Natural Language Processing

Lecture 2: *text classification*

Dan Goldwasser

*more like 10000000000

The 10000 feet view*

- NLP is about models that can process human languages.
- That’s a pretty open definition.. **We need a better definition!**

Is the word “time” a <u>verb</u> or a <u>noun</u>?

“Time flies like an arrow.”

Interpretation 1: *Time moves forward similar to how an arrow flies.*
Interpretation 2: *This is a command, telling you to measure the speed of the flies similar to how you would measure the speed of an arrow.*
…*(several other options)*

Which one of these two options?

Text Classification

- Classification is a fundamental and well-understood machine learning tool.
- It also provides a useful abstraction for many NLP tasks!
- Typically relying on a supervised learning pipeline:
  - Collect a large dataset of relevant texts, annotate with relevant labels.
  - Build a classifier using one of many possible learning algorithms.
  - Hopefully, it **generalizes** to new data.
- In the next two lecture we'll talk about this pipeline (evaluation, learning algorithms, etc.).

BUT also …

Text Classification

- Text classification is more than "*dumping data on an algorithm*"

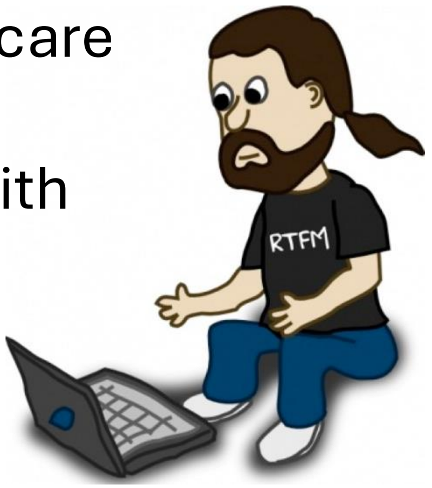
- **It's a way to model different aspects of text interpretation.**
  - I.e., how do you formulate the classification task?
  - How do you decide what are the relevant labels?

- **It's a way to introduce relevant linguistic knowledge.**
  - What are the relevant features?
  - What kind of assumptions are you making by picking different choices?

Sentiment Classification

- **Sentiment Analysis**
  - Interpretation of product reviews: positive/negative/neutral
  - Prediction is done overall entire text
- **Aspect based sentiment**
  - Identify the product aspects users care about
  - identify and associate sentiment with these aspects based on text

I just bought *company-A* newest laptop. The display is *awesome*, the speakers are *not that great*. I'm *happy* with the performance, but I think they charge too much for it!



Display: Positive
Speakers: Negative
Performance: Positive
Price: Negative

Sentiment Classification

- How should we define it as a classification task?
- What could be useful features here?
- What assumptions are you making?

I just bought *company-A* newest laptop. The display is ***awesome***, the speakers are ***not that great***. I'm ***happy*** with the performance, but I think they charge too much for it!

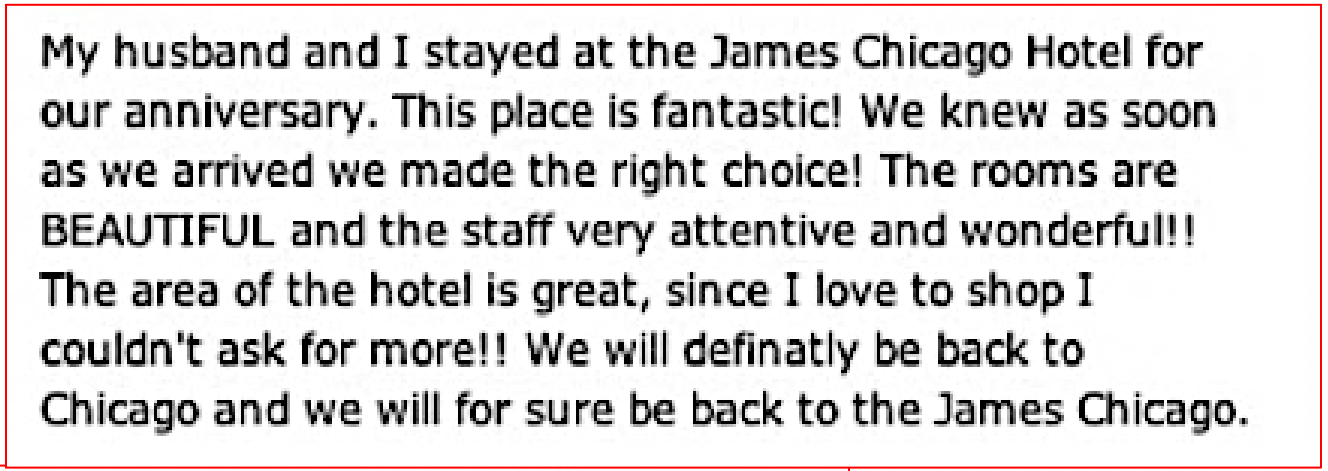
Sentiment Classification, *take-2*

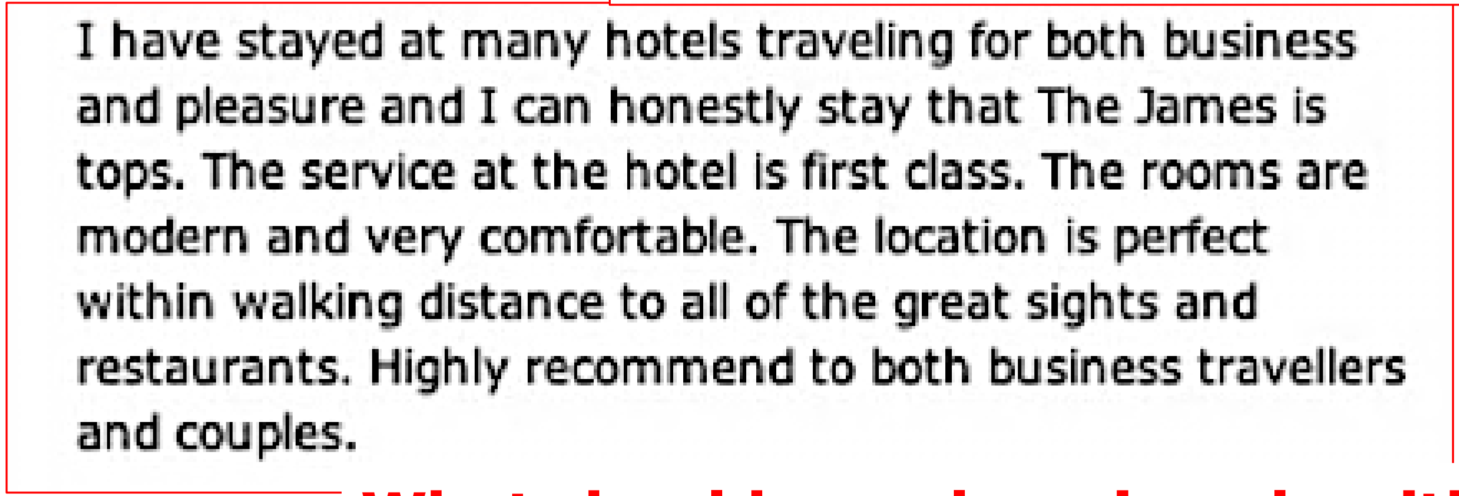
Dude, I just watched this horror flick! Selling points: nightmares scenes, torture scenes, terrible monsters that was so bad a##!

Don’t buy the popcorn it was terrible, the monsters selling it must have wanted to torture me, it was so bad it gave me nightmares!

Deceptive Reviews

Which of these two hotel reviews is deceptive opinion spam?


My husband and I stayed at the James Chicago Hotel for our anniversary. This place is fantastic! We knew as soon as we arrived we made the right choice! The rooms are BEAUTIFUL and the staff very attentive and wonderful!! The area of the hotel is great, since I love to shop I couldn't ask for more!! We will definatly be back to Chicago and we will for sure be back to the James Chicago.


I have stayed at many hotels traveling for both business and pleasure and I can honestly stay that The James is tops. The service at the hotel is first class. The rooms are modern and very comfortable. The location is perfect within walking distance to all of the great sights and restaurants. Highly recommend to both business travellers and couples.

What should your learning algorithm look at?

Finding Deceptive Opinion Spam by Any Stretch of the Imagination. Ott etal. ACL 2011

Power Relations

Can subtle, *domain-independent* linguistic cues reveal (situational) power?

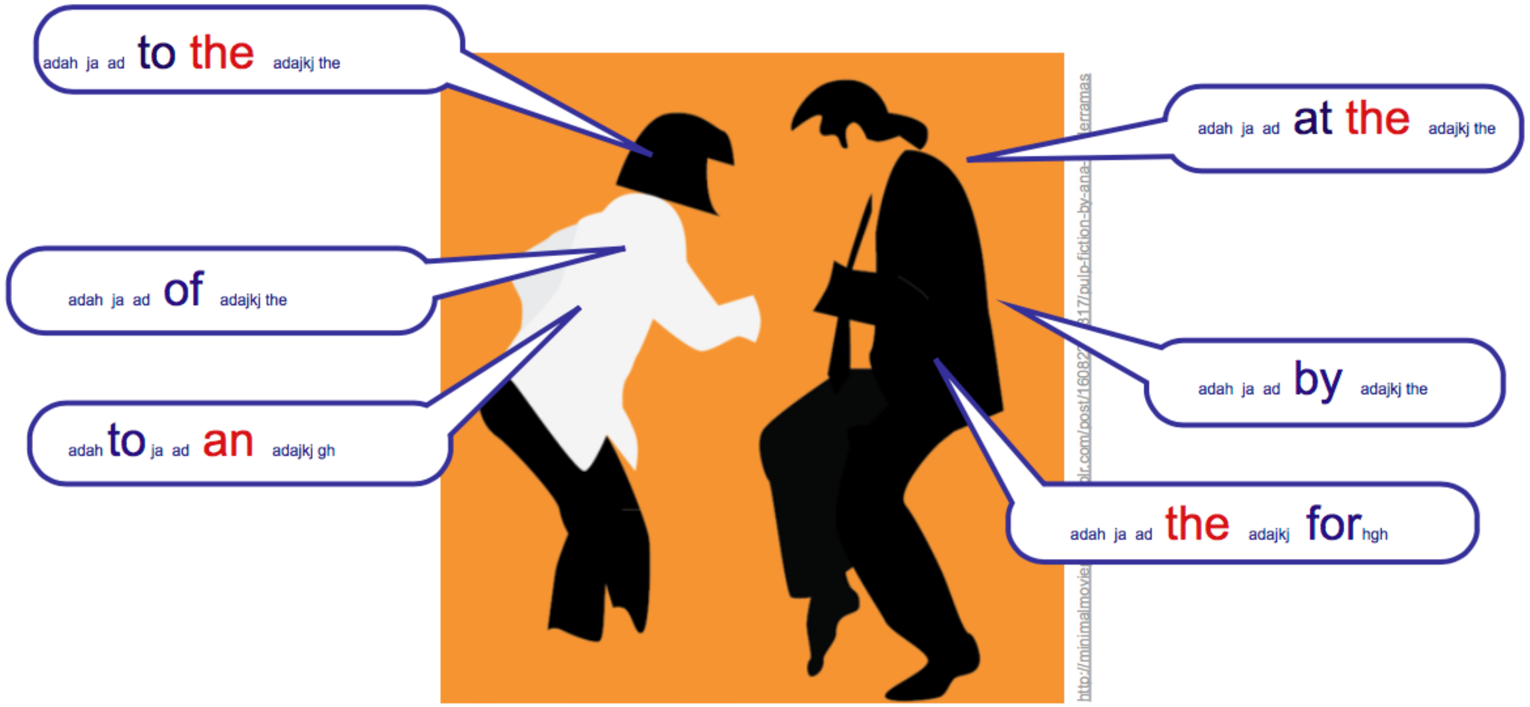
Intuition: each domain (i.e., situation) defines relevant expression of power. How can we generalize?

