Learning with Text

Jason Rennie
jrennie@csail.mit.edu

Text Classification

- What is the topic of this thread?
- What restaurant does it discuss?
- Are the reviews positive or negative? Who likes it, who doesn't?
- What aspects do they like/dislike?

How did you figure this out?

Outline

- Topic Classification
- Compression as Learning
- Learning with Sequences (Information Extraction)

Topic Classification

- **Assumption**: "bag-of-words" is enough.
- Presence of "keywords" tend to identify topic.

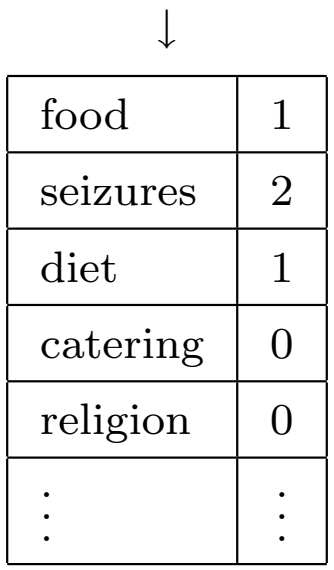
Keyword examples:

- revenue, quarter, sales, year, billion, profit, share
- surgery, skin, patient, overdose, cardiac, brain
- skate, Messier, player, ice, league, penalty, blue

Representation

```
From: dyer@spdcc.com (Steve Dyer)
Subject: Re: food-related seizures?

My comments about the Feingold Diet have no relevance to your
daughter's purported FrostedFlakes-related seizures.  I can't imagine
why you included it.
```



food	1
seizures	2
diet	1
catering	0
religion	0
⋮	⋮

Representation

- Punctuation is removed, case is ignored, words are separated into tokens. Known as "feature vector" or "bag-of-words" representation.
- Vector length is size of vocabulary. Common vocabulary size is 10,000-100,000. Classification problem is very high dimensional.

Topic Classification

- Represent document as vector of word counts.
- Can use any number of supervised learning techniques to learn parameters (SVM, RLSC, RLR, AdaBoost, etc.)
- "Keywords" will get largest weights.

Word Weighting

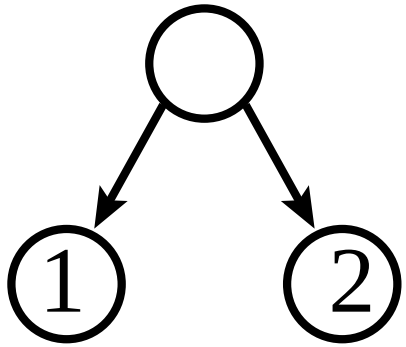
- Better results are achieved using transformed weights instead of raw counts.
- Term frequency: $x_i = \log(1 + x_i)$
- Inverse Doc. frequency: $x_i = x_i \log\left(\frac{\text{total \# docs}}{\text{docs w/term}}\right)$
- Length normalization: $x_i = \frac{x_i}{\|\vec{x}\|}$

Informative Words

- IDF is a sort of informativeness criterion
- Can we do better?
- Keywords tend to occur somewhat rarely.
- Keywords also tend to be bursty
  - Don't occur in many documents
  - But, high frequency when they do occur
- Suggests a mixture model...

Mixture Models

- Randomly selects between two different models
- For example
  - Prior model (off-topic)
  - High frequency model (on-topic)



- Can capture distributions that have two different "modes" of operation.
- (Simple) Unigram model: $p_S(d;\theta) = \theta^h(1-\theta)^{n-h}$
- Mixture of two unigrams:

$$p_M(d;\lambda_\tau,\tau_1,\tau_2) = \lambda_\tau \tau_1^h(1-\tau_1)^{n-h} + (1-\lambda_\tau)\tau_2^h(1-\tau_2)^{n-h}$$

Mixture Models

- Like having two weighted coins, each with a different chance of heads
- For each thread, randomly choose a coin.
- For each word position, flip coin.
- Heads: chosen word occurs.
- Tails: some other word occurs.

Informative Words

- Informative words have a peaked distribution.
  - They occur with high frequency in a few threads
- Mixture should provide much better fit than unigram
- **Idea**: train mixture, simple models, use log-odds ratio
- **Informativeness score**:
  - Find maximum-likelihood parameters for mixture: $\lambda_\tau, \tau_1, \tau_2$
  - Find maximum-likelihood parameter for simple unigram: θ
  - Calculate log-odds ratio:

$$\text{score} = \log \frac{p_M(d; \lambda_\tau, \tau_1, \tau_2)}{p_S(d; \theta)}$$

Informal Communication

- There are 4 billion web pages
- There are 4 billion e-mails sent out *every day*

Informal communication:

- is threaded, and
- has author information.

