt=300_noun
wt=movie_noun
wt=looks_verb
wt=sooo_adv
wt=friggin_adv
wt=bad_adj
wt=ass_noun
```

FEATURE ENGINEERING... (deep learning might help ease the burden)

```
subtree=S_NP_movie-S_VP_looks-S_VP_NP_bad_ass
```

```
subtree=NP_sooo_bad_ass
```

# Complexity of features

- Features can be defined on very deep aspects of the linguistic content, including syntactic and rhetorical structure.
- The models for these can be quite complex, and often require significant training material to learn them, which means it is harder to employ them for languages without such resources.
  - I'll show an example for part-of-speech tagging in a bit.
- Also: the more fine-grained the feature, the more likely it is rare to see in one's training corpus. This requires more training data, or effective semi-supervised learning methods.

# Recall the four sentiment datasets

| Dataset | Topic | Year | # Train | # Dev | #Test | Reference |
|---|---|---|---|---|---|---|
| Debate08 | Obama vs McCain debate | 2008 | **795** | 795 | 795 | Shamma, et al. (2009) "Tweet the Debates: Understanding Community Annotation of Uncollected Sources." |
| HCR | Health care reform | 2010 | **839** | 838 | 839 | Speriosu et al. (2011) "Twitter Polarity Classification with Label Propagation over Lexical Links and the Follower Graph." |
| STS | (Stanford) Twitter Sentiment | 2009 | - | 216 | - | Go et al. (2009) "Twitter sentiment classification using distant supervision" |
| IMDB | IMDB movie reviews | 2011 | **25,000** | 25,000 | - | Mas et al. (2011) "Learning Word Vectors for Sentiment Analysis" |

# Logistic regression, in domain

- When training on labeled documents from the same corpus.
- Models trained with Liblinear (via ScalaNLP Nak)
- Note: for IMDB, the logistic regression classifier only predicts positive or negative (because there are no neutral training examples), effectively making it easier than for the lexicon-based method.

| | Debate08 | HCR | STS | IMDB |
|---|---|---|---|---|
| Opinion Lexicon + IMDB1000 | 62.4 | 49.1 | 56 | 66.1 |
| Logistic Regression w/ bag-of-words | 60.9 | 56 | (no labeled training set) | 86.7 |
| Logistic Regression w/ extended features | **70.2** | **60.5** | - | |

# Logistic regression (using extended features), cross-domain

Evaluation corpora

Training corpora

| | Debate08 | HCR | STS |
|---|---|---|---|
| Debate08 | 70.2 | 51.3 | 56.5 |
| HCR | 56.4 | 60.5 | 54.2 |
| Debate08+HCR | **70.3** | **61.2** | **59.7** |

- In domain training examples add 10-15% absolutely accuracy (56.4 -> 70.2 for Debate08, and 51.3 -> 60.5 for HRC).
- More labeled examples almost always help, especially if you have no in-domain training data (e.g. 56.5/54.2 -> 59.7 for STS).

# Accuracy isn't enough, part 1

- The class balance can shift considerably without affecting the accuracy!

## D08+HRC on STS

| | - | ~ | + | |
|---|---|---|---|---|
| - | 58 | 12 | 5 | 75 |
| ~ | 7 | 24 | 2 | 33 |
| + | 34 | 27 | 47 | 108 |
| | 99 | 63 | 54 | 216 |

$$\frac{58+24+47}{216} = 59.7$$

## (Made up) Positive-heavy classifier

| | - | ~ | + | |
|---|---|---|---|---|
| - | 8 | 12 | 55 | 75 |
| ~ | 7 | 24 | 11 | 33 |
| + | 1 | 1 | 106 | 108 |
| | 16 | 28 | 172 | 216 |

$$\frac{8+15+106}{216} = 59.7$$

# Accuracy isn't enough, part 1

- Need to also consider the per-category precision, recall, and f-score.

| | - | ~ | + | |
|---|---|---|---|---|
| - | 58 | 12 | 5 | 75 |
| ~ | 7 | 24 | 2 | 33 |
| + | 34 | 27 | 47 | 108 |
| | 99 | 63 | 54 | 216 |

Acc: 59.7

| | P | R | F |
|---|---|---|---|
| - | 58.6 | 77.3 | 66.7 |
| ~ | 38.1 | 72.7 | 50 |
| + | 87 | 43.5 | 58 |
| Avg | 61.2 | 64.5 | 58.2 |

Big differences in precision for the three categories!