Who's in charge?

What should your learning algorithm look at?

Power Relations

Communicative behaviors are "patterned and coordinated, like a dance" [Niederhoffer and Pennebaker 2002]


adah ja ad to the adajkj the
adah ja ad of adajkj the
adah to ja ad an adajkj gh
adah ja ad at the adajkj the
adah ja ad by adajkj the
adah ja ad the adajkj for hgh
http://minimalmovie...lr.com/post/16082...317/pulp-fiction-by-ana-...erramas

Echoes of Power: Language Effects and Power Differences in Social Interaction.
Danescu-Niculescu-Mizil et-al . WWW 2012.

Text Classification

FunFact: Many complex NLP tasks can be broken down into several interconnected **classification** tasks. *For example – autoregressive language models, **predict** the next word!*

- **Assigning text to categories**
  - Spam/phishing detection
  - Sentiment/stance analysis
  - Topic classification
  - Authorship
  - Author profile: gender, age, education,…
- **Many, many more examples!**

Build an intuition – *which problems are easy and which ones hard?*

What makes a text classification task hard?

Basic Definitions

- Given: D a set of labeled examples $\{<x,y>\}$
- ***Goal**: Learn a function* $f(x)$ *s.t.* $f(x) = y$
  - ***Note**: y can be binary, or categorical (multi-class)*
  - The input x is represented as a vector of **features**
- Break D into three parts:
  - **Training** set (used by the learning algorithm)
  - **Test** set (evaluate the learned model)
  - **Development** set (tuning the learning algorithm)
- **Evaluation**:
  - performance measured over the test set
  - **Accuracy**: proportion of correct predictions (test data)

Precision and Recall

- Given a dataset, we train a classifier that gets 99% accuracy
- **Did we do a good job?**
- Build a classifier for brain tumor:
  - 99.9% of brain scans do not show signs of tumor
  - ***Did we do a good job?***
- By simply saying "NO" to all examples we reduce the error by a factor of 10!
  - ***Clearly Accuracy is not the best way to evaluate the learning system when the data is heavily skewed!***
- **Intuition**: we need a measure that captures the class we care about! (rare)

Precision and Recall

- The learner can make two kinds of mistakes:
  - False Positive
  - False Negative

	True Label: 1	True Label: 0
Predicted: 1	True Positive	False Positive
Predicted: 0	False Negative	True Negative

- **Precision:**
- *"when we predicted the rare class, how often are we right?"*

$$\frac{\text{True Pos}}{\text{Predicted Pos}} = \frac{\text{True Pos}}{\text{True Pos} + \text{False Pos}}$$

- **Recall**
- "Out of all the instances of the rare class, how many did we catch?"

$$\frac{\text{True Pos}}{\text{Actual Pos}} = \frac{\text{True Pos}}{\text{True Pos} + \text{False Neg}}$$

F Score

- Precision and Recall give us two reference points to compare learning performance

	Precision	Recall
Algorithm 1	0.5	0.4
Algorithm 2	0.7	0.1
Algorithm 3	0.02	1

- *Which algorithm is better?*

We need a single score

- Option 1: **Average** $\frac{P+R}{2}$
- Option 2: **F-Score** $2\frac{PR}{P+R}$

Properties of f-score:

- Ranges between 0-1
- Prefers precision and recall with similar values

Linear Classification Models

- *Linear* relationship between input and output

Vector generated by a feature function

$$\hat{y} = \arg\max_{y} \theta^{\mathsf{T}} \mathbf{f}(\mathbf{x}, y)$$

Feature weights

Predicted Class

Bag of Words Feature Representation

- **BoW:** Simplest (yet surprisingly effective) choice

I	You	He	Liked	liked	Apples	Pen	.	?	!
1	0	0	0	1	1	0	1	0	0
0	0	0	1	0	0	0	0	0	1
0	0	0	1	0	0	0	0	1	0

- **Issues**: dropping function words, long tail, stemming
  - Should **liked** and **Liked** be separate features?
- Features are associated with weights.
- Final decision: $\hat{y} = \arg\max_{y} \boldsymbol{\theta}^{\top} \mathbf{f}(\mathbf{x}, y)$

Simple Example: Naïve Bayes

- **Naïve Bayes**: simple probabilistic classifier
  - Given a set of labeled data:
    - Documents **D**, each associated with a label **v**
    - Simple feature representation: BoW
  - **Learning**:
    - Construct a probability distribution P(v|d)
  - **Prediction**:
    - Assign the label with the highest probability
- Relies on **strong** simplifying assumptions

Naïve Bayes: Independence Assumptions

- Basic idea: (sentiment analysis)
    - "I loved this movie, it's awesome! I couldn't stop laughing for two hours!"
    - Mapping input to label can be done by *representing the frequencies of individual words*
    - **Document = word counts**
- *Simple, yet surprisingly powerful representation!*
- *Used in Naïve Bayes as independence assumptions*

Bayes Rule

- Naïve Bayes is a simple probabilistic classification method, based on Bayes rule.

$$\mathbf{P(v \mid d) = P(d \mid v)\, {}^{P(v)}\!/_{P(d)}}$$

Naïve Bayes

- The learner considers a set of candidate labels, and picks the most probable given the observed data.
- Such maximally probable assignment is called maximum a posteriori assignment (**MAP**); Bayes theorem is used to compute it:

$$v_{MAP} = \text{argmax}_{v \in V} P(v|D) = \text{argmax}_{v \in v} P(D|v)\, P(v)/P(D)$$

$$= \text{argmax}_{v \in V} P(D|v)\, P(v)$$

Since P(D) is the same for all $v \in V$

Naïve Bayes

- How can we compute P(v |D)?
  - Basic idea: represent document as a set of features, such as BoW features

$$v_{MAP} = \operatorname{argmax}_{v_j \in V} P(v_j \mid x) = \operatorname{argmax}_{v_j \in V} P(v_j \mid x_1, x_2, \ldots, x_n)$$

$$v_{MAP} = \operatorname{argmax}_{v_j \square V} \frac{P(x_1, x_2, \ldots, x_n \mid v_j) P(v_j)}{P(x_1, x_2, \ldots, x_n)}$$

$$= \operatorname{argmax}_{v_j \square V} P(x_1, x_2, \ldots, x_n \mid v_j) P(v_j)$$

Parameter Estimation

$$V_{MAP} = \text{argmax}_v \, P(x_1, x_2, \ldots, x_n \mid v) \, P(v)$$

- Given training data we can estimate the two terms
- Estimating $P(v)$ is **easy**. For each value v count how many times it appears in the training data.

***Question**: Assume binary x_i's. How many parameters does the model require?*

- However, it is not feasible to estimate $P(x_1,\ldots, x_n \mid v)$
    - In this case we have to estimate, for each target value, the probability of each instance (most of which will not occur)
- In order to use a Bayesian classifier in practice, we *need to make assumptions* that will allow us to estimate these quantities.

Detour: the chain rule

The joint probability P(x_1,..,x_n) can be rewritten as a product:

$$\mathrm{P}(X_n,\ldots,X_1) = \mathrm{P}(X_n | X_{n-1},\ldots,X_1)\cdot\mathrm{P}(X_{n-1},\ldots,X_1)$$

Repeating this process n times for each variable results in:

$$\mathrm{P}\left(\bigcap_{k=1}^{n} X_k\right) = \prod_{k=1}^{n}\mathrm{P}\left(X_k \,\middle|\, \bigcap_{j=1}^{k-1} X_j\right)$$

For example:

$$\begin{aligned}\mathrm{P}(X_4,X_3,X_2,X_1) &= \mathrm{P}(X_4 \mid X_3,X_2,X_1)\cdot\mathrm{P}(X_3,X_2,X_1)\\ &= \mathrm{P}(X_4 \mid X_3,X_2,X_1)\cdot\mathrm{P}(X_3 \mid X_2,X_1)\cdot\mathrm{P}(X_2,X_1)\\ &= \mathrm{P}(X_4 \mid X_3,X_2,X_1)\cdot\mathrm{P}(X_3 \mid X_2,X_1)\cdot\mathrm{P}(X_2 \mid X_1)\cdot\mathrm{P}(X_1)\end{aligned}$$

https://en.wikipedia.org/wiki/Chain_rule_(probability)

NB: Independence Assumptions

Conditional Independence:

Assume feature probabilities are independent given the label

$$P(x_i | y) = P(x_i | x_k; y)$$ (forall feature pairs, i,k)

$$\boxed{P(Y|\mathbf{X}) \propto P(\mathbf{X}|Y)P(Y)}$$ **Bayes rule**

$$\propto \boxed{\prod_{i=1}^{m} P(X_i|Y)} P(Y)$$

Naive assumption

Question: *How many parameters do we need to estimate now?*

Simple example

$P(y)\ P(x_1, x_2, x_3 | y) = P(x_1, x_2, x_3, y)$

using the chain rule

$P(x1, x2, x3, y) = P(x1 | x2, x3, y)\ P(x2, x3, y)$

$= P(x1 | x2, x3, y)\ P(x2 | x3, y)\ P(x3, y)$

$= P(x1 | x2, x3, y)\ P(x2 | x3, y)\ P(x3 | y)\ P(y)$

assuming conditional independence

$= P(x_1 | y)\ P(x_2 | y)\ P(x_3 | y)\ P(y)$ $\leftarrow$ we end up with Naïve Bayes

Naïve Baye: independence assumptions

- Bag of words representation:
  - Word position can be ignored
- Conditional Independence: Assume feature probabilities are independent given the label
  - $P(x_i|v_j) = P(x_i|x_{i-1}; v_j)$

What is the difference from "x and y are independent"

- Both assumptions are **<u>not true</u>**
  - Help simplify the model
  - *Simple models work well*

how many features are needed now?