Informative Words

- Author information can help us identify *uninformative* words.
- Again look at mixture model fit.
- This time, look at author statistics.
- h is number of occurrences for a particular author
- Find maximum likelihood parameters: $\lambda_\alpha, \alpha_1, \alpha_2$

Informativeness Criterion

- Score words by log-odds of thread-mixture and author-mixture:

$$\text{score} = \log \frac{p_M(\vec{w}; \lambda_\tau, \tau_1, \tau_2)}{p_M(\vec{w}; \lambda_\alpha, \alpha_1, \alpha_2)}$$

Some Experiments

- Restaurant name extraction:

	F1
Baseline	52.8%
IDF	53.2%
thread-simple log-odds	53.2%
thread-author log-odds	54.0%

Text Classification

- What is the topic of this thread?
- What restaurant does it discuss?
- Are the reviews positive or negative? Who likes it, who doesn't?
- What aspects do they like/dislike?

How did you figure this out?

Language Understanding

- We:
  - translate the document into an internal representation that is easy to manipulate. We "understand" the document.
- We:
  - have architechture designed for quick language acquisition,
  - have years of experience using language,
  - are embodied,
  - can clarify misunderstandings through interaction, and
  - experience concepts used in language.

Language Understanding

- Computers:
  - see the world as 1s and 0s.

Computers don't have a chance!
Or do they?

How We Learn

- Few people have "photographic" memory.
- We usually understand and remember things by associating them with concepts we know.
- We learn/understand most quickly when something is closely related to what we know; we have learning something that is unrelated to what we know.

We learn most easily when we can *compress* the new information.

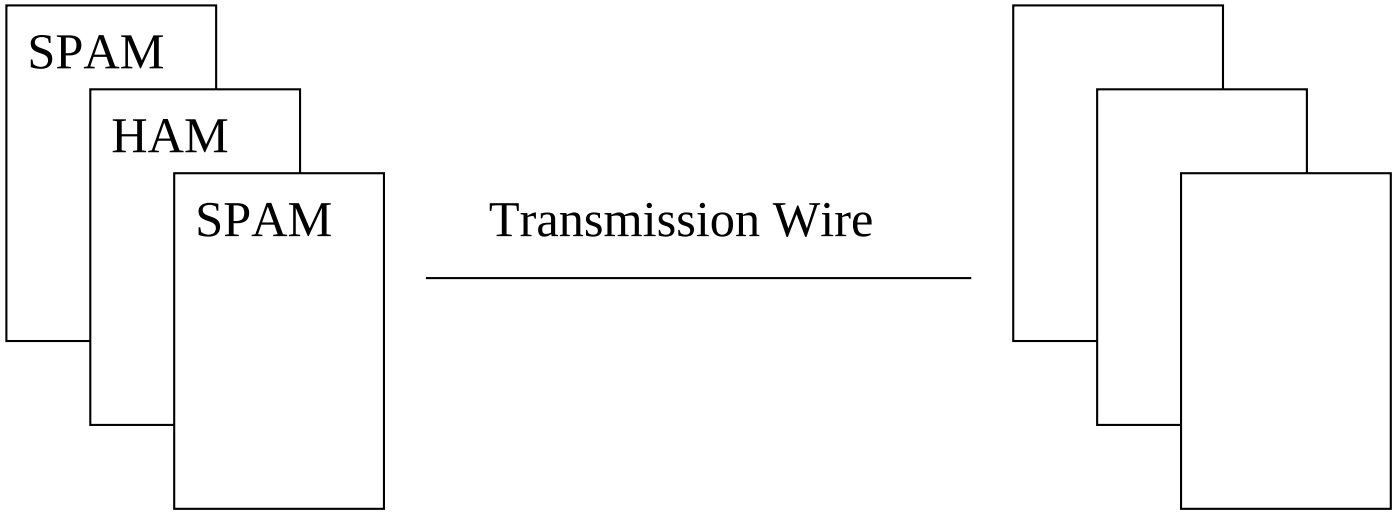
Minimum Description Length

- Concept introduced by Jorma Rissanen (1978).
- **Idea:** Best generalization achieved by smallest encoding of training examples.

