Statistical NLP
Spring 2010

Lecture 4: Text Categorization

Dan Klein – UC Berkeley

Overview

- So far: language models give P(s)
  - Help model fluency for various noisy-channel processes (MT, ASR, etc.)
  - N-gram models don't represent any deep variables involved in language structure or meaning
  - Usually we want to know something about the input other than how likely it is (syntax, semantics, topic, etc)
- Next: Naïve Bayes models
  - We introduce a single new global variable
  - Still a very simplistic model family
  - Lets us model hidden properties of text, but only very non-local ones…
  - In particular, we can only model properties which are largely invariant to word order (like topic)

Text Categorization

- Want to classify documents into broad semantic topics (e.g. politics, sports, etc.)

Obama is hoping to rally support for his $825 billion stimulus package on the eve of a crucial House vote. Republicans have expressed reservations about the proposal, calling for more tax cuts and less spending. GOP representatives seemed doubtful that any deals would be made.

California will open the 2009 season at home against Maryland Sept. 5 and will play a total of six games in Memorial Stadium in the final football schedule announced by the Pacific-10 Conference Friday. The original schedule called for 12 games over 12 weekends.

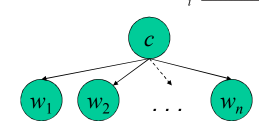
- Which one is the politics document? (And how much deep processing did that decision take?)
- One approach: bag-of-words and Naïve-Bayes models
- Another approach later…
- Usually begin with a labeled corpus containing examples of each class

Naïve-Bayes Models

- Idea: pick a topic, then generate a document using a language model for that topic.
- Naïve-Bayes assumption: all words are independent given the topic.

$$P(c, w_1, w_2, \dots w_n) = P(c)\prod_i \boxed{P(w_i \mid c)}$$

We have to smooth these!



w_n = STOP

- Compare to a unigram language model:

$$P(w_1, w_2, \dots w_n) = \prod_i P(w_i)$$

Using NB for Classification

- We have a joint model of topics and documents

$$P(c, w_1, w_2, \dots w_n) = P(c)\prod_i P(w_i \mid c)$$

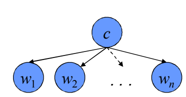
- Gives posterior likelihood of topic given a document

$$P(c \mid w_1, w_2, \dots w_n) = \frac{P(c)\prod_i P(w_i \mid c)}{\sum_{c'}\left[P(c')\prod_i P(w_i \mid c')\right]}$$

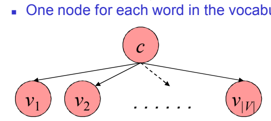
- What about totally unknown words?
- Can work shockingly well for textcat (especially in the wild)
- How can unigram models be so terrible for language modeling, but class-conditional unigram models work for textcat?
- Numerical / speed issues
- How about NB for spam detection?

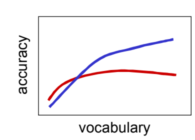
Two NB Formulations

- Two NB event models for text categorization
  - The class-conditional unigram model, a.k.a. multinomial model
    - One node per word in the document
    - Driven by words which are present
    - Multiple occurrences, multiple evidence
    - Better overall – plus, know how to smooth



  - The binominal (binary) model
    - One node for each word in the vocabulary





    - Incorporates explicit negative correlations
    - Know how to do feature selection (e.g. keep words with high mutual information with the class variable)

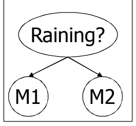
Example: Barometers

Reality

Raining | Sunny

P(+,+,r) = 1/8 | P(-,-,r) = 3/8 | P(+,+,s) = 3/8 | P(-,-,s) = 1/8

NB Model



NB FACTORS:
- P(s) = 1/2
- P(-|s) = 1/4
- P(-|r) = 3/4

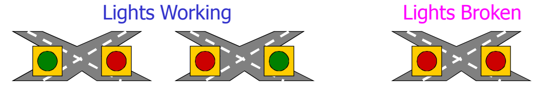
PREDICTIONS:
- P(r,-,-) = (½)(¾)(¾)
- P(s,-,-) = (½)(¼)(¼)
- P(r|-,-) = 9/10
- P(s|-,-) = 1/10

Overconfidence!