Independence Assumptions

Conditional Independence: $P(x_i \mid v_j) = P(x_i \mid x_{i-1}; v_j)$

Dude, I just watched this horror flick! Selling points: nightmares scenes, torture scenes, terrible monsters that was so bad a##!

$$P(\text{“terrible”} \mid pos) \overset{?}{=} P(\text{“terrible”} \mid \text{“food”}, pos)$$

$$P(\text{“terrible”} \mid pos) \overset{?}{=} P(\text{“terrible”} \mid \text{“monsters”}, pos)$$

Naïve Bayes

$$V_{MAP} = \mathrm{argmax}_v\, P(x_1, x_2, \ldots, x_n \mid v)P(v)$$

$$P(x_1, x_2, \ldots, x_n \mid v_j) =$$

$$= P(x_1 \mid x_2, \ldots, x_n, v_j)P(x_2, \ldots, x_n \mid v_j)$$

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

$$= \ldots\ldots.$$

$$= P(x_1 \mid x_2, \ldots, x_n, v_j)P(x_2 \mid x_3, \ldots, x_n, v_j)P(x_3 \mid x_4, \ldots, x_n, v_j)\ldots P(x_n \mid v_j)$$

Assumption: feature values are independent given the target value

$$= \square_{i=1}^{n} P(x_i \mid v_j)$$

Estimating Probabilities

Assume a document classification problem, using word features

How do we estimate $P(\text{word}_k \mid v)$ *?*

$$P(\text{word}_k \mid v) = \frac{\#(\text{word}_k \text{ appears in training in v documents})}{\#(\text{v documents})} = \frac{n_k}{n}$$

Data sparsity is a problem

-- if n is small, the estimate is not accurate

-- if n_k is 0, it will dominate the estimate: we will never predict v if a word that never appeared in training (with v) appears in the test data

Zero counts are a problem

- If an attribute value does not occur in training example, we assign **zero** probability to that value: $P(x_j \mid y) = 0$

- How does that affect the conditional probability $P(y \mid x)$?
  (assuming that x_j appears in x)

- **It equals 0**
- Why is this a problem?
- Adjust for zero counts by "smoothing" probability estimates

Robust Estimation of Probabilities

- There are many ways to do it, some better justified;
- An empirical issue.

$$P(x_k \mid v) = \frac{n_k + mp}{n + m}$$

- **Here:**
  - **n_k** is *#(of occurrences of the word in the presence of v)*
  - **n** *is #(of occurrences of the label v)*
  - **p** is a *prior estimate of v (e.g., uniform)*
  - **m** is *equivalent sample size (# of labels)*
- **Laplace Rule**: for the Boolean case, p=1/2 , m=2

$$P(x_k \mid v) = \frac{n_k + 1}{n + 2}$$

Smoothing: Laplace correction

X_1

Y		Low	Medium	High
	Yes	10	13	17
	No	2	13	0

Laplace correction

$$\hat{\theta}_i = \frac{x_i + \alpha}{N + \alpha d} \qquad (i = 1, \ldots, d),$$

Simple version:

Numerator: ***add 1***

Denominator: ***add k***, *where k=number of possible values of X*

$$P[X_1 = \text{High} \mid Y = \text{No}] = \frac{0 \quad \boxed{+1}}{(2+13+0) \boxed{+3}}$$

Adds uniform prior

Numerical Stability

- Recall: NB classifier: $\propto \prod_{i=1}^{m} P(X_i|Y)P(Y)$
  - **Multiplying probabilities can get us into problems!**
  - Imagine computing the probability of 2000 independent coin flips
  - Most programming environments: $(.5)^{2000}=0$

Numerical Stability

- Our problem: **Underflow Prevention**
- Recall: $log(xy) = log(x) + log(y)$
- better to sum logs of probabilities rather than multiplying probabilities.
- Class with highest final un-normalized log probability score is still the most probable.

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

Naïve Bayes

- Easy to implement
- **Converges very quickly**
  - Learning is just counting
- **Performs well in practice**
  - Applied to many document classification tasks
  - If data set is small, NB can perform better than sophisticated algorithms
- **Strong independence assumptions**
  - If assumptions hold: **NB is the optimal classifier**
  - Even if not, can perform well

Naïve Bayes: two classes case

- Assume *d*-dimensional binary input vector, classifying between two classes.
- We can rewrite the decision rule as follows:

$$\frac{P(\mathbf{x}|y=1)P(y=1)}{P(\mathbf{x}|y=0)P(y=0)} \geq 1$$

- And using the Naïve Bayes assumptions:

$$\underset{j}{\frac{P(y=1)}{P(y=0)}} \cdot \prod_{i=0}^{d} \frac{P(x_j|y=1)}{P(x_j|y=0)} \geq 1$$

Naïve Bayes: two classes case

Last step: $\frac{P(y=1)}{P(y=0)} \cdot \prod_{i=0}^{d} \frac{P(x_j|y=1)}{P(x_j|y=0)} \geq 1$

- **Let's simplify notation:**
  - Rename $P(y=1)$ as p
  - Rename $P(x_j=1|y=1)$ as a_j
  - Rename $P(x_j=1|y=0)$ as b_j
- As a result, we can define:

$$P(x_j|y=0) = b_j^{x_j}\,(1-b_j)^{(1-x_j)} \qquad P(x_j|y=1) = a_j^{x_j}\,(1-a_j)^{(1-x_j)}$$

And express the decision rule: $$\frac{p}{1-p} \cdot \prod_{j=0}^{d} \frac{a_j^{x_j}\,(1-a_j)^{(1-x_j)}}{b_j^{x_j}\,(1-b_j)^{(1-x_j)}} \geq 1$$

Naïve Bayes: two classes case

- Now, collect the constants:

$$\left(\frac{p}{1-p} \prod_{j=0}^{d} \frac{1-a_j}{1-b_j} \right) \cdot \prod_{j=0}^{d} \left(\frac{a_j}{b_j} \cdot \frac{1-b_j}{1-a_j} \right)^{x_j} \geq 1$$

- And, take the log:

$$\log \left(\frac{p}{1-p} \prod_{j=0}^{d} \frac{1-a_j}{1-b_j} \right) + \sum_{j=0}^{d} x_j \log \left(\frac{a_j}{b_j} \cdot \frac{1-b_j}{1-a_j} \right) \geq 0$$

Naïve Bayes: two classes case

$$\underbrace{\log\left(\frac{p}{1-p}\prod_{j=0}^{d}\frac{1-a_j}{1-b_j}\right)} + \sum_{j=0}^{d} x_j \underbrace{\log\left(\frac{a_j}{b_j}\cdot\frac{1-b_j}{1-a_j}\right)} \geq 0$$

Note that the value of this term is not dependent on x, it is a constant value! Let's denote it as b

This term depends on x_j, let's rename it as w_j

Making this substitution we can a familiar term:

$$b + \sum_{j=0}^{d} x_j w_j \geq 0$$

"Aha! Moment": NB is a linear classifier!

NB Expressivity Revisited

- The independence assumptions made by NB capture the connection between each feature and the output.
  - Unlike DT where each path defines this connection over a combination of features.
- If we define the NB over the log P(Y) P(X|Y) we get

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

Which looks like a Linear model : Ax+b

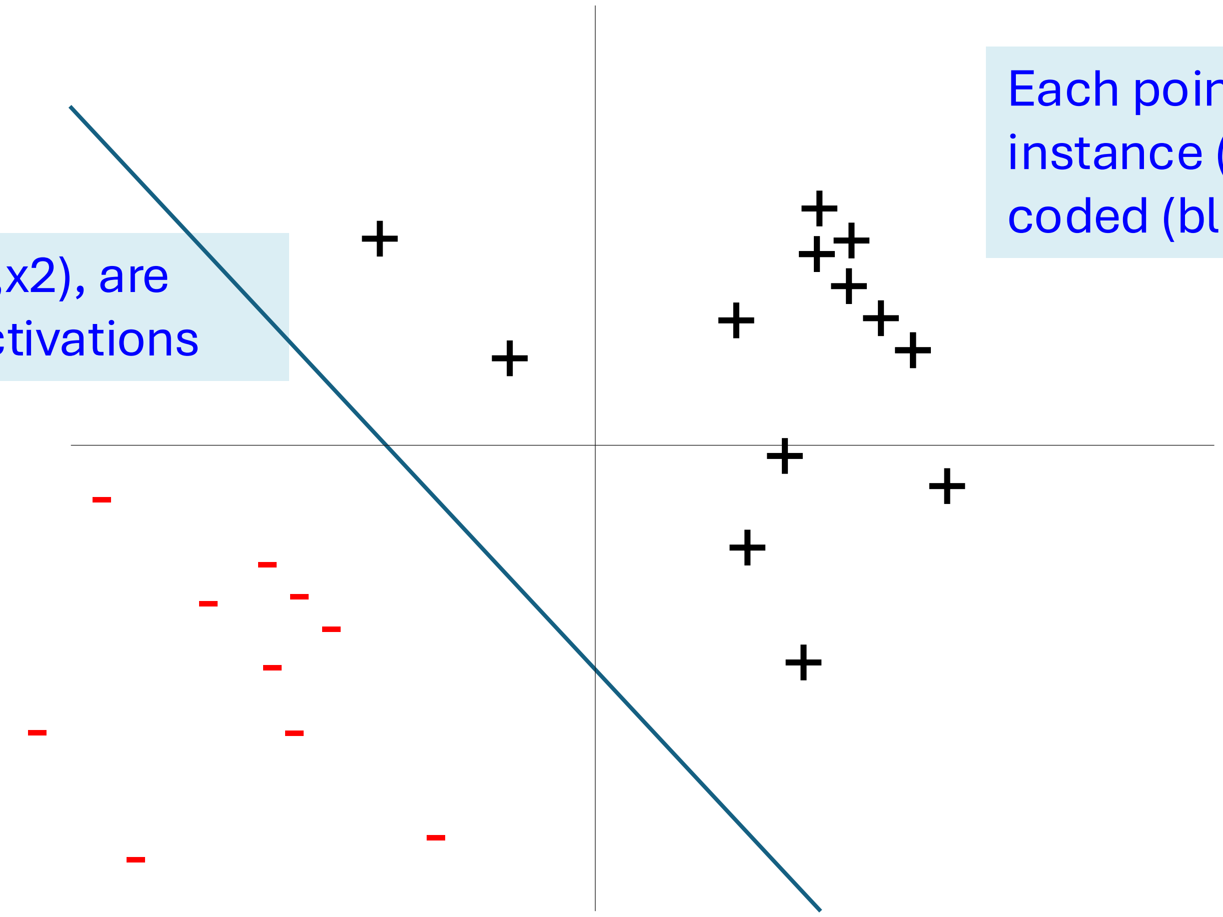
Let's take a closer look!

