
Machine Learning Approaches to NLP

Part I

Sameer Maskey

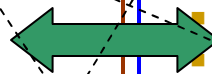
NLP

Many Topics Related
Tasks \leftrightarrow Potential Solutions

ML

- Morphology (including word segmentation)
- Part of speech tagging
- Syntax and parsing
- Grammar Engineering
- Word sense disambiguation
- Lexical semantics
- Mathematical Linguistics
- Textual entailment and paraphrasing
- Discourse and pragmatics
- Knowledge acquisition and representation
- Noisy data analysis
- Machine translation
- Multilingual language processing
- Language generation
- Summarization
- Question answering
- Information retrieval
- Information extraction
- Topic classification and information filtering
- Non-topical classification (sentiment/genre analysis)
- Topic clustering
- Text and speech mining
- Text classification
- Evaluation (e.g., intrinsic, extrinsic, user studies)
- Development of language resources
- Rich transcription (automatic annotation)
- ...

- Reinforcement Learning
- Online Learning
- Ranking
- Graphs and Embedding
- Gaussian Processes
- Dynamical Systems
- Kernels
- Codebook and Dictionaries
- Clustering Algorithms
- Structured Learning
- Topic Models
- Transfer Learning
- Weak Supervision
- Learning Structures
- Sequential Stochastic Models
- Active Learning
- Support Vector Machines
- Boosting
- Learning Kernels
- Information Theory and Estimation
- Bayesian Analysis
- Regression Methods
- Inference Algorithms
- Analyzing Networks & Learning with Graphs
- ...



Text Mining

- Data Mining: finding nontrivial patterns in databases that may be previously unknown and could be useful
- Text Mining:
 - Find interesting patterns/information from unstructured text
 - Discover new knowledge from these patterns/information
- Information Extraction, Summarization, Opinion Analysis, etc can be thought as some form of text mining
- Let us look at an example


Patterns in Unstructured Text






Rate This Item to Improve Your Recommendations

I own it  Rate this item

Customer Reviews

Average Customer Rating

 (585 customer reviews)


5 star:  (460)
4 star:  (86)
3 star:  (13)
2 star:  (11)
1 star:  (15)


[Ease of use](#)  (112)
[Image quality](#)  (112)
[Construction quality](#)  (110)
[Battery life](#)  (109)
> [See and rate all 14 attributes.](#)

All Amazon reviewers may not rate the product, may just write reviews, we may have to infer the rating based on text review

Most Helpful Customer Reviews

1,568 of 1,594 people found the following review helpful:

 **Great camera, one of the best low(er)-end DSLRs on the market,** April 23, 2008

By [Hyun Yu](#)  - [See all my reviews](#)

[TOP 1000 REVIEWER](#) [REAL NAME](#) [VINE™ VOICE](#)

My journey with DSLRs began back in 2003 with the original Digital Rebel. DSLRs changed my photography for the better like nothing else. Five years and some 25,000 shots later, it's still going strong. Along the way I upgraded to the Canon 30D, which is a fantastic camera as well. When the 40D was announced I decided to wait until the 50D sometime in 2009, but wanted a newer backup/second body for my photography needs. So when the XSi/450D was announced, it sounded like a perfect fit for my needs.

I got it from Amazon.com three days ago, and have given it a pretty good workout since then, having shot about 650 shots under a variety of shooting conditions and with a number of different Canon and third-party lenses. The following are my impressions.

The build feels very good. The camera feels wonderfully light yet well built. I'm 6ft tall with average size hands, and the camera feels good in my hand. The battery grip, to me, defeats the purpose of having a small, light DSLR, so I opted for a Hakuba/Opteka grip (it's a plate that screws into the tripod socket that enables you to use the excellent Canon E1 hand strap with it) and I couldn't be happier. I'm not a fan of neck straps, so this works well for me (see the uploaded photo for the configuration).

Some of these patterns could be exploited to discover knowledge

Patterns may exist in unstructured text

Review of a camera in Amazon

Text to Knowledge

- Text

- Words, Reviews, News Stories, Sentences, Corpus, Text Databases, Real-time text, Books