Example: Stoplights

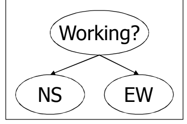
Reality

Lights Working | Lights Broken



P(g,r,w) = 3/7 | P(r,g,w) = 3/7 | P(r,r,b) = 1/7

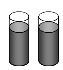
NB Model



NB FACTORS:
- P(w) = 6/7
- P(r|w) = 1/2
- P(g|w) = 1/2
- P(b) = 1/7
- P(r|b) = 1
- P(g|b) = 0

P(b|r,r) = 4/10 (what happened?)

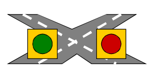
(Non-)Independence Issues



- Mild Non-Independence
  - Evidence all points in the right direction
  - Observations just not entirely independent
  - Results
    - Inflated Confidence
    - Deflated Priors
  - What to do? Boost priors or attenuate evidence

$$P(c, w_1, w_2, \ldots w_n) \text{"="} P(c)^{boost>1} \prod_i P(w_i \mid c)^{boost<1}$$

- Severe Non-Independence
  - Words viewed independently are misleading
  - Interactions have to be modeled
  - What to do?
    - Change your model!



Language Identification

- How can we tell what language a document is in?

The 38th Parliament will meet on Monday, October 4, 2004, at 11:00 a.m. The first item of business will be the election of the Speaker of the House of Commons. Her Excellency the Governor General will open the First Session of the 38th Parliament on October 5, 2004, with a Speech from the Throne.

La 38e législature se réunira à 11 heures le lundi 4 octobre 2004, et la première affaire à l'ordre du jour sera l'élection du président de la Chambre des communes. Son Excellence la Gouverneure générale ouvrira la première session de la 38e législature avec un discours du Trône le mardi 5 octobre 2004.

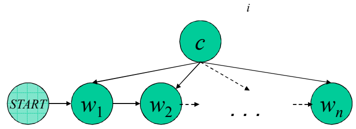
- How to tell the French from the English?
  - Treat it as word-level textcat?
    - Overkill, and requires a lot of training data
    - You don't actually need to know about words!
  - Option: build a character-level language model

Σύμφωνο σταθερότητας και ανάπτυξης
Patto di stabilità e di crescita

Class-Conditional LMs

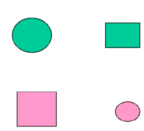
- Can add a topic variable to richer language models

$$P(c, w_1, w_2, \ldots w_n) = P(c) \prod_i P(w_i \mid w_{i-1}, c)$$



- Could be characters instead of words, used for language ID (HW2)
- Could sum out the topic variable and use as a language model
- How might a class-conditional n-gram language model behave differently from a standard n-gram model?

Clustering / Pattern Detection



Soccer team wins match
Stocks close up 3%
Investing in the stock market has …
The first game of the world series …

- Problem 1: There are many patterns in the data, most of which you don't care about.

Clustering vs. Classification

- Classification: we specify which pattern we want, features uncorrelated with that pattern are idle

P(w|sports)	P(w|politics)	P(w|headline)	P(w|story)
the 0.1	the 0.1	the 0.05	the 0.1
game 0.02	game 0.005	game 0.01	game 0.01
win 0.02	win 0.01	win 0.01	win 0.01

- Clustering: the clustering procedure locks on to whichever pattern is most salient, statistically
  - P(content words | class) will learn topics
  - P(length, function words | class) will learn style
  - P(characters | class) will learn "language"

Model-Based Clustering

- Clustering with probabilistic models:

Unobserved (Y)	Observed (X)
??	LONDON -- Soccer team wins match
??	NEW YORK – Stocks close up 3%
??	Investing in the stock market has …
??	The first game of the world series …

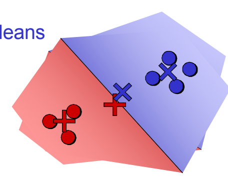
Build a model of the domain: $P(x, y, \theta)$

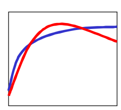
Often: find θ to maximize: $P(x|\theta) = \sum_y P(x, y|\theta)$

- Problem 2: The relationship between the structure of your model and the kinds of patterns it will detect is complex.