Linear Classifiers

- Linear threshold functions
  - Associate a weight (w_i) with each feature (x_i)
  - **Prediction:** $\text{sign}(b + w^T x) = \text{sign}(b + \Sigma\, w_i\, x_i)$
    - $b + w^T x \geq 0$ predict $y=1$
    - Otherwise, predict $y=-1$
- ***NB is a linear threshold function***
  - Weight vector is assigned by computing conditional probabilities
- *In fact, linear threshold functions are a very popular representation!*

Linear Classifiers

$$\mathrm{sign}(b + w^T x) = \mathrm{sign}\,(b + \Sigma\, w_i\, x_i)$$

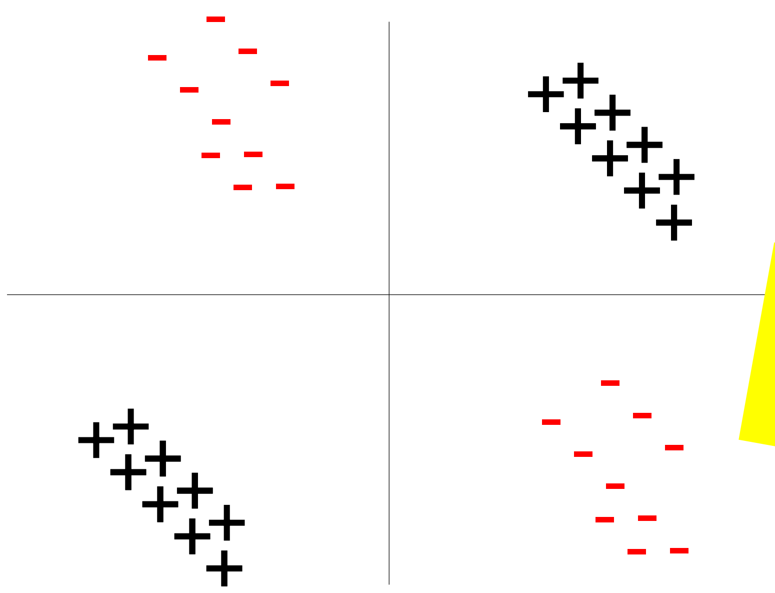


The coordinates (e.g., x1,x2), are determined by feature activations

Each point in this space is an instance (x), the label color coded (black,red)

Expressivity

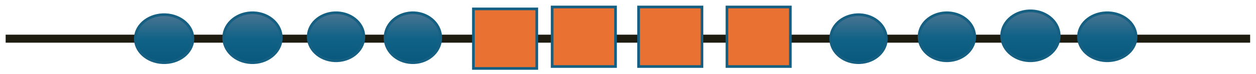
- How expressive are linear functions, i.e., is there *a linear function that is consistent with the data*
- **A famous non-linearly separable example (XOR):**



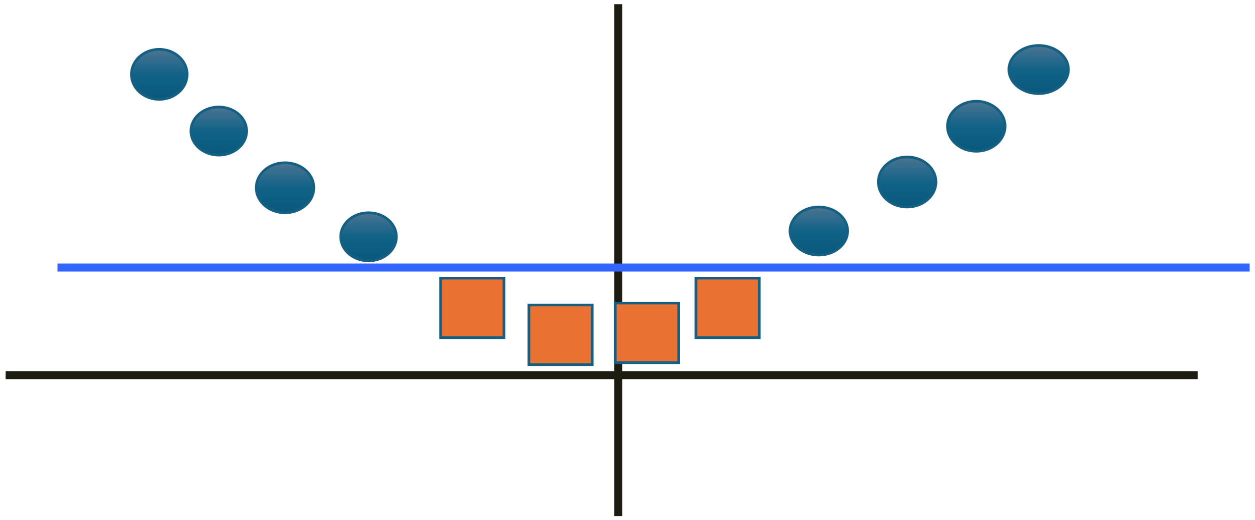
Is this a "deal breaker"?

Expressivity

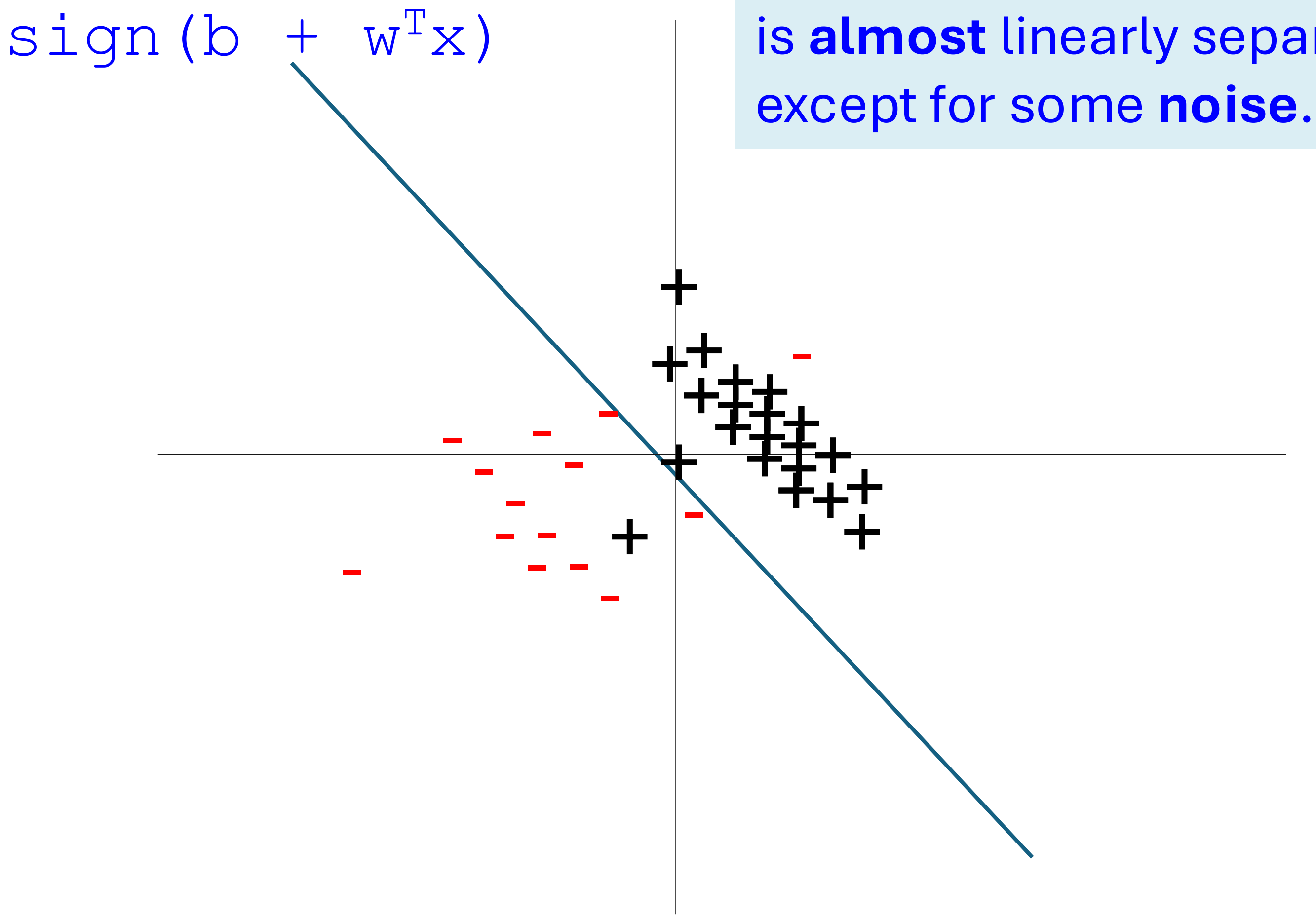
By transforming the feature space these functions can be made linear



Represent each point in 2D as (x,x^2)



Expressivity



More realistic scenario: the data is **almost** linearly separable, except for some **noise**.

Features

- Low expressivity models depends on having a **relevant** feature representation.
  - *Relevant: can characterize the target function without blowing up the space*
- We can design complex features

$$\phi_1(x) = \begin{cases} 1 & x_1 \ \textit{is capitalized} \\ 0 & \textit{otherwise} \end{cases}$$

$$\phi_k(x) = \begin{cases} 1 & x \ \textit{contains "good" more than twice} \\ 0 & \textit{otherwise} \end{cases}$$

Feature choices

- So far we have discussed BoW representation
  - *In fact, you can use a very rich representation*
- Example: identifying names

using capitalization patterns

$$\phi_1(x) = \begin{cases} 1 & x_1 \;\; is \;\; capitalized \\ 0 & otherwise \end{cases}$$

- *Question: what are good features for..*
  - *Deception detection?*
  - *Identifying status on social media?*
  - *Predicting ”virality”?*
  - *Identifying emergency situations?*
- *Can/should we go beyond word representations?*

Perceptron

- One of the earliest learning algorithms
  - Introduced by Rosenblatt 1958 to model neural learning
- **Goal**: directly search for a separating hyperplane
  - If one exists, perceptron will find it
  - If not, …
- **Online** algorithm
  - Considers one example at a time (NB – looks at entire data)
- **Error driven** algorithm
  - Updates the weights only when a mistake is made

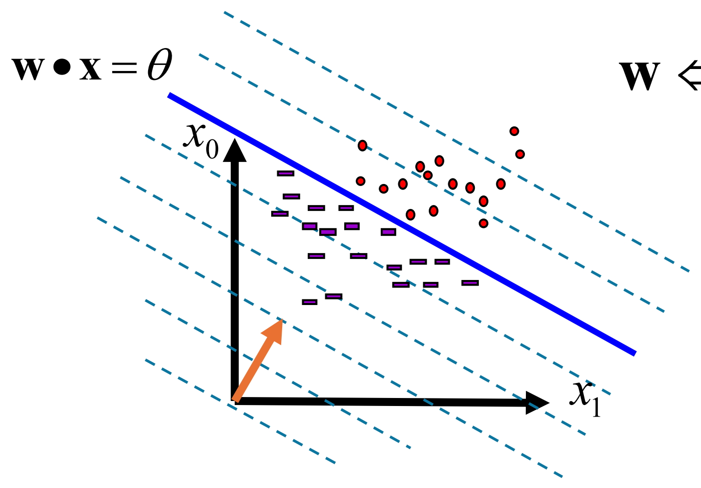
Perceptron

- We learn f:X→ {-1,+1} represented as **f =sign{w•x)**
- Where X= $\{0,1\}^n$
- Given Labeled examples: $\{(x_1, y_1), (x_2, y_2),\ldots(x_m, y_m)\}$

```
1. Initialize w=0 $\in \mathbf{R}^n$
2. Cycle through all examples, until no more errors are made
    a. Predict the label of instance x to be y' = sgn{w•x)
    b. If y'≠y, update the weight vector:
        w = w + r y x    (r - a constant, learning rate)
        Otherwise, if y'=y, leave weights unchanged.
```