Compression as Learning

Learning problem:

- Documents and labels (e.g. ham/spam) at one end of wire.
- Copy of documents at other end of wire.
- What is fewest number of bits needed to transmit labels?



Encoding Length

Must encode:

- Parameters that specify encoding of labels (prior dist.)
- Labels of examples (conditional dist.)

Total encoding length is

$$-\log p(\vec{\theta}) - \sum_i \log p(y_i|x_i;\vec{\theta})$$

Topic Classification

- Substrings may provide better compression than words
  - Suffixes, prefixes, roots may be more general
- Consider rule-based classification
  - If document contains **string**, assign category **label**
- Length of rule is (1) encoding of string, plus (2) encoding of label

A Simple MDL Algorithm

- Let length = (# examples) $\cdot \log_2$(# labels)
- Let ruleSet = {}
- while (length < oldLength)
  - foreach newRule
    - calcLength({ruleSet, newRule})
  - ruleSet = {ruleSet, bestNewRule}
  - length = calcLength(ruleSet)
- Related to Boosting

MDL Learning of String Features

␣x	comp.os.xwindows
␣windows	comp.os.ms-windows.misc
␣car␣	rec.autos
for␣sale	misc.forsale
␣turk	talk.politics.mideast
486	comp.sys.ibm.pc.hardware
3.1	comp.os.ms-windows.misc
␣␣$	misc.forsale
t␣condition	misc.forsale

Encoding Real-Valued Parameters

- How to encode real-valued parameters?
- Exact encoding seems impossible.
- Limited-precision encoding is tricky.
  - How to choose the level of precision?

The Bits-Back Argument (Hinton & Zemel, 1994)

- Let $p(\theta)$ be prior distribution on parameters
- Send θ with probability $q(\theta)$
- Encoding length:

$$-\int q(\theta) \log p(\theta) d\theta$$

The Bits-Back Argument

- Choice of θ (from q) can encode other information: $H(q)$ bits
- Net parameter encoding length is

$$\begin{aligned} L(p,q) &= -\int q(\theta) \log p(\theta) d\theta - H(q) \\ &= \int q(\theta) \log \frac{q(\theta)}{p(\theta)} d\theta \\ &= KL(q||p) \end{aligned}$$

- Encoding length of data is

$$-\sum_i \log p(y_i|x_i;\vec{\theta})$$

Bits-Back: Example

- Let the conditional distribution be log-linear:

$$\log p(y_i|x_i;\vec{\theta}) \propto \vec{\phi}(x_i, y_i) \cdot \vec{\theta}$$

- Let prior, $p(\theta)$, be a zero-mean, constant-width (σ^2) Gaussian.
- Let $q(\theta)$ be a constant-width (σ^2) Gaussian with mean $\vec{\theta}$.
- Net parameter encoding length is

$$\frac{\|\vec{\theta}\|^2}{2\sigma^2} + \text{constant}$$

Bits-Back: Example

- Total encoding length is:

$$-\vec{f}(x_i, y_i) \cdot \vec{\theta} + \log \sum_y e^{\vec{f}(x_i, y) \cdot \vec{\theta}} + \frac{\|\theta\|^2}{2\sigma^2}.$$

- Equivalent to maximum-likelihood objective for Regularized Logistic Regression.
- Kernel "trick" can be applied to incorporate high-dimensional feature spaces.

String Kernels

- MDL string learning algorithm enumerates a huge range of features each round—not very efficient.
- Kernels allow efficient manipulation of high-dimensional feature spaces.
- Lodhi et al. (2001) showed how to efficiently calculate string kernels.

String Kernels

	c-a	c-t	a-t	b-a	b-t	c-r	a-r	b-r
$\vec{\phi}(\text{cat})$	λ^2	λ^3	λ^2	0	0	0	0	0
$\vec{\phi}(\text{car})$	λ^2	0	0	0	0	λ^3	λ^2	0
$\vec{\phi}(\text{bat})$	0	0	λ^2	λ^2	λ^3	0	0	0
$\vec{\phi}(\text{bar})$	0	0	λ^2	0	0	0	λ^2	λ^3

$$K(\text{car}, \text{cat}) = \frac{\vec{\phi}(\text{car})}{\|\vec{\phi}(\text{car})\|} \frac{\vec{\phi}(\text{cat})}{\|\vec{\phi}(\text{cat})\|} = \frac{\lambda^4}{2\lambda^4 + \lambda^6}$$

String Kernels

- Use dynamic programming to calculate kernel efficiently
- Let $\mathbf{i} = \{i_1, \ldots, i_{|i|}\}$ be an increasing sequence of indices
- $s[\mathbf{i}]$ is a substring of s
- Define modified feature function:

$$\vec{\phi}'_u(s) = \begin{cases} \lambda^{|s|-i_1+1} & \text{if } \exists \mathbf{i} \text{ s.t. } s[\mathbf{i}] = u \\ 0 & \text{otherwise} \end{cases}$$

Measures from beginning of substring in s to end of s.