# Accuracy isn't enough, part 2

- Errors on neutrals are typically less grievous than positive/negative errors, yet raw accuracy makes one pay the same penalty.

## D08+HRC on STS

| | - | ~ | + | |
|---|---|---|---|---|
| - | 58 | 12 | 5 | 75 |
| ~ | 7 | 24 | 2 | 33 |
| + | 34 | 27 | 47 | 108 |
| | 99 | 63 | 54 | 216 |

- One solution: allow varying penalties such that no points are awarded for positive/negative errors, but some partial credit is given for positive/neutral and negative/neutral ones.

# Accuracy isn't enough, part 3

- Who says the gold standard is correct? There is often significant variation among human annotators, especially for positive vs neutral and negative vs neutral.
- Solution one: work on your annotations (including creating conventions) until you get very high inter-annotator agreement.
  - This arguably reduces the linguistic variability/subtlety characterized in the annotations.
  - Also, humans often fail to get the intended sentiment, e.g. sarcasm.
- Solution two: measure performance differently. For example, given a set of examples annotated by three or more human annotators and the machine, is the machine distinguishable from the humans in terms of the amount it disagrees with their annotations?

# Accuracy isn't enough, part 4

- Often, what is of interest is an aggregate sentiment for some topic or target. E.g. given a corpus of tweets about cars, 80% of the mentions of the Ford Focus are positive while 70% of the mentions of the Chevy Malibu are positive.
- Note: you can get the sentiment value wrong for some of the documents while still getting the overall, aggregate sentiment correct (as errors can cancel each other).
- Note also: generally, this requires aspect-based analysis (more later).

# Caveat emptor, part 1

- In measuring accuracy, the methodology can vary dramatically from vendor to vendor, at times in unclear ways.
- For example, some seem to measure accuracy by presenting a human judge with examples annotated by a machine. The human then marks which examples they believe were incorrect. Accuracy is then num_correct/num_examples.
  - Problem: people get lazy and often end up giving the machine the benefit of the doubt.
- I have even heard that some vendors take their high-confidence examples and do the above exercise. This is basically cheating: high-confidence machine label assignments are on average more correct than low-confidence ones.

# Caveat emptor, part 2

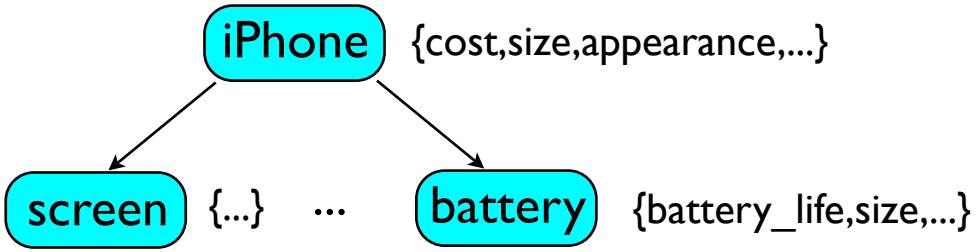
- Performance on in-domain data is nearly always better than out-of-domain (see the previous experiments).
- The nature of the world is that the language of today is a step away from the language of yesterday (when you developed your algorithm or trained your model).
- Also, because there are so many things to talk about (and because people talk about everything), a given model is usually going to end up employed in domains it never saw in its training data.

# Caveat emptor, part 3

- With nice, controlled datasets like those given previously, the experimenter has total control over which documents her algorithm is applied too.
- However, a deployed system will likely confront many irrelevant documents, e.g.
  - documents written in other languages
  - Sprint the company wants tweets by their customers, but also get many tweets of people talking about the activity of sprinting.
  - documents that match, but which are not about the target of interest
  - documents that should have matched, but were missed in retrieval
- Thus, identification of relevant documents and even sub-documents with relevant targets, is an important component of end-to-end sentiment solutions.

# Identifying targets and aspects

- We can specify targets, their sub-components, and their attributes:



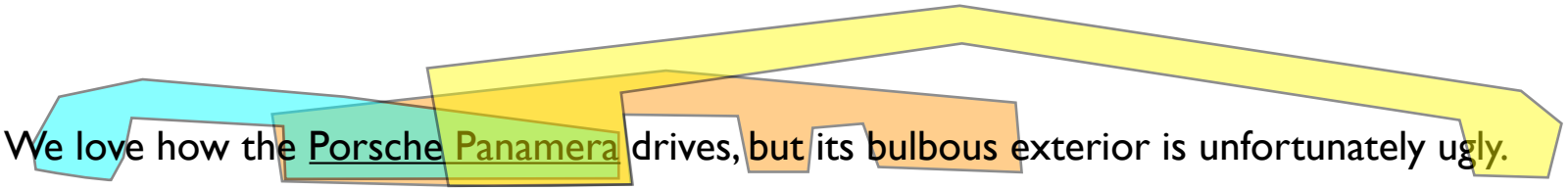

- But language is varied and evolving, so we are likely to miss many ways to refer to targets and their aspects.