Note about threshold

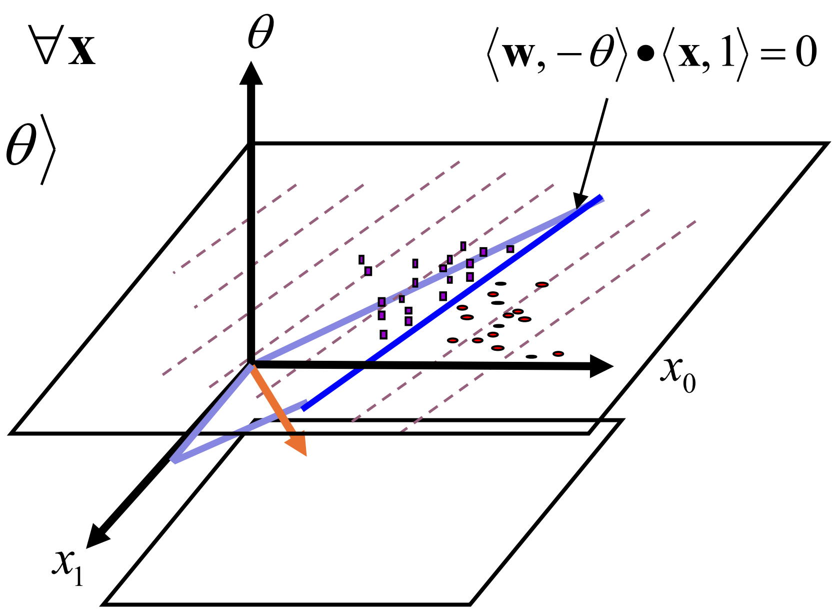
- On previous slide, Perceptron has no threshold
- But we don't lose generality:



$$\mathbf{x} \Leftrightarrow \langle \mathbf{x}, 1 \rangle \quad \forall \mathbf{x}$$

$$\mathbf{w} \Leftrightarrow \langle \mathbf{w}, -\theta \rangle$$

$$\Leftrightarrow$$


θ
$\langle \mathbf{w}, -\theta \rangle \bullet \langle \mathbf{x}, 1 \rangle = 0$
x_0
x_1

Perceptron Convergence

- The Perceptron algorithm defines a search procedure over the space of linear functions.
  - ***When do we move from one search-state to another?***
- The stopping criteria of the perceptron algorithm is "*no more mistakes over the training data*"
- **What kind of convergence guarantees can we get?**
  - We analyze Perceptron in terms of the **number of mistakes** the algorithm will make until convergence.
    - *Note that it is different from number of iterations of the data!*
- This analysis will rely on the notion of **margin**

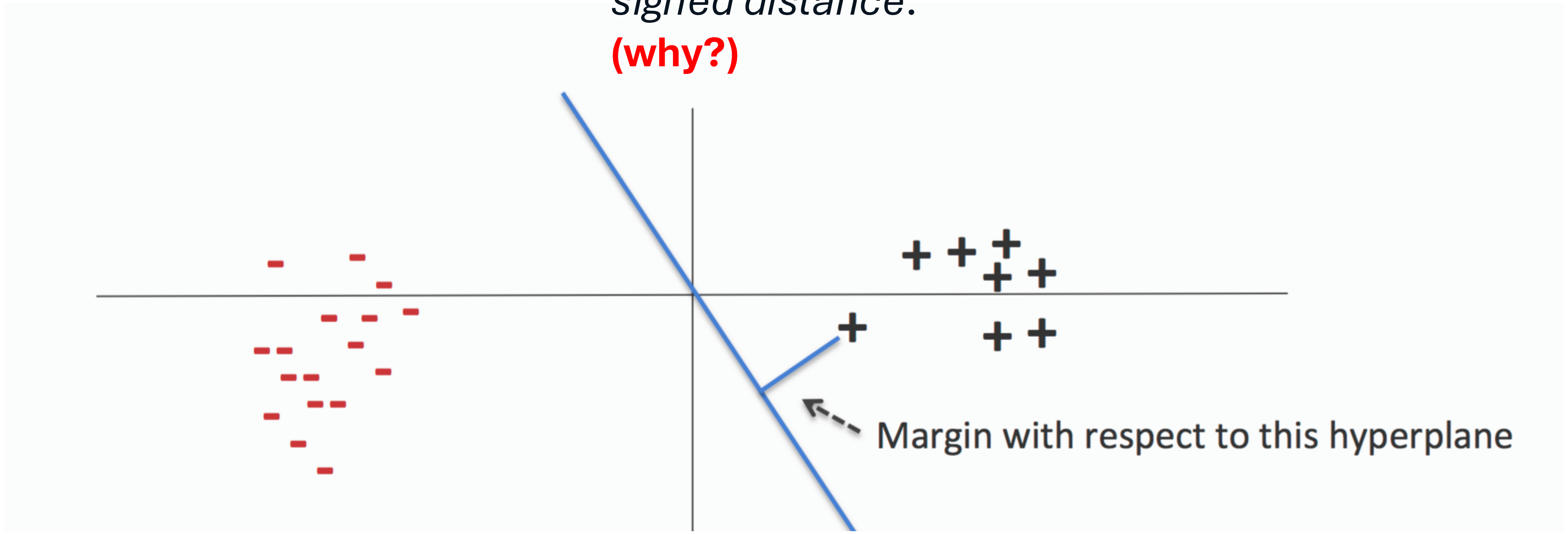
Margin

- The margin of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.

Distance: $\frac{|\mathbf{wx}+b|}{\|\mathbf{w}\|}$

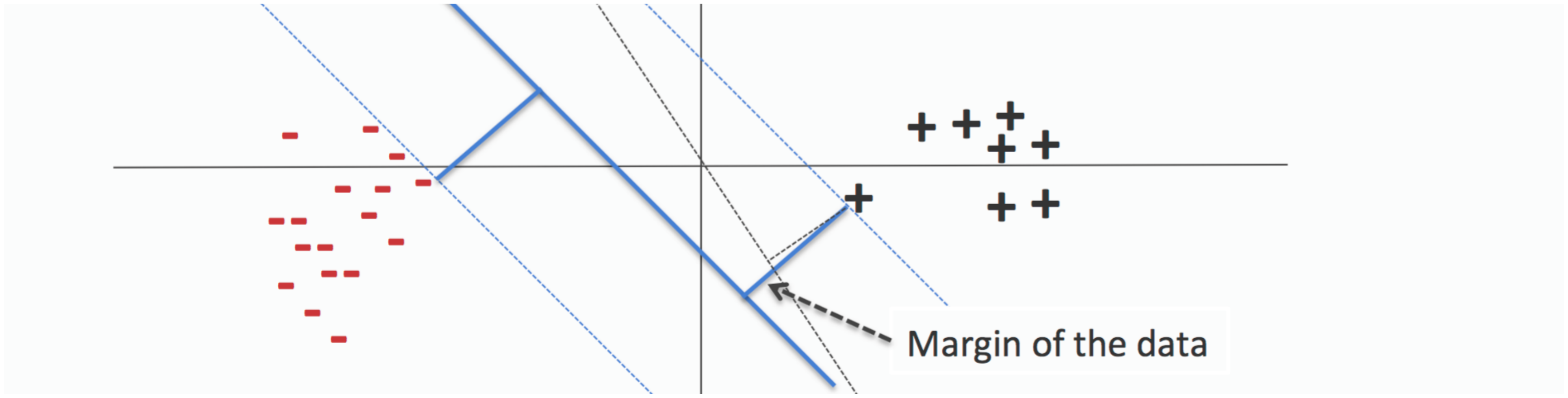
To help the discussion, let's consider the signed distance: **(why?)**

$$\frac{(\mathbf{wx_i}+b)\,y_i}{\|\mathbf{w}\|}$$



Margin

- The margin of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.
- The margin of a data set (γ) is the maximum margin possible for that dataset using any weight vector.



Mistake Bound for Perceptron

- Let D={(x_i, y_i)} be a labeled dataset that is separable
- Let $||x_i||< R$ for all examples.
- Let γ be the margin of the dataset D.
- Then, the perceptron algorithm will make at most R^2/γ^2 mistakes on the data.

Perceptron: Realistic Version

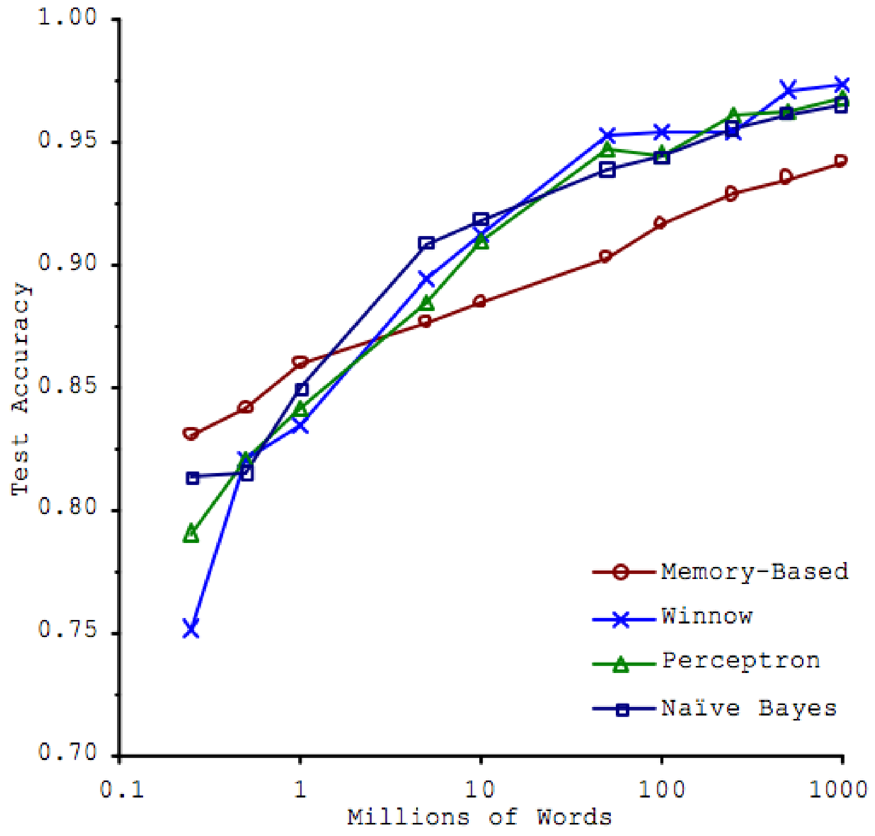
- We learn **f:X→ {-1,+1}** represented as **f =sgn{w•x)**
- Where **X={0,1}^{n}**
- Given Labeled examples: **{ (x_1, y_1) , (x_2, y_2) ,… (x_m, y_m) }**

```
1.  Initialize w=0 ∈ R^n
2.  For Iter = 0,…,T
        3. Iterate over all the examples
            a. Predict the label of instance x to be y' = sgn{w•x)
            b. If y'≠y, update the weight vector:
              w = w + r y x      (r - a constant, learning rate)
              Otherwise, if y'=y, leave weights unchanged.
```