Many methods to use
for discovering
knowledge from text

- Knowledge

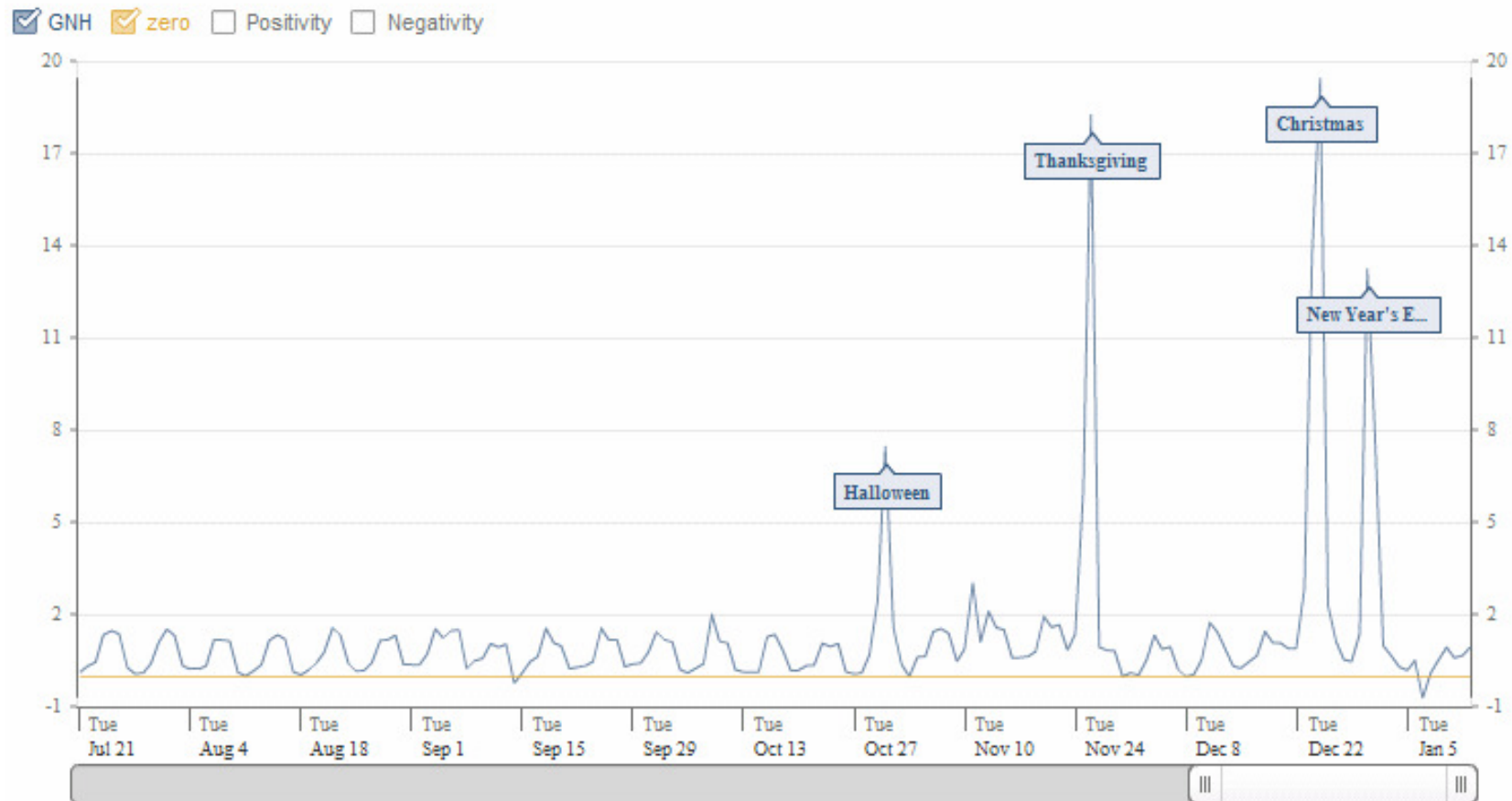
- Ratings, Significance, Patterns, Scores, Relations

Unstructured Text → Score

Facebook's "Gross National Happiness Index"

- Facebook users update their status
 - "...is writing a paper"
 - "... has flu ☹️"
 - "... is happy, yankees won!"
- Facebook updates are unstructured text
- Scientists collected all updates and analyzed them to predict "Gross National Happiness Index"

Facebook's "Gross National Happiness Index"



How do you think they extracted this **SCORE** from a **TEXT** collection of status updates?

Facebook Blog Explains

- “The result was an index that measures how happy people on Facebook are from day-to-day by looking at the number of positive and negative words they're using when updating their status. When people in their status updates use more positive words - or fewer negative words - then that day as a whole is counted as happier than usual.”

**Looks like they are COUNTING!
+ve and -ve words in status updates**

Let's Build Our ML Model to Predict Happiness 😊

- Simple Happiness Score
 - Our simpler version of happiness index compared to facebook
 - Score ranges from 0 to 10
- There are a few things we need to consider
 - We are using status updates
 - We do not know what words are positive and negative
 - We do not have any training data

Our Prediction Problem

■ Training data

- Assume we have $N=100,000$ status updates
- Assume we have a simple list of positive and negative words
- Let us also assume we asked a human annotator to read each of the 100,000 status update and give a happiness Score (Y_i) between 0 to 10
 - "...is writing a paper" ($Y_1 = 4$)
 - "... has flu ☹" ($Y_2 = 1.8$)
 - .
 - .
 - .
 - "... is happy, game was good!" ($Y_{100,000} = 8.9$)

■ Test data

- "... likes the weather" ($Y_{100,001} = ?$)

Given labeled set of 100K Status updates, how do we build Statistical/ML model that will predict the score for a new status update

Features to Represent Text

- "...is writing a paper" (Y1 = 4)
- "... has flu ☹" (Y2 = 1.8)
- .
- .
- .
- "... is happy, game was good!"
(Y100,000 = 8.9)



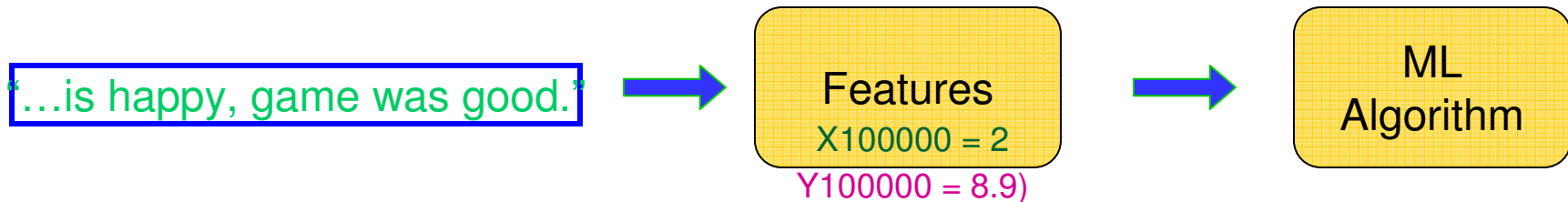
Features



ML
Algorithm

Representing Text of Status Updates As a Feature Vector

- What kind of feature can we come up with that would relate well with happiness score
- How about represent status update as
 - Count (+ve words in the sentence) (not the ideal representation, will talk about better representation later)
 - For the 100,000th sentence in our previous example:
 - "...is happy, game was good." Count is 2
 - Status Update 100,000th is represented by
 - ($X_{100000} = 2, Y_{100000} = 8.9$)



Modeling Technique

- We want to predict happiness score (Y_i) for a new status update
- If we can model our training data with a statistical/ML model, we can do such prediction

- (1, 4)

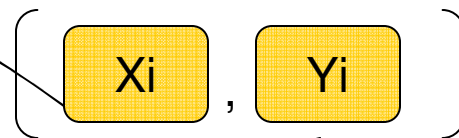
- (0, 1.8)

- .