  - E.g. A person declaring knowledge about phones might forget (or not even know) that "juice" is a way of referring to power consumption.
  - Also: there are many ways of referring to product lines (and their various releases, e.g. *iPhone 4s*) and their competitors, and we often want to identify these semi-automatically.
- Much research has worked on bootstrapping these. See Bing Liu's tutorial for an excellent overview:
  - http://www.cs.uic.edu/~liub/FBS/Sentiment-Analysis-tutorial-AAAI-2011.pdf

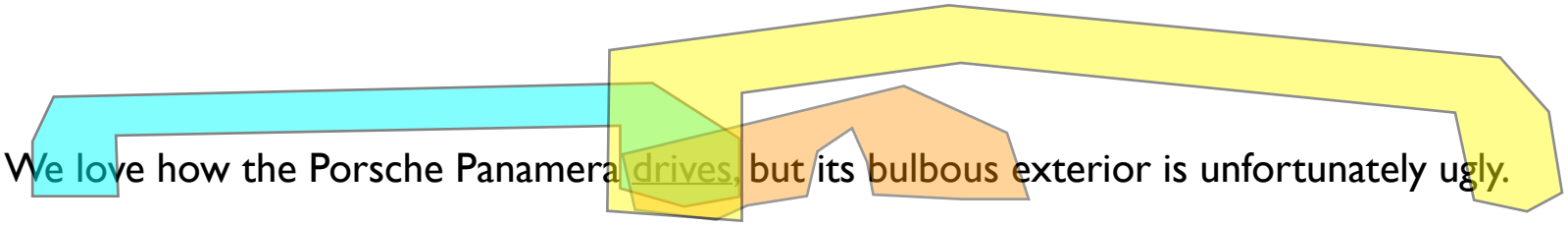
# Target-based feature engineering

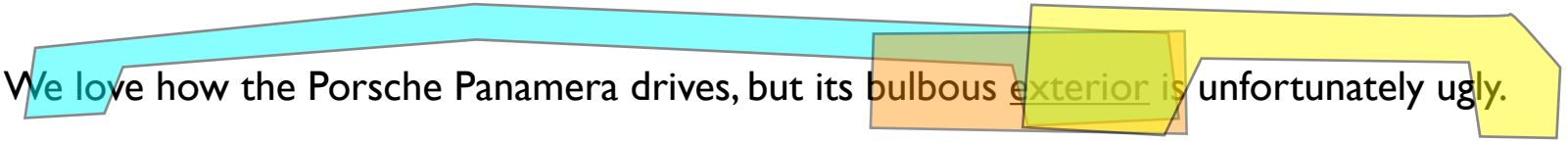
- Given a sentence like “We love how the Porsche Panamera drives, but its bulbous exterior is unfortunately ugly.”
  - NER to identify the “Porsche Panamera” as the target
  - Aspect identification to see that opinions are being expressed about the car’s driving and styling.
  - Sentiment analysis to identify positive sentiment toward the driving and negative toward the styling.
- Targeted sentiment analysis require positional features
  - use string relationship to the target or aspect
  - or use features from a parse of the sentence (if you can get it)

# Positional features

- In addition to the standard document-level features used previously, we build features particularized for each target.




- These are just a subset of the many possible features.

# Challenges

- Positional features greatly expands the space of possible features.
- We need more training data to estimate parameters for such features.
- Highly specific features increase the risk of overfitting to whatever training data you have.
- Deep learning has a lot of potential to help with learning feature representations that are effective for the task by reducing the need for careful feature engineering.
- But obviously: we need to be able to use this sort of evidence in order to do the job well via automated means.

# Polarity classification for Twitter

Obama looks good. #tweetdebate #current

McCain is not answering the questions #tweetdebate

Sen McCain would be a very popular President - $5000 tax refund per family! #tweetdebate

•
•
•

"it's like you can see Obama trying to remember all the "talking points" and get his slogans out there #tweetdebate"

Logistic regression... and... done!

What if instance labels aren't there?

# No explicitly labeled examples?

**Positive/negative ratio using polarity lexicon.**

➡ Easy & works okay for many cases, but fails spectactularly elsewhere.