Practical Example

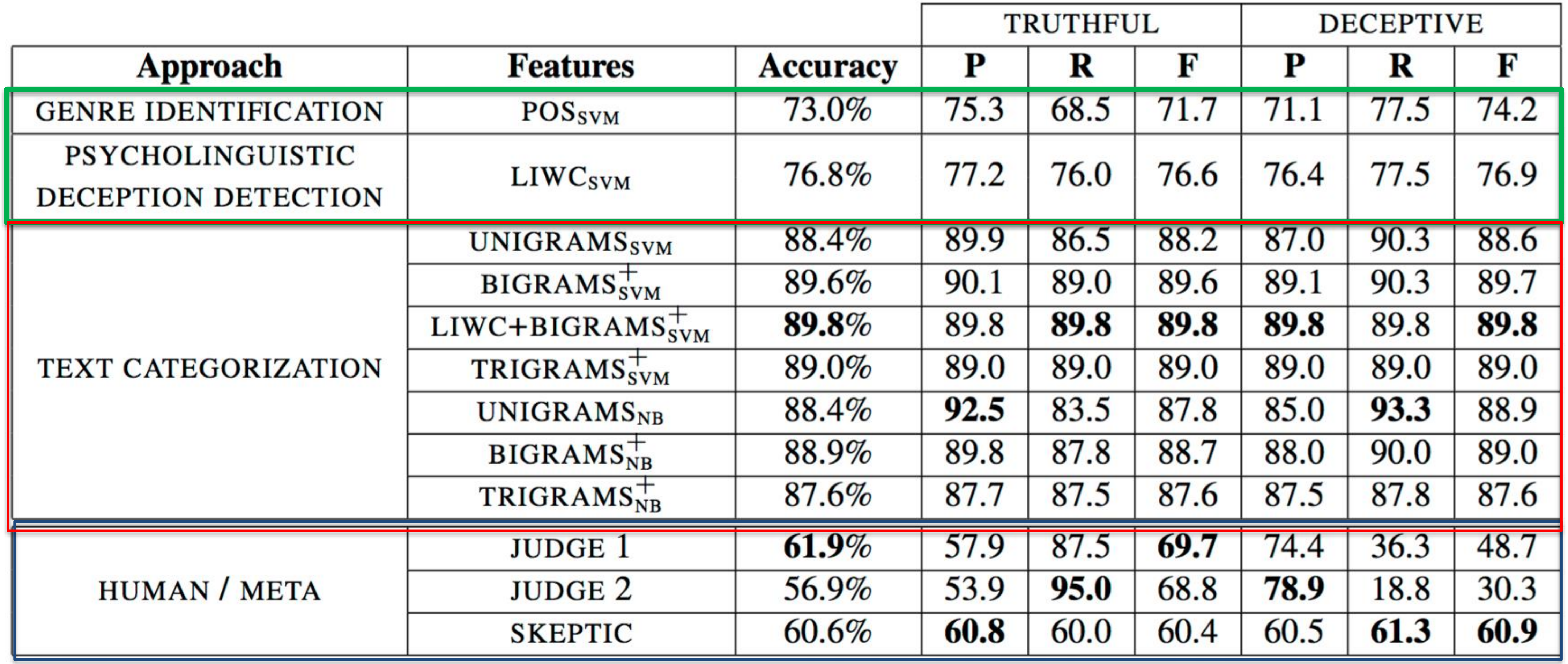
*Task: **context sensitive spelling***

“I didn’t know {weather,whether} to laugh or cry”



Source: Scaling to very very large corpora for natural language disambiguation Michele Banko, Eric Brill. MSR, 2001.

Case Study 1: Deception Detection



			TRUTHFUL			DECEPTIVE		
Approach	**Features**	**Accuracy**	**P**	**R**	**F**	**P**	**R**	**F**
GENRE IDENTIFICATION	POS_{SVM}	73.0%	75.3	68.5	71.7	71.1	77.5	74.2
PSYCHOLINGUISTIC DECEPTION DETECTION	LIWC_{SVM}	76.8%	77.2	76.0	76.6	76.4	77.5	76.9
TEXT CATEGORIZATION	$\text{UNIGRAMS}_{\text{SVM}}$	88.4%	89.9	86.5	88.2	87.0	90.3	88.6
	$\text{BIGRAMS}^{+}_{\text{SVM}}$	89.6%	90.1	89.0	89.6	89.1	90.3	89.7
	$\text{LIWC+BIGRAMS}^{+}_{\text{SVM}}$	**89.8%**	89.8	**89.8**	**89.8**	**89.8**	89.8	**89.8**
	$\text{TRIGRAMS}^{+}_{\text{SVM}}$	89.0%	89.0	89.0	89.0	89.0	89.0	89.0
	$\text{UNIGRAMS}_{\text{NB}}$	88.4%	**92.5**	83.5	87.8	85.0	**93.3**	88.9
	$\text{BIGRAMS}^{+}_{\text{NB}}$	88.9%	89.8	87.8	88.7	88.0	90.0	89.0
	$\text{TRIGRAMS}^{+}_{\text{NB}}$	87.6%	87.7	87.5	87.6	87.5	87.8	87.6
HUMAN / META	JUDGE 1	**61.9%**	57.9	87.5	**69.7**	74.4	36.3	48.7
	JUDGE 2	56.9%	53.9	**95.0**	68.8	**78.9**	18.8	30.3
	SKEPTIC	60.6%	**60.8**	60.0	60.4	60.5	**61.3**	**60.9**

Case Study 2: Detecting Social Meaning

- Credits:

Slides based on D. Jurafsky slides.

Describes work done in papers:

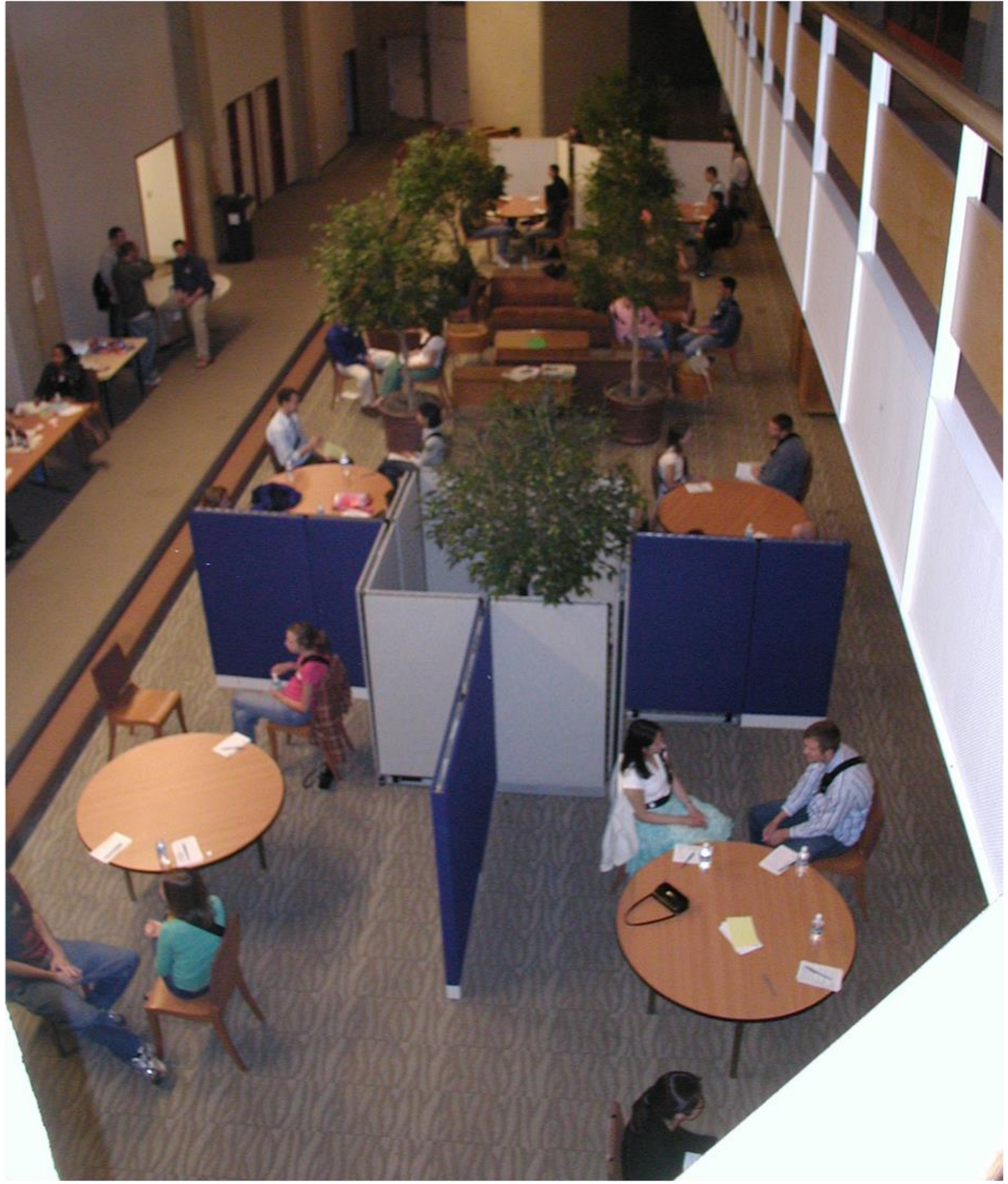
Dan Jurafsky, Rajesh Ranganath, and Dan McFarland. 2009. **Extracting Social Meaning: Identifying Interactional Style in Spoken Conversation**. Proceedings of NAACL HLT 2009.

Rajesh Ranganath, Dan Jurafsky, and Dan McFarland. 2009. **It's Not You, it's Me: Detecting Flirting and its Misperception in Speed-Dates**. EMNLP-2009

Detecting social meaning:

- Given speech and text from a conversation
- Can we detect `styles', like whether a speaker is
  - **Awkward?**
  - **Flirtatious?**
  - **Friendly?**
- Can we tell if the speakers like each other?
- Dataset:
  - 991 4-minute "speed-dates"
  - Each participant rated their partner and themselves for these styles

speed
date
setup



What do you do for fun? Dance?

Uh, dance, uh, I like to go, like camping. Uh, snowboarding, but I'm not good, but I like to go anyway.

You like boarding.

Yeah. I like to do anything. Like I, I'm up for anything.

Really?

Yeah.

Are you open-minded about most everything?

Not everything, but a lot of stuff-

What is not everything [laugh]

I don't know. Think of something, and I'll say if I do it or not. [laugh]

Okay. [unintelligible].

Skydiving. I wouldn't do skydiving I don't think.

Yeah I'm afraid of heights.

F: Yeah, yeah, me too.

M: [laugh] Are you afraid of heights?