- .

- .

- (2, 8.9)



- What modeling technique can we use?
 - Linear Regression is one choice

Linear Regression

→ We want to find a function that given our x it would map it to y

→ One such function :

$$f(x) = \theta_0 + \theta_1 x$$

→ Different values of thetas give different functions

→ What is the best theta such that we have a function that makes least error on predictions when compared with y

For notation convenience we may use $f(x)$ dropping θ in $f_\theta(x)$ or $f(x; \theta)$ when it is understood

Predicted vs. True

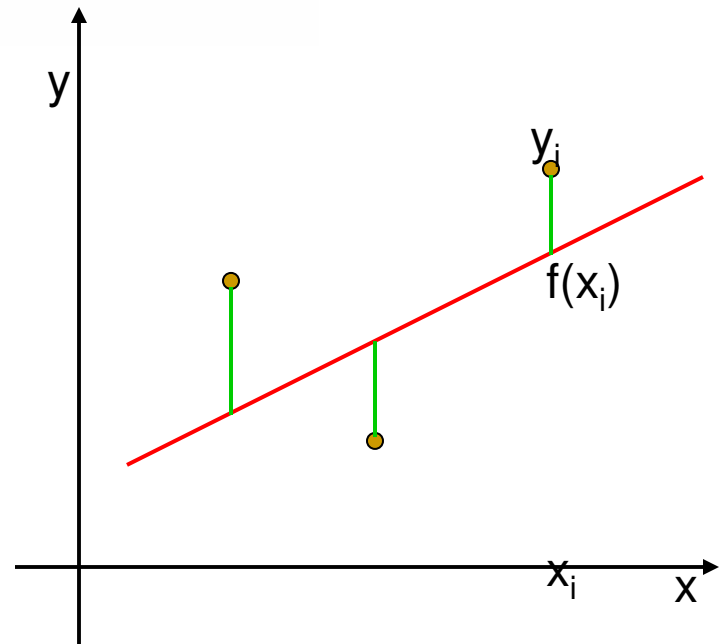
- Our function $f(x)$ approximates y
- Given a true value of y we can compute the error $f(x)$ made against true y
- For any point x_i we can compute such error by $y_i - f(x_i)$; or by squared error $\{y_i - f(x_i)\}^2$
- But we have N points so the total error/Loss L on squared error would be

$$L = \sum_{i=1}^N (y_i - f(x_i))^2$$

Sum of Squared Errors

- Plugging in $f(x)$ and averaging the error across all training data points we get the empirical loss

$$J(\theta) = \frac{1}{N} \sum_{i=1}^N (y_i - (\theta_0 + \theta_1 x_i))^2$$



Finding the Minimum

→ We can (but not always) find a minimum of a function by setting the derivative or partial derivatives to zero

→ Here we can take partials on thetas and set them to zero

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{\partial}{\partial \theta_0} \frac{1}{2N} \sum_{i=1}^N (y_i - (\theta_0 + \theta_1 x_i))^2$$

$$\frac{\partial J(\theta)}{\partial \theta_1} = \frac{\partial}{\partial \theta_1} \frac{1}{2N} \sum_{i=1}^N (y_i - (\theta_0 + \theta_1 x_i))^2$$

Solving for Weights

$$\begin{aligned}\frac{\partial J(\theta)}{\partial \theta_1} &= \frac{\partial}{\partial \theta_1} \frac{1}{2N} \sum_{i=1}^N (y_i - (\theta_0 + \theta_1 x_i))^2 \\ &= \frac{1}{2N} \sum_{i=1}^N \frac{\partial}{\partial \theta_1} (y_i - \theta_0 - \theta_1 x_i)^2 \\ &= \frac{1}{2N} \sum_{i=1}^N 2(y_i - \theta_0 - \theta_1 x_i) \frac{\partial}{\partial \theta_1} (y_i - \theta_0 - \theta_1 x_i) \\ &= \frac{1}{N} \sum_{i=1}^N (y_i - \theta_0 - \theta_1 x_i)(-x_i) = 0\end{aligned}$$

$$\begin{aligned}\frac{\partial J(\theta)}{\partial \theta_0} &= \frac{\partial}{\partial \theta_0} \frac{1}{2N} \sum_{i=1}^N (y_i - (\theta_0 + \theta_1 x_i))^2 \\ &= \frac{1}{N} \sum_{i=1}^N (y_i - \theta_0 - \theta_1 x_i)(-1) = 0\end{aligned}$$

Empirical Loss is Minimized With Given Values for the Parameters

→ Solving the previous equations we get following values for the thetas

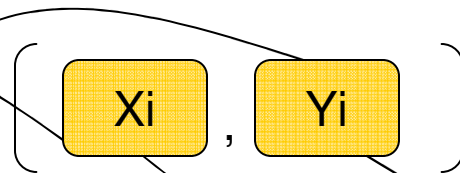
$$\theta_1 = \frac{\sum_{i=1}^N x_i y_i - \frac{1}{N} \sum_{i=1}^N x_i \sum_{i=1}^N y_i}{\sum_{i=1}^N x_i^2 - \frac{1}{N} \sum_{i=1}^N x_i \sum_{i=1}^N x_i}$$

$$\theta_0 = \frac{1}{N} \sum_{i=1}^N y_i - \frac{1}{N} \theta_1 \sum_{i=1}^N x_i$$