String Kernels: Recursion

$$K'_0(s,t) = 1, \ \forall s,t$$

$$K'_i(s,t) = 0, \ \text{if} \ \min(|s|,|t|) < i$$

$$K_i(s,t) = 0, \ \text{if} \ \min(|s|,|t|) < i$$

$$K'_i(sx,t) = \lambda K'_i(s,t) + \sum_{j:t_j=x} K'_{i-1}(s,t[1:j-1])\lambda^{|t|-j+2}, \ i < n$$

$$K_n(sx,t) = K_n(s,t) + \sum_{j:t_j=x} K'_{n-1}(s,t[1:j-1])\lambda^2$$

String Kernel Results (Lodhi et al., 2001)

	F1	Precision	Recall
3 S-K	0.925	0.981	0.878
5 S-K	0.936	0.992	0.888
6 S-K	0.936	0.992	0.888
W-K	0.925	0.989	0.867

Sequence Classification

- So far, we've dealt with single-label data:

$$\{(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)\} \tag{1}$$

- Many Natural Language tasks deal with sequences of labels:

$$\{(\vec{x}_1, \vec{y}_1), \ldots, (\vec{x}_n, \vec{y}_n)\} \tag{2}$$

- Training one set of parameters per class is not practical.

Information Extraction

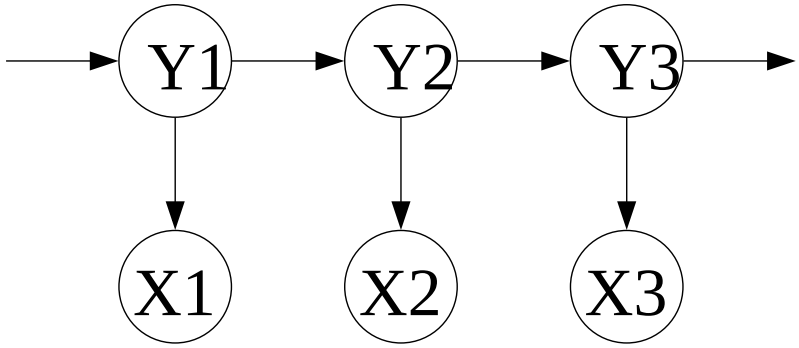
- How do we identify restaurant names, company names, locations, etc.?
- Syntactic features: punctuation, capitalization, spelling
- Context features: neighboring words, part-of-speech
- Labels of neighboring words.

IE is a sequence classification problem

N	N	N	R	R	N	L	L
I	ate	at	Blue	Room	in	Kendall	Square

Hidden Markov Models

- For document classification, we assumed document labels didn't depend on each other.
- For information extraction, label depends on labels of neighboring words
- HMMs assume that observed word depends on label, each label depends on previous label



Hidden Markov Models

- HMMs are a *generative* model.
- Composed of
  - Model for producing next label: $p(y^{(t)}|y^{(t-1)})$
  - Model for producing words: $p(x^{(t)}|y^{(t)})$
- Traning consists of counting
- Labeling involves finding maximum-likelihood sequence
  - Forward-backward algorithm is used for efficient calculation

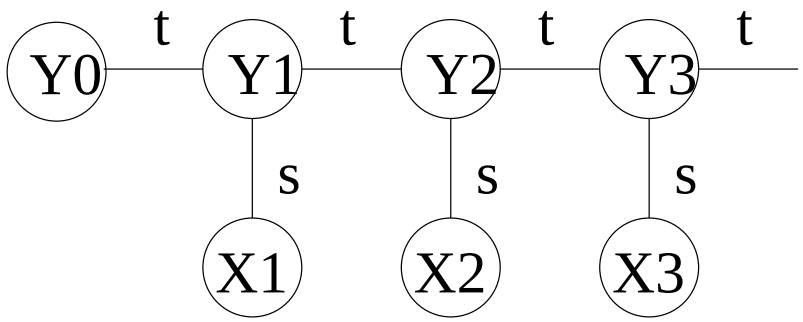
Generative vs. Discriminative

- Discriminative models work best for single-label classification, what about for sequences?
- Problem: we can't train one set of parameters per possible label-sequence.
- But, like HMMs, if we only allow dependence between adjacent labels, learning can be made efficient.