F: [laugh] Yeah [laugh]

The Speed Date corpus

- **991 4-minute dates**
  - 3 events, each with ~20x20=400 dates, some data loss
  - Participants: graduate student volunteers in 2005
    - participated in return for the chance to date
- **Speech**
  - ~60 hours, from shoulder sash recorders; high noise
- **Transcripts**
  - ~800K words, hand-transcribed, w/turn boundary times
- **Surveys**
  - (Pre-test surveys, event scorecards, post-test surveys)
  - Date perceptions and follow-up interest
  - General attitudes, preferences, demographics
- Largest experiment with audio, text, + survey info

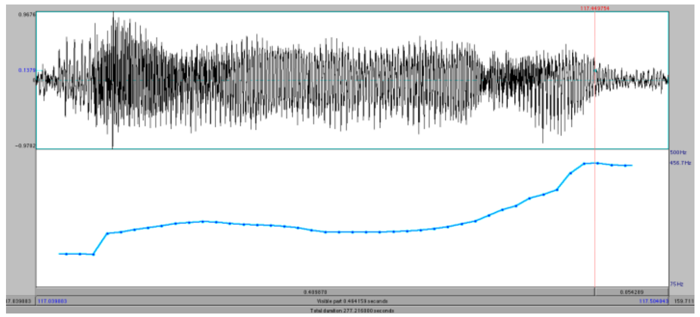
What we attempted to predict

- **Conversational style:**
  - *How often did **you** behave in the following ways on this date?*
  - *How often did **they** behave in the following ways on this date?*
    - On a scale of 1-10 (1=never, 10=constantly)

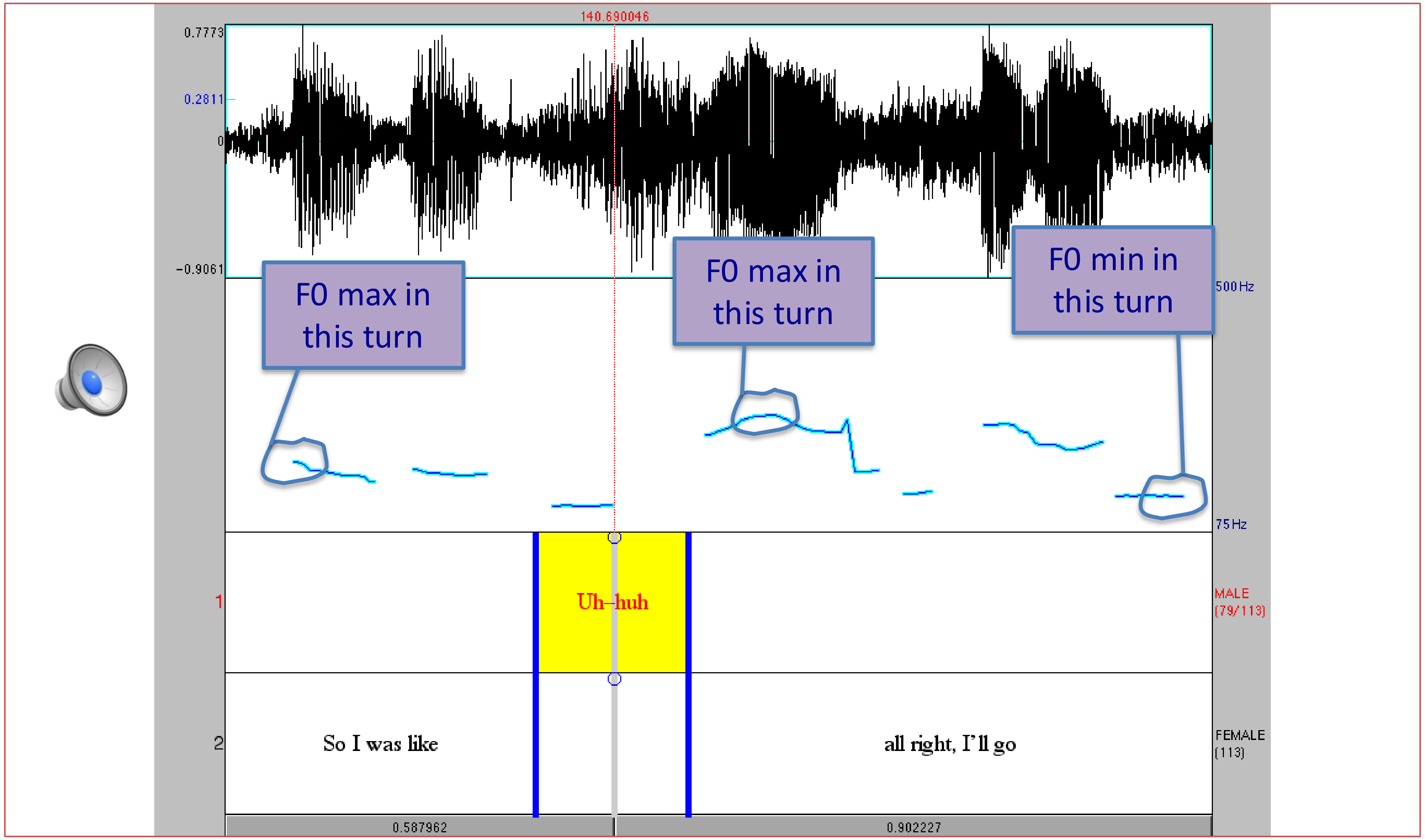
1. **flirtatious**
2. **friendly**
3. **awkward**
4. assertive

Features

- **Prosodic**
  - pitch (min, mean, max, std)
  - intensity (min, max, mean, std)
  - duration of turn
  - rate of speech (words per second)
- **Dialog**
  - questions
  - backchannels (*"uh-huh", "yeah"*)
  - appreciations (*"Wow!", "That's great!")*
- **Lexical**
  - negative emotion (*bad, weird, crazy, hate*) words
  - storytelling words (past tense) + food words (*eat, dinner*)
  - love and sexual/emotional words (*love, passionate, screw*)
  - personal pronouns (*I, you, we, us*)



Features extracted within turns



Features: Pitch

- F0 min, max, mean
  - Thus to compute, e.g., F0 min for a conversation side
    - Take F0 min of each turn (not counting zero values)
    - Average over all turns in the side
    - "F0 min, F0 max, F0 mean"
  - We also compute measures of **variation**
    - Standard deviation, pitch range
    - F0 min sd, F0 max sd, F0 mean sd
    - pitch range = (f0 max – f0 min)

Features: other prosodic

- Intensity min, max, mean, std
  - computed as for pitch
- Duration of turn
- Total time for conversation side
- Rate of speech (words per second)

Dialog act features

- **Questions** # of questions in side
- **Laughter** # of instances of laughter in side
- **Turns** total # of turns in a side
- **Backchannels** # of backchannels in side
  - *Uh-huh. Yeah. Right. Oh, okay.*
- **Appreciations** # of appreciations in side
  - *Wow. That's true. Oh, great! Oh, gosh!*
- *Regular expressions drawn from hand-labeled Switchboard Dialogue Act Corpus (Jurafsky, Biasca, Shriberg 1997)*

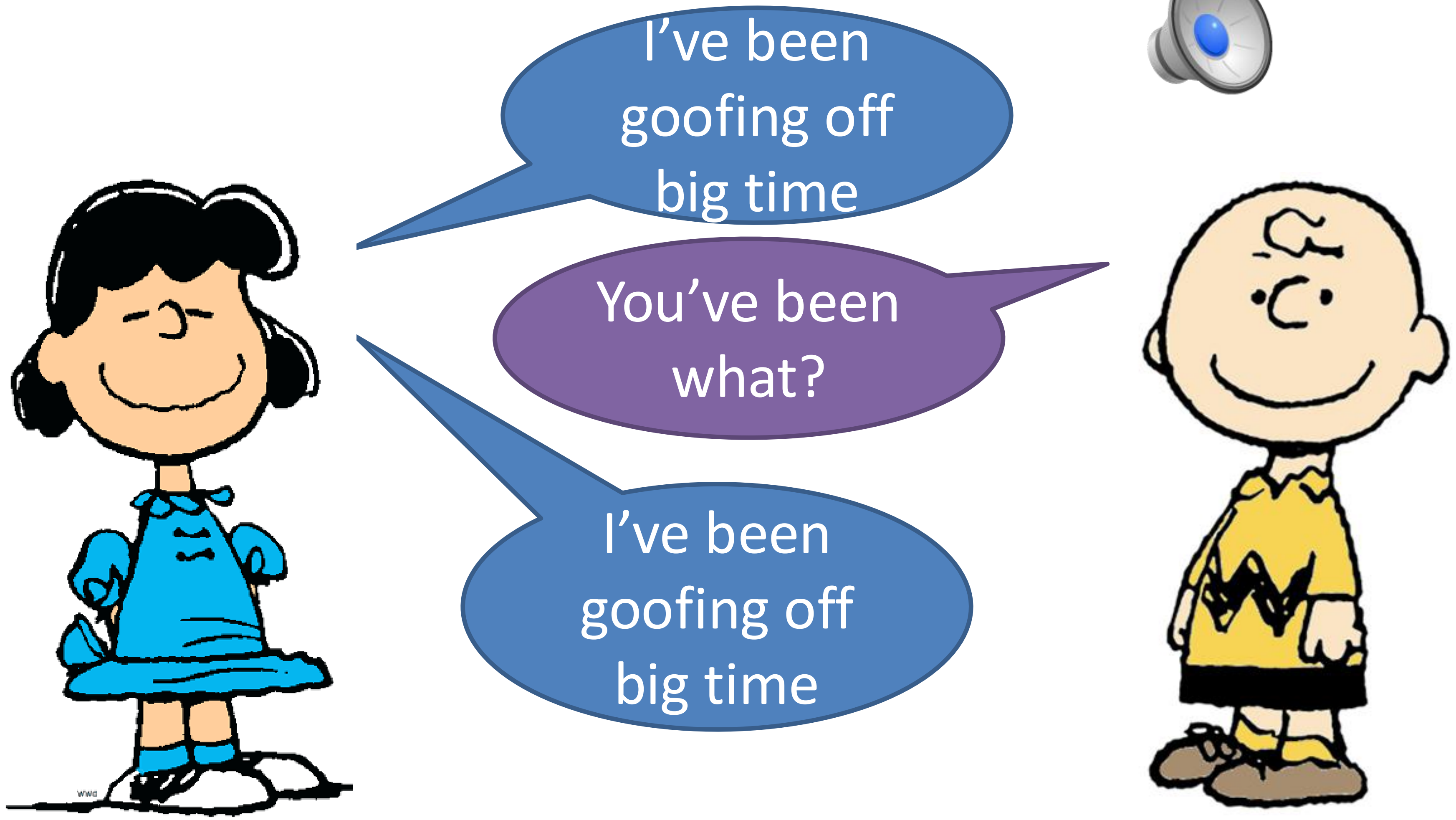
Appreciations

- Wow.
- Oh, wow.
- That's great.
- That's good.
- That's right.
- Oh, no.
- Oh, my goodness.
- That's true.
- Well, that's good.
- Oh, that's great.
- Oh, gosh.
- Great.