Implementing Simple Linear Regression

- Given our training data on status update with happiness score

- (1, 4)
- (0, 1.8)
- .
- .
- .
- .
- (2, 8.9)



Training Our Regression Model:

Just need to implement for loop that computes numerators and denominators in equations here. And we get optimal thetas

$$\theta_1 = \frac{\sum_{i=1}^N x_i y_i - \frac{1}{N} \sum_{i=1}^N x_i \sum_{i=1}^N y_i}{\sum_{i=1}^N x_i^2 - \frac{1}{N} \sum_{i=1}^N x_i \sum_{i=1}^N x_i}$$

For Prediction/Testing:

Given optimal thetas, plug in the x value in our equation to get y

$$\theta_0 = \frac{1}{N} \sum_{i=1}^N y_i - \frac{1}{N} \theta_1 \sum_{i=1}^N x_i$$

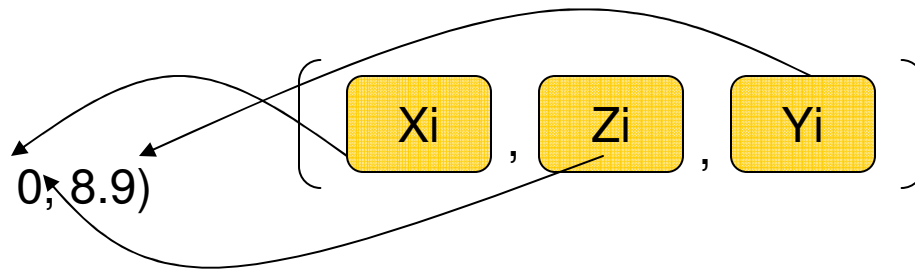
Simple Happiness Scoring Model too Simple?

- So far we have a regression model that was trained on a training data of facebook status updates (text) and labeled happiness score
- Status updates words were mapped to one feature
 - Feature counted number of +ve words
- Maybe too simple?
 - How can we improve the model?
 - Can we add more features?
 - How about count of -ve words as well

Let Us Add One More Feature

- Adding one more feature Z_i representing count of -ve words, now training data will look like the following

- (1, 3, 4)
- (0, 6, 1.8)
- .
- .
- .
- .
- (2, 0, 8.9)



- What would our linear regression function would look like

$$f(x, z) = \theta_0 + \theta_1 x + \theta_2 z$$

Estimation of y i.e. $f(x, z)$ is now a plane instead of a line

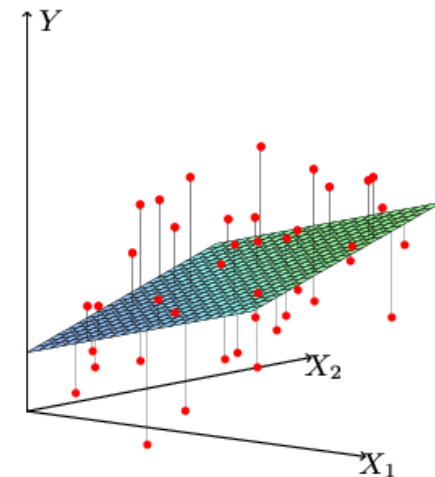


Figure 3.1: *Linear least squares fitting with $X \in \mathbb{R}^2$. We seek the linear function of X that minimizes the sum of squared residuals from Y . [3]*

Empirical Loss with K Features and N Data Points in Matrix Representation

- Representing empirical loss in Matrix form

$$J(\theta) = \frac{1}{2N} \left\| \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ Y_N \end{bmatrix} - \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1K} \\ 1 & x_{21} & x_{22} & \dots & x_{2K} \\ & & \dots & & \\ & & \dots & & \\ & & \dots & & \\ 1 & x_{N1} & x_{N2} & \dots & x_{NK} \end{bmatrix} \cdot \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \cdot \\ \cdot \\ \theta_K \end{bmatrix} \right\|^2$$

Y **X** **θ**

Solve by Setting Partial Derivatives to Zero

- Remember, to find the minimum empirical loss we set the partial derivatives to zero
- We can still do the same in matrix form, we have to set the derivatives to zero

$$\begin{aligned}\frac{\partial J(\theta)}{\partial \theta} &= \frac{1}{2N} \frac{\partial}{\partial \theta} \|Y - X\theta\|^2 \\ &= \frac{1}{2N} \frac{\partial}{\partial \theta} (Y - X\theta)^T (Y - X\theta) = 0\end{aligned}$$

- Solving the above equation we get our best set of parameters

$$\theta^* = (X^T X)^{-1} X^T Y$$

Implementation of Multiple Linear Regression

$$\theta^* = (X^T X)^{-1} X^T Y$$

- Given out N training data points we can build X and Y matrix and perform the matrix operations
- Can use MATLAB
- Or write your own, Matrix multiplication implementation
- Get the theta matrix
- For any new test data plug in the x values (features) in our regression function with the best theta values we have

More Features? Feature Engineering

- So far we have only two features, is it good enough?
- Should we add more features?
- What kind of features can we add?
 - Ratio of +ve/-ve words
 - Normalized count of +ve words
 - Is there a verb in the sentence?
- We need to think what are the kinds of information that may better estimate the Y values
- If we add above 3 features, what is the value of K?