**Emoticons as labels + logistic regression.**

➡ Easy, but emoticon to polarity mapping is actually vexed.

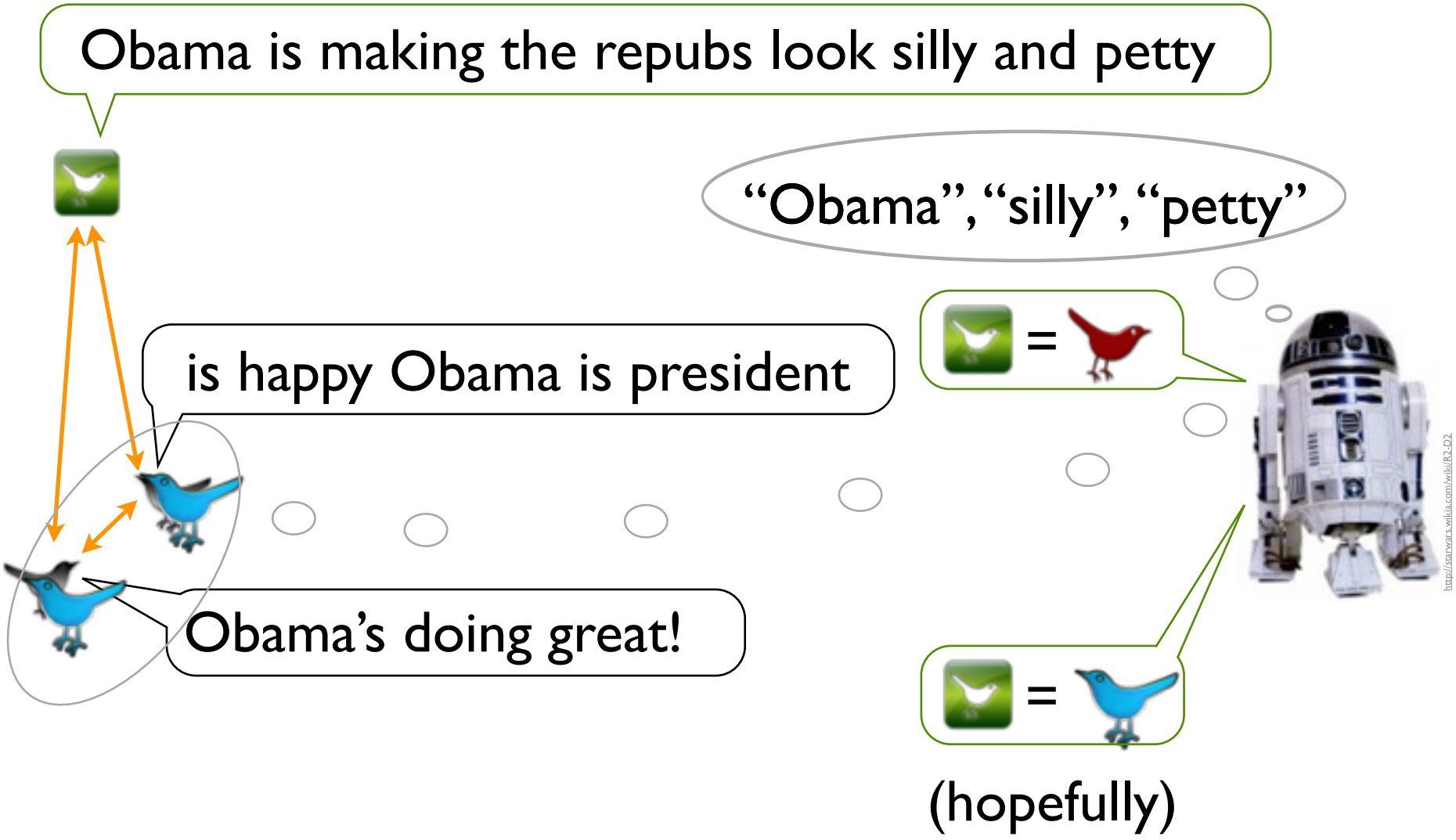
**Label propagation using the above as seeds.**

➡ Noisy labels provide soft indicators, the graph smooths things out.