Backchannels

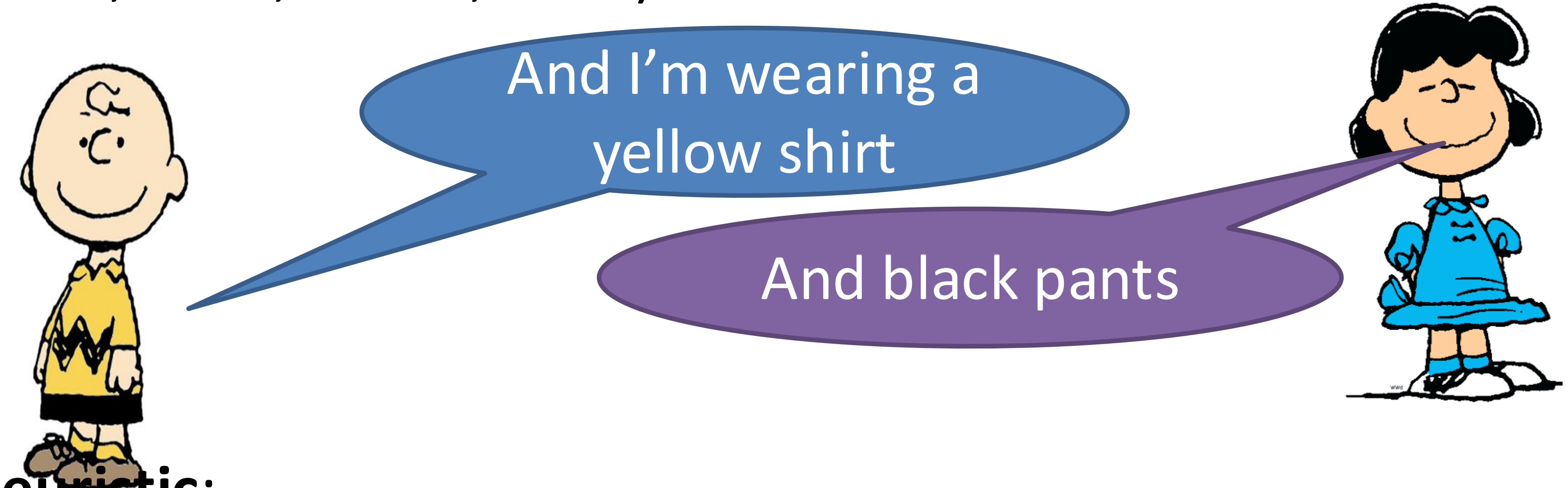
- Uh-huh
- Yeah
- Right
- Oh
- Yes
- Huh
- Oh, yeah
- Okay
- Sure
- Really
- Oh, really

Clatifications


I’ve been goofing off big time
You’ve been what?
I’ve been goofing off big time

Collaborative Completion

- a turn where a speaker completes the utterance begun by the alter (Lerner, 1991; Lerner, 1996).



Heuristic:

- first word of *sentence*$_i$ is predictable from last two words of *sentence*$_{i-1}$ (using a trigram grammar trained on Switchboard)

Dialog feature: Collaborative Completion

- **Heuristic**: first word of $sentence_i$ is predictable from last two words of $sentence_{i-1}$
- **Result**: Tends to find "locally coherent phrasal answers"
    - M: What year did you graduate?
    - F: From high school?
    - F: What department are you in?
    - M: The business school.
  - But not:
    - F: What department are you in?
    - M: I'm in the teacher education program.

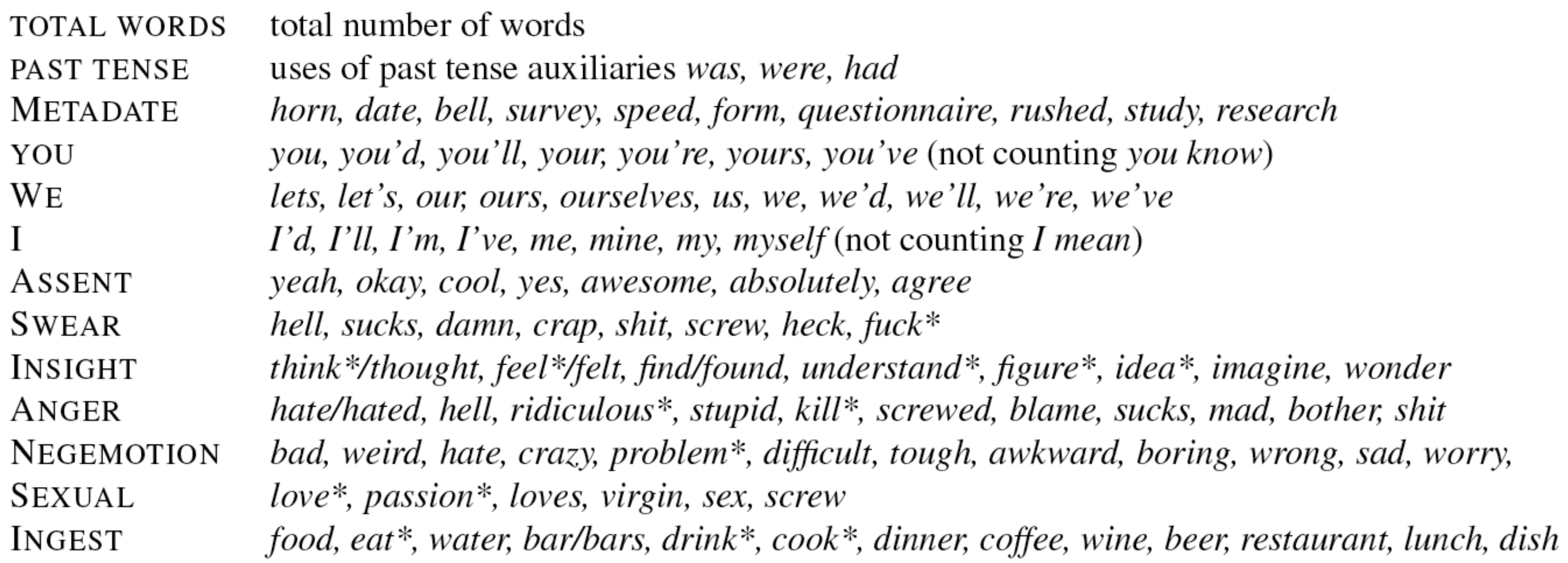
Disfluency Features

- **UH/UM**: # of filled pauses (uh or um) in side
  - M: Um, eventually, yeah, but right now I want to get some more experience, uh, in research.
  - F: Oh.
  - M: Uh, so I will probably work for, uh, a research lab for, uh, big companies.
- **RESTART**: # of disfluent restarts in side
  - Uh, I–there's a group of us that came in–
- **OVERLAP**: # of turns in side where speakers overlapped
  - M: But-and also obviously–
    - F: It sounds bigger.
  - M: –people in the CS school are not quite as social in general as other–

LIWC

- Linguistic Inquiry and Word Count
  - Pennebaker, Francis, & Booth, 2001
- dictionary of 2300 words grouped into > 70 classes
  - **negative emotion** (bad, weird, hate, problem, tough)
  - **sexual** (love, loves, lover, passion, passionate, sex,)
  - **1st person pronouns** (I me mine myself I'd I'll I'm…)
  - **2nd person pronouns** (you, you'd you'll your you've…)
  - **ingest** (food, eat, eats, cook, dinner, drink, restaurant…)
  - **swear** (hell, sucks, damn, fuck,…)
  - …
- after 9/11
  - greater negative emotion
  - more socially engaged

Lexical Features



TOTAL WORDS	total number of words
PAST TENSE	uses of past tense auxiliaries *was, were, had*
METADATE	*horn, date, bell, survey, speed, form, questionnaire, rushed, study, research*
YOU	*you, you'd, you'll, your, you're, yours, you've* (not counting *you know*)
WE	*lets, let's, our, ours, ourselves, us, we, we'd, we'll, we're, we've*
I	*I'd, I'll, I'm, I've, me, mine, my, myself* (not counting *I mean*)
ASSENT	*yeah, okay, cool, yes, awesome, absolutely, agree*
SWEAR	*hell, sucks, damn, crap, shit, screw, heck, fuck**
INSIGHT	*think*/thought, feel*/felt, find/found, understand*, figure*, idea*, imagine, wonder*
ANGER	*hate/hated, hell, ridiculous*, stupid, kill*, screwed, blame, sucks, mad, bother, shit*
NEGEMOTION	*bad, weird, hate, crazy, problem*, difficult, tough, awkward, boring, wrong, sad, worry,*
SEXUAL	*love*, passion*, loves, virgin, sex, screw*
INGEST	*food, eat*, water, bar/bars, drink*, cook*, dinner, coffee, wine, beer, restaurant, lunch, dish*

Architecture: 6 binary classifiers

  - **Female ±Awkward, Male ±Awkward,**
  - **Female ±Friendly, Male ±Friendly,**
  - **Female ±Flirtatious, Male ±Flirtatious,**

- **Multiple classifier experiments**
  - L1-regularized logistic regression
  - SVM w/RBF kernel
  - 5-fold cross-validation
    - tested on held-out test set of 10% highest and 10% lowest
    - 5 folds: 3 train, 1 validation, 1 test

Experiments

- K-fold cross validation.
- 5 folds: 3 train, 1 validation, 1 test
- Randomized the data ordering, repeated k-fold cross validation 25 times.

- Feature weights (θ)
  - We calculated a separate θ for each randomized run.
  - Resulting in a vector of weights for each feature.
  - We kept any features if the median of its weight vector was non-zero

Results with SVM: predicting flirt intention

- Using my speech to predict whether I say I am flirting

	Male speaker	Female speaker
I say I'm flirting	72%	76%

Results with SVM: Predicting flirt perception

- Using my speech to predict whether partner says I am flirting

	Male speaker	**Female speaker**
Partner says I'm flirting	80%	68%

Summary: flirt detection

- Using my speech to predict whether I am flirting

	Male speaker	Female speaker
I say I'm flirting	72%	76%
Partner says I'm flirting	80%	68%

Detecting awkward and friendly speakers

- Using what I do & what my date does to predict what my date calls me
- Simpler (logistic regression) classifier

	Awkward		Friendly	
	M	F	M	F
Using speaker words/speech	63%	51	72	68
+ partner words/speech	**64**	**64**	**73**	**75**

What makes someone seem friendly?
"Collaborative conversational style"

- Related to the "collaborative floor" of Edelsky (1981), Coates (1996)
  - Collaborative completions (Lerner 1991, 1996)
    - M: And I'm wearing a green shirt.
    - F: And blue pants.
  - Clarifications
    - *F: I'm working at Pottery Barn this summer.*
    - *M: I'm sorry, who?*
  - Other questions
  - You
  - Laughter
  - Plus perhaps
    - Appreciations (for women)
    - Overlaps (for men)

What makes someone seem friendly? "Collaborative conversational style"

- Related to the "collaborative floor" of Edelsky (1981), Coates (1996)
  - Collaborative completions (Lerner 1991, 1996)
    - M: And I'm wearing a green shirt.
    - F: And blue pants.
  - Clarifications
    - *F: I'm working at Pottery Barn this summer.*
    - *M: I'm sorry, who?*
  - Other questions
  - You
  - Laughter
  - Plus perhaps
    - Appreciations (for women)
    - Overlaps (for men)

Based on results analysis, which features we the most predictive

Conclusions – for daters

- Talking about your advisor is a bad idea on a date
- Sympathy is a good idea, if you're a guy
- Passion is good, if you're a woman
- Food is good, if you eat

Conclusions – for computer science

- We can do automatic extraction of rich social variables from speech and text.
- For at least this variable ("does speaker intend to flirt") we beat human performance