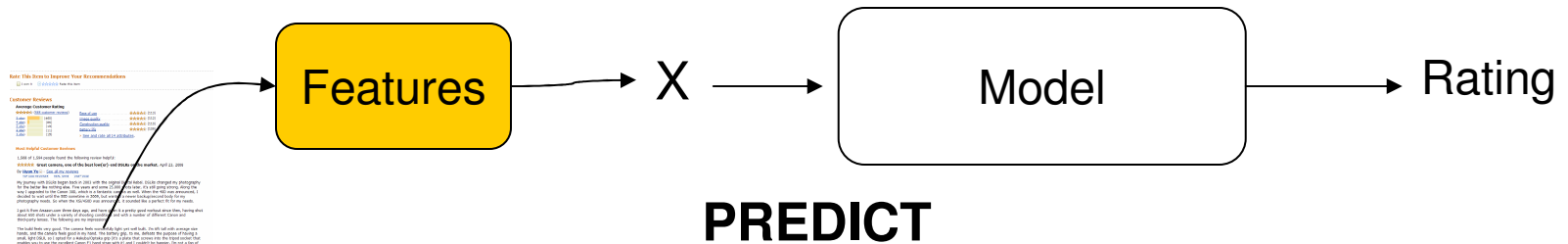
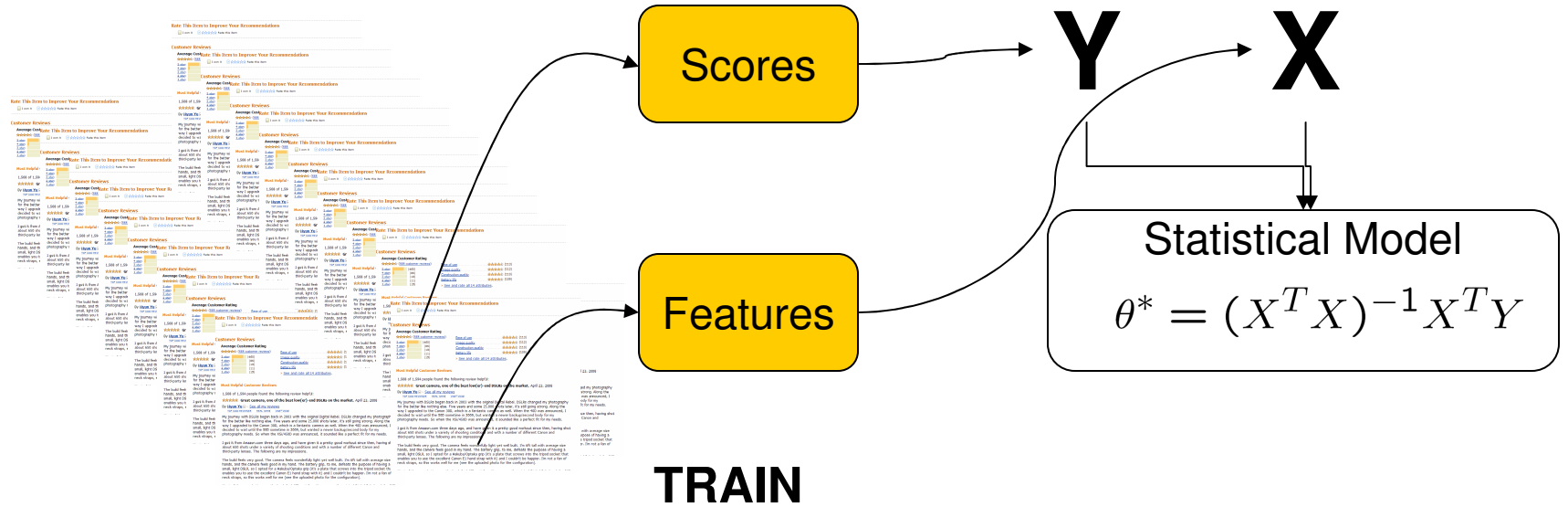
Testing Our Model

- Our goal was to build the best statistical model that would automate the process of scoring a chunk of text (Happiness Score)
- How can we tell how good is our model?
- Remember previously we said let us assume we have 100,000 status updates
- Instead of using all 100K sentences let use the first 90K to build the model
- Use rest of 10K to test the model

Scores from Text, What Else Can They Represent?

- Given a facebook status update we can predict happiness score
- But we can use the same modeling technique in many other problems
 - Summarization: Score may represent importance
 - Question Answering: Score may represent relevance
 - Information extraction : Score may represent relation
- We need to engineer features according to the problem
- Many uses of the statistical technique we learned today

Reviews to Automatic Ratings

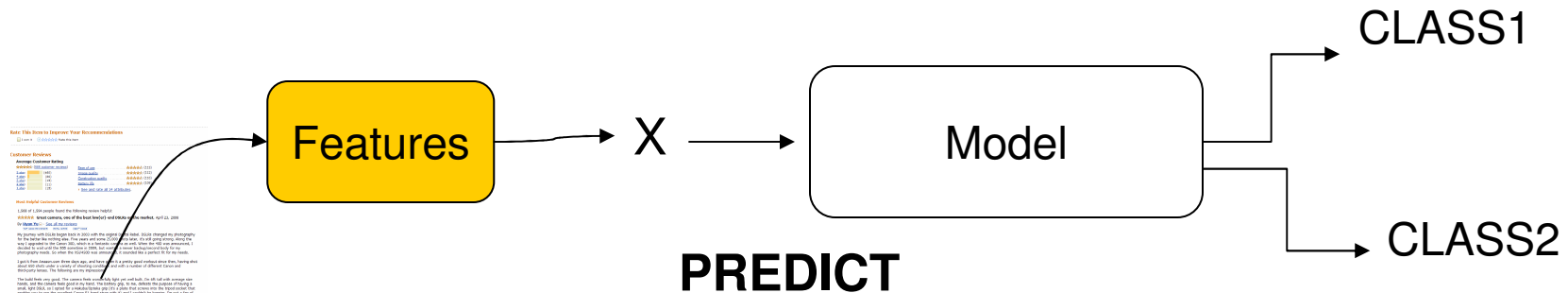


Unstructured Text to Binary Labels

- Let us change the type of problem a bit
- Instead of a real valued happiness score between 0 and 10, let us assume our annotators just provide unhappy (0) or happy (1)
 - Or it can be Amazon review for a product dislike (0) and like (1)
- Can we and should we still model this kind data with regression?