Learning Models with EM

- Hard EM: alternate between
  - E-step: Find best "completions" Y for fixed θ
  - M-step: Find best parameters θ for fixed Y
- Example: K-Means





- Problem 3: Data likelihood (usually) isn't the objective you really care about
- Problem 4: You can't find global maxima anyway

Hard EM for Naïve-Bayes

- Procedure: (1) we calculate best completions:

$$y^* = \arg\max_y \mathrm{P}(y) \prod_i \mathrm{P}(x_i|y)$$

- (2) compute relevant counts from the completed data:

$$c(w, y) = \sum_{x \in D} \sum_i [1(x_i = w, y^* = y)]$$

- (3) compute new parameters from these counts (divide)
- (4) repeat until convergence
- Can also do this when some docs are labeled

EM: More Formally

- Hard EM: $\arg\max_{\theta, y} \mathrm{P}(y, \theta|x)$
- Improve completions

$$y^* = \arg\max_y \mathrm{P}(y, \theta^*|x) = \arg\max_y \mathrm{P}(y|x, \theta^*)$$

- Improve parameters

$$\theta^* = \arg\max_\theta \mathrm{P}(y^*, \theta|x) = \arg\max_\theta \mathrm{P}(\theta|x, y^*)$$

- Each step either does nothing or increases the objective

Soft EM for Naïve-Bayes

- Procedure: (1) calculate posteriors (soft completions):

$$\mathrm{P}(y|x) = \frac{\mathrm{P}(y) \prod_i \mathrm{P}(x_i|y)}{\sum_{y'} \mathrm{P}(y') \prod_i \mathrm{P}(x_i|y')}$$

- (2) compute expected counts under those posteriors:

$$c(w, y) = \sum_{x \in D} \mathrm{P}(y|x) \sum_i [1(x_i = w, y)]$$

- (3) compute new parameters from these counts (divide)
- (4) repeat until convergence

EM in General

- We'll use EM over and over again to fill in missing data
  - Convenience Scenario: we want P(x), including y just makes the model simpler (e.g. mixing weights for language models)
  - Induction Scenario: we actually want to know y (e.g. clustering)
  - NLP differs from much of statistics / machine learning in that we often want to interpret or use the induced variables (which is tricky at best)
- General approach: alternately update y and θ
  - E-step: compute posteriors $P(y|x,\theta)$
    - This means scoring all completions with the current parameters
    - Usually, we do this implicitly with dynamic programming
  - M-step: fit θ to these completions
    - This is usually the easy part – treat the completions as (fractional) complete data
  - Initialization: start with some noisy labelings and the noise adjusts into patterns based on the data and the model
  - We'll see lots of examples in this course
- EM is only locally optimal (why?)

Heuristic Clustering?

- Many methods of clustering have been developed
  - Most start with a pairwise distance function
  - Most can be interpreted probabilistically (with some effort)
  - Axes: flat / hierarchical, agglomerative / divisive, incremental / iterative, probabilistic / graph theoretic / linear algebraic
- Examples:
  - Single-link agglomerative clustering
  - Complete-link agglomerative clustering
  - Ward's method
  - Hybrid divisive / agglomerative schemes

Document Clustering

- Typically want to cluster documents by topic
- Bag-of-words models usually do detect topic
  - It's detecting deeper structure, syntax, etc. where it gets really tricky!
- All kinds of games to focus the clustering
  - Stopword lists
  - Term weighting schemes (from IR, more later)
  - Dimensionality reduction (more later)

Word Senses

- Words have multiple distinct meanings, or senses:
  - Plant: living plant, manufacturing plant, …
  - Title: name of a work, ownership document, form of address, material at the start of a film, …
- Many levels of sense distinctions
  - Homonymy: totally unrelated meanings (river bank, money bank)
  - Polysemy: related meanings (star in sky, star on tv)
  - Systematic polysemy: productive meaning extensions (metonymy such as organizations to their buildings) or metaphor
  - Sense distinctions can be extremely subtle (or not)
- Granularity of senses needed depends a lot on the task
- Why is it important to model word senses?
  - Translation, parsing, information retrieval?