Conditional Random Fields (Lafferty et al., 2001)

- State features: function of current observation and state
- Transition features: function of current state and previous state
- Global feature vector for an example:

$$\vec{F}(\vec{x}, \vec{y}) = \sum_i \vec{f}(\vec{x}, \vec{y}, i)$$



Conditional Random Fields

- Log-linear conditional model:

$$\log p(\vec{y}|\vec{x}) = \vec{\lambda} \cdot \vec{F}(\vec{x}, \vec{y}) - \log Z_{\vec{\lambda}}$$

- Let $\{(\vec{x}_1, \vec{y}_1), \ldots, (\vec{x}_n, \vec{y}_n)\}$ be training data.
- Want to maximize the likelihood of the labels:

$$L_{\vec{\lambda}} = \sum_i \log p(\vec{y}_i|\vec{x}_i)$$

Conditional Random Fields

- Need to evalulate the gradient (Sha & Pereira, 2003):

$$\nabla L_{\vec{\lambda}} = \sum_i \left[\vec{F}(\vec{x}_i, \vec{y}_i) - E_{p(\vec{y}|\vec{x}_i)} \vec{F}(\vec{x}_i, \vec{y}) \right]$$

- Maximum is achieved when expectations match empirical counts.
- Efficient calculation of expectation is similar to forward-backward HMM algorithm

Conditional Random Fields

- Define the transition matrix:

$$M_i[y, y'] = \exp\left(\vec{\lambda} \cdot \vec{f}(\vec{x}, \vec{y}, i)\right)$$

- Define forward and backward vectors:

$$\vec{\alpha}_i = \begin{cases} \vec{\alpha}_{i-1} M_i & 1 \leq i \leq n \\ \vec{1} & i = 0 \text{ (start state)} \end{cases}$$

$$\vec{\beta}_i^T = \begin{cases} M_{i+1} \vec{\beta}_{i+1}^T & 1 \leq i \leq n \\ \vec{1} & i = n+1 \text{ (end state)} \end{cases}$$

Conditional Random Fields

- The j^{th} element of $\vec{\alpha}_i$ is the (unnormalized) forward probability of being in state j at step i.
- The j^{th} element of $\vec{\beta}_i$ is the (unnormalized) backward probability of being in state j at step i.

Conditional Random Fields

- The normalization constant is

$$Z_{\vec{\lambda}} = \vec{\alpha}_n \cdot \vec{1}^T,$$

- The expectation is calculated individually for each feature:

$$E_{p(\vec{y}|\vec{x})} F(\vec{x}, \vec{y}) = \sum_{\vec{y}} p(\vec{y}|\vec{x}) F(\vec{x}, \vec{y})$$

$$= \sum_i \frac{\alpha_{i-1}(f_i * M_i)\beta_i^T}{Z_{\vec{\lambda}}}$$

- f_i is the feature corresponding to the entry of the M matrix
- $*$ denotes component-wise matrix multiplication

CRF: Experiments

- Lafferty et al. (2001) compared CRFs vs. HMMs for part-of-speech tagging.

	error	oov error
HMM	5.69%	45.99%
CRF	5.55%	48.05%
CRF†	4.27%	23.76%

Table 1: † - with spelling features

- SVM version: Max-Margin Markov Networks (Taskar et al., 2003)

Summary

- Text Learning occurs in many different forms
- Informal communication presents interesting problems
- Mixture models can be used to find "informative" words
- Compression serves as a framework for learning
  - Connection to regularized maximum likelihood training
- Information extraction problems take the form of sequence learning

References

Hinton, G. E., & Zemel, R. S. (1994). Autoencoders, minimum description length, and Helmholtz free energy. *Advances in Neural Information Processing Systems 6.*

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the Eighteenth International Conference on Machine Learning.*

Lodhi, H., Shawe-Taylor, J., Cristianini, N., & Watkins, C. J. C. H. (2001). Text classification using string kernels. *Advances in Neural Information Processing Systems 13.*

Sha, F., & Pereira, F. (2003). Shallow parsing with conditional random fields. *Proceedings of HLT-NAACL.*

Taskar, B., Guestrin, C., & Koller, D. (2004). Max-margin markov

networks. *Advances in Neural Information Processing Systems 16*.