Text Categorization/Classification

- Given any text (sentence, document, stories, etc), we want to classify it into some predefined class set



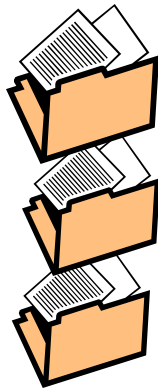
- Training Data consists of Y values that are 0 and 1
 - Review is good or bad
 - Status update is happy or sad

Predicted Output Are Class Labels

- "...is writing a paper"
- "... has flu ☹️"
- "... is happy, yankees won!"



SAD
SAD
HAPPY

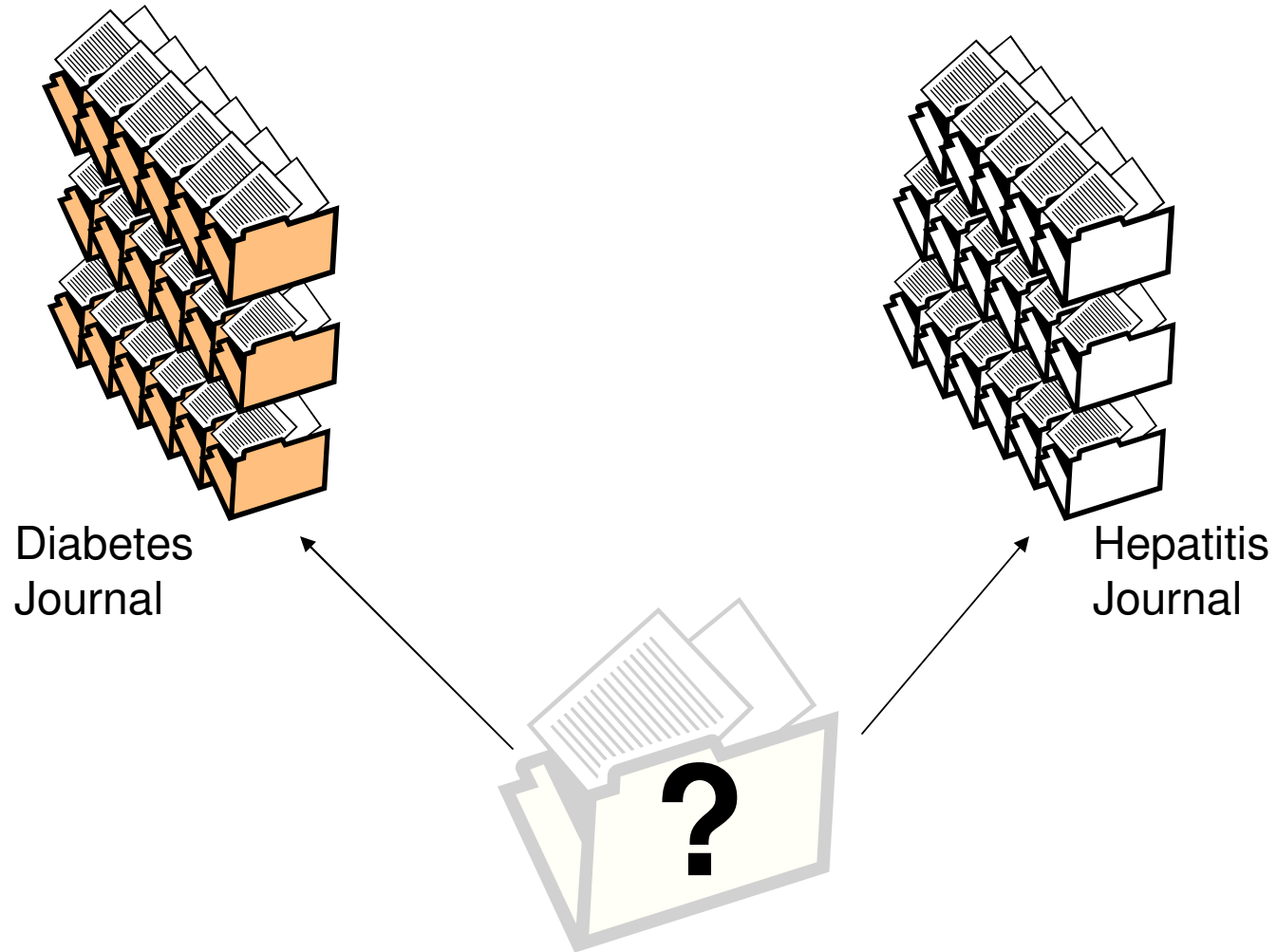


DIABETES
HEPATITIS
HEPATITIS

Class Prediction from Text

- If 'y' outputs are binary classes we may want to use a different modeling technique
- Binary classifiers could model such data
- We need to choose our models according to the problem we are handling
- We probably need better representation of text as well