**If you have annotations, you can use those too.**

➡ Including ordered labels like star ratings: see Talukdar & Crammer 2009

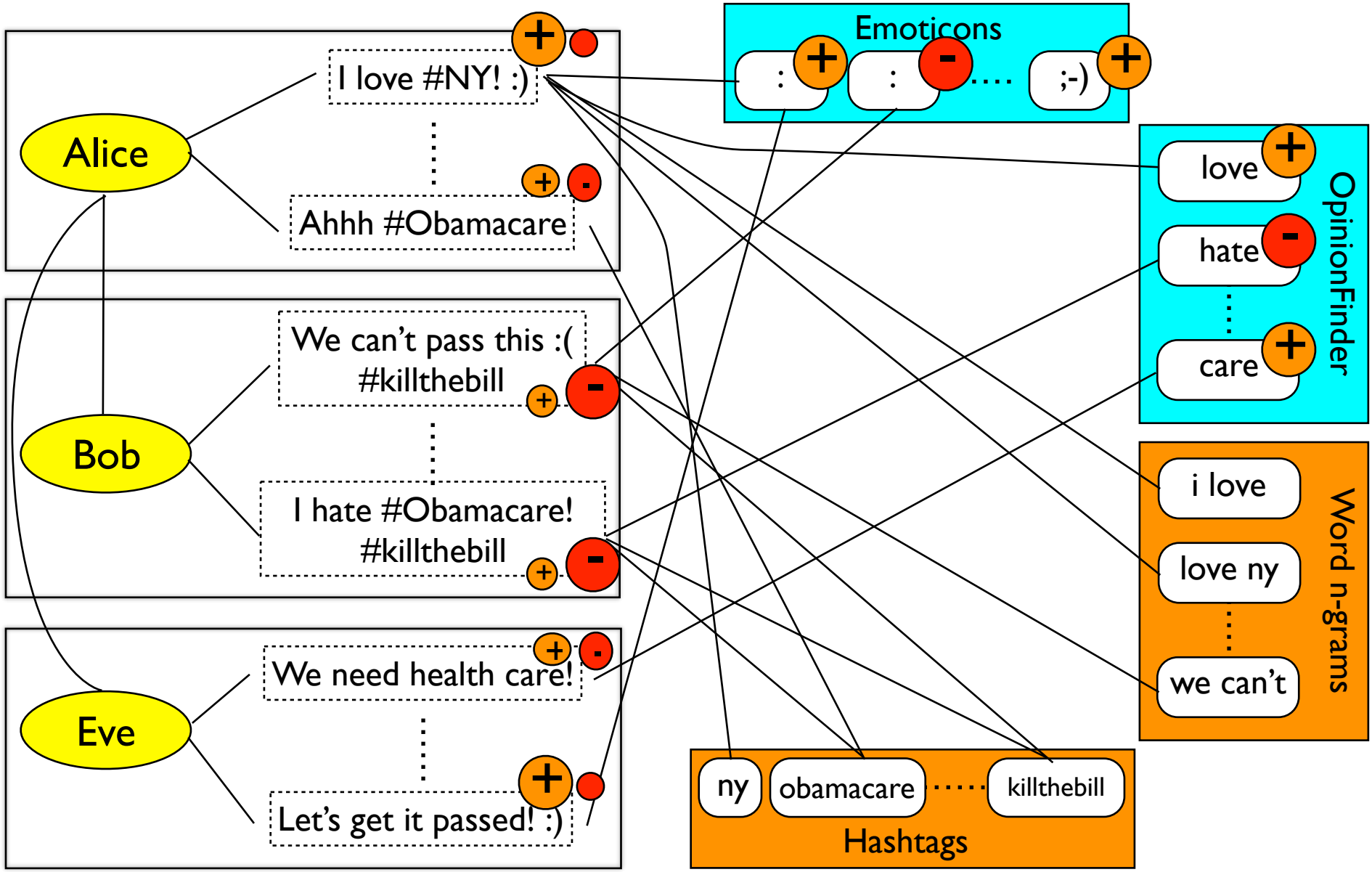
# Using social interaction: Twitter sentiment




Papers: Speriosu et al. 2011; Tan et al. KDD 2011

bird images from http://www.mytwitterlayout.com/

# Twitter polarity graph with knowledge and noisy seeds



# Results: polarity assignment (positive/negative, no neutral)

| | Stanford Twitter Sentiment | Obama-McCain Debate | Health Care Reform |
|---|---|---|---|
| Random | 50 | 50 | 50 |
| Lexicon Ratio | 72.1 | 59.1 | 58.1 |
| Emoticon-trained (Logistic regression) | 83.1 | 61.3 | 62.9 |
| Label propagation | **84.7** | **66.7** | **71.2** |

Take-home message: label propagation can make effective use of labeled features (from external knowledge sources) and noisy annotations.