Word Sense Disambiguation

- Example: living plant vs. manufacturing plant
- How do we tell these senses apart?
  - "context"

  The manufacturing plant which had previously sustained the town's economy shut down after an extended labor strike.

  - Maybe it's just text categorization
  - Each word sense represents a topic
  - Run a naive-bayes classifier?
- Bag-of-words classification works ok for noun senses
  - 90% on classic, shockingly easy examples (line, interest, star)
  - 80% on senseval-1 nouns
  - 70% on senseval-1 verbs

Various Approaches to WSD

- Unsupervised learning
  - Bootstrapping (Yarowsky 95)
  - Clustering
- Indirect supervision
  - From thesauri
  - From WordNet
  - From parallel corpora
- Supervised learning
  - Most systems do some kind of supervised learning
  - Many competing classification technologies perform about the same (it's all about the knowledge sources you tap)
  - Problem: training data available for only a few words

Resources

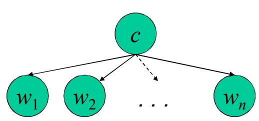
- WordNet
  - Hand-build (but large) hierarchy of word senses
  - Basically a hierarchical thesaurus
- SensEval -> SemEval
  - A WSD competition, of which there have been 3+3 iterations
  - Training / test sets for a wide range of words, difficulties, and parts-of-speech
  - Bake-off where lots of labs tried lots of competing approaches
- SemCor
  - A big chunk of the Brown corpus annotated with WordNet senses
- OtherResources
  - The Open Mind Word Expert
  - Parallel texts
  - Flat thesauri

Verb WSD

- Why are verbs harder?
  - Verbal senses less topical
  - More sensitive to structure, argument choice
- Verb Example: "Serve"
  - [function] The tree stump serves as a table
  - [enable] The scandal served to increase his popularity
  - [dish] We serve meals for the homeless
  - [enlist] She served her country
  - [jail] He served six years for embezzlement
  - [tennis] It was Agassi's turn to serve
  - [legal] He was served by the sheriff

Knowledge Sources

- So what do we need to model to handle "serve"?
  - There are distant topical cues
    - …. point … court ………………… serve ……… game …



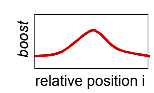
$$P(c, w_1, w_2, \ldots w_n) = P(c)\prod_i P(w_i \mid c)$$

Weighted Windows with NB

- Distance conditioning
  - Some words are important only when they are nearby
  - …. as …. point … court ………………… serve ……… game …
  - …. ……………………………………… serve as……. ……….

$$P(c, w_{-k}, \ldots, w_{-1}, w_0, w_{+1}, \ldots w_{+k'}) = P(c)\prod_{i=-k}^{k'} P(w_i \mid c, bin(i))$$

- Distance weighting
  - Nearby words should get a larger vote
  - … court …… serve as……… game …… point



$$P(c, w_{-k}, \ldots, w_{-1}, w_0, w_{+1}, \ldots w_{+k'}) = P(c)\prod_{i=-k}^{k'} P(w_i \mid c)^{boost(i)}$$

Better Features

- There are smarter features:
  - Argument selectional preference:
    - serve NP[meals] vs. serve NP[papers] vs. serve NP[country]
  - Subcategorization:
    - [function] serve PP[as]
    - [enable] serve VP[to]
    - [tennis] serve <intransitive>
    - [food] serve NP {PP[to]}
  - Can capture poorly (but robustly) with local windows
  - … but we can also use a parser and get these features explicitly
- Other constraints (Yarowsky 95)
  - One-sense-per-discourse (only true for broad topical distinctions)
  - One-sense-per-collocation (pretty reliable when it kicks in: manufacturing plant, flowering plant)

Complex Features with NB?

- Example: Washington County jail served 11,166 meals last month - a figure that translates to feeding some 120 people three times daily for 31 days.
- So we have a decision to make based on a set of cues:
  - context:jail, context:county, context:feeding, …
  - local-context:jail, local-context:meals
  - subcat:NP, direct-object-head:meals
- Not clear how build a generative derivation for these:
  - Choose topic, then decide on having a transitive usage, then pick "meals" to be the object's head, then generate other words?
  - How about the words that appear in multiple features?
  - Hard to make this work (though maybe possible)
  - No real reason to try