Text Classification



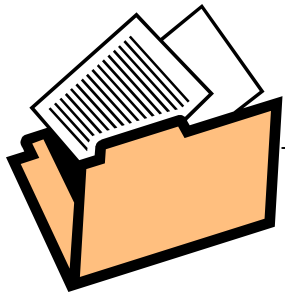
Text Similarity

- To classify a new journal paper into either diabetes group or hepatitis group we could probably compute similarity of this document with the two groups
- How do we compute similarity between text or group of documents?
- First, we need representation of text that takes account of all the information in it?
 - Count of +ve words may not be enough

Text/Document Feature Vectors

- Document Vectors
 - Documents can be represented in different types of vectors: binary vector, multinomial vector, feature vector
- Binary Vector: For each dimension, 1 if the word type is in the document and 0 otherwise
- Multinomial Vector: For each dimension, count # of times word type appears in the document
- Feature Vector: Extract various features from the document and represent them in a vector. Dimension equals the number of features


Example of a Multinomial Document Vector



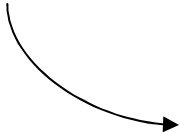
Screening of the critically acclaimed film NumaFung Reserved tickets can be picked up on the day of the show at the box office at Arledge Cinema. Tickets will not be reserved if not paid for in advance.

4	THE	4
2	TICKETS	2
2	RESERVED	2
2	OF	2
2	NOT	2
2	BE	2
2	AT	2
1	WILL	1
1	UP	1
1	SHOW	1
1	SCREENING	1
1	PICKED	1
1	PAID	1
1	ON	1
1	OFFICE	1
1	NUMAFUNG	1
1	IN	1
1	IF	1
1	FOR	1
1	FILM	1
1	DAY	1
1	CRITICALLY	1
1	CINEMA	1
1	CAN	1
1	BOX	1
1	ARLEDGE	1
1	ADVANCE	1
1	ACCLAIMED	1

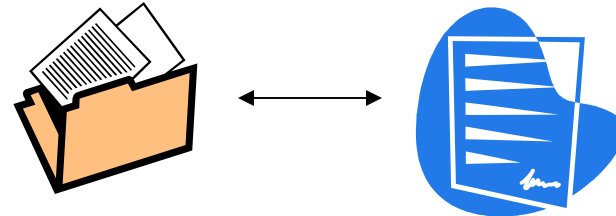
Example of a Multinomial Document Vector



4 THE	4
2 SEATS	2
2 RESERVED	2
2 OF	2
2 NOT	2
2 BE	2
2 AT	2
1 WILL	1
1 UP	1
1 SHOW	1
1 SHOWING	1
1 PICKED	1
1 PAID	1
1 ON	1
1 OFFICE	1
1 VOLCANO	1
1 IN	1
1 IF	1
1 FOR	1
1 FILM	1
1 DAY	1
1 CRITICALLY	1
1 CINEMA	1
1 CAN	1



Find out how similar two text vectors are to find document similarity?



Text Similarity : Cosine of Text Vectors

- Given a set of vectors we can find the cosine similarity between them

$$\text{Cos}\theta = \frac{v_1 \cdot v_2}{\|v_1\| \|v_2\|}$$

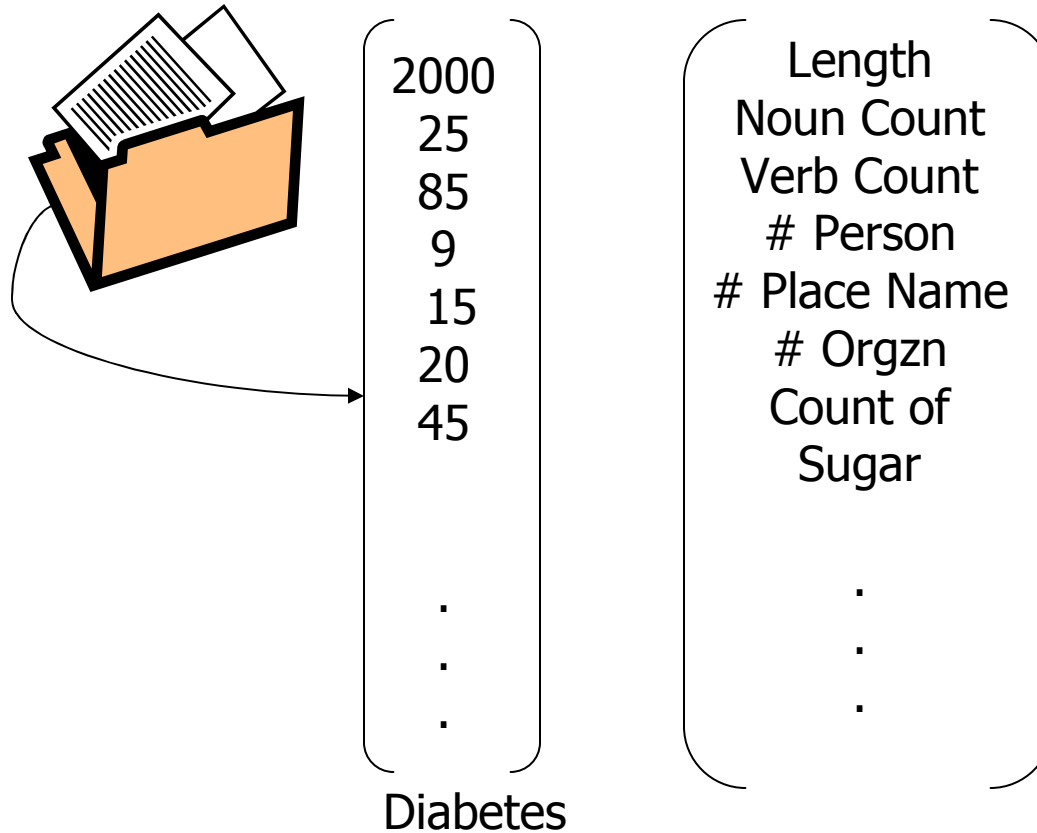
- $\text{Cos } 90 = 0$, vectors are not similar
- Higher cosine value = higher similarity

Feature Vectors

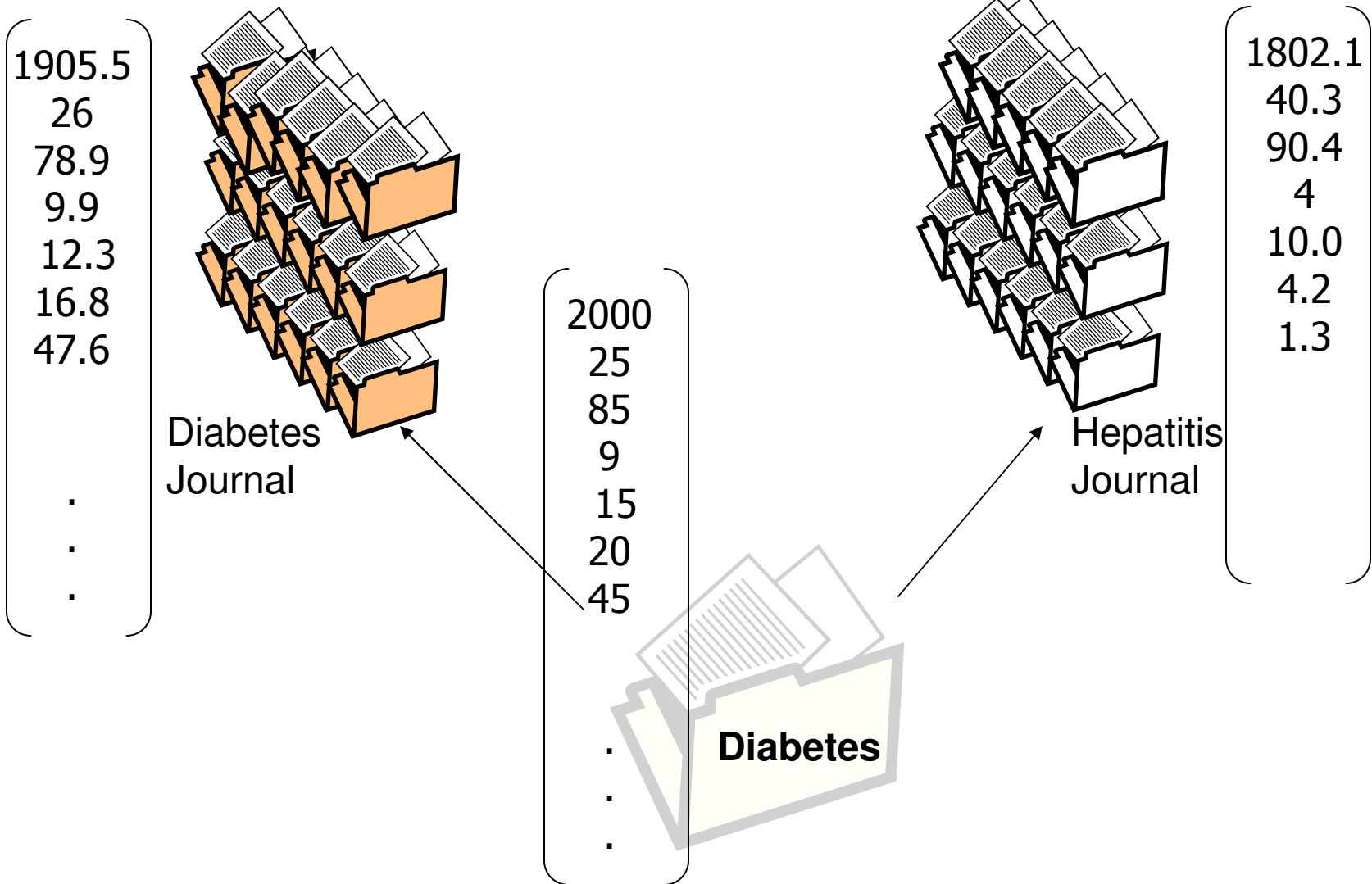
- Instead of just using words to represent documents, we can also extract features and use them to represent the document
- We can extract features like document length (LN), number of nouns (NN), number of verbs (VB), number of person names (PN), number of place (CN) names, number of organization names (ON), number of sentences (NS), number of pronouns (PNN)

Feature Vectors

- Extracting such features you get a feature vector of length 'K' where 'K' is the number of dimensions (features) for each document



Text Classification with Cosine Similarity on Feature Vectors



Cosine Similarity Based Text Classifier

- Build multinomial vectors or feature vectors for each document in the given class
 - Each dimension represent count of the given word or feature
 - Can take average of the vectors to represent a corpus
- 'N' averaged vectors would be the model for 'N' classes of the documents
- For any new document compute similarity of its multinomial vector to the 'N' class vectors
- Highest Cosine Similarity represents the class

Summary

- Text Mining and Linear Regression Model
 - Linear regression can be used to predict scores for unstructured text
 - Extract features from text : (example : count of +ve words)
 - Given the scores and features build regression model by minimizing total loss
- Text Categorization and Linear Classifiers
 - Represent text with binary, multinomial, feature vectors
 - Compute cosine similarity to documents of different classes
 - Many other linear classifiers for text categorization : perceptron is one of them

References

- [1] Christopher Bishop, Pattern Recognition and Machine Learning, Springer, 2006
- [2] Hastie, Tibshirani and Friedman, Elements of Statistical Learning, 2001