Lecture 24:
Document and Web Applications

Sam Roweis

March 31, 2004

Machine Learning Problems for Text/Web Data

• Document / Web Page Classification or Detection
  1. Does this document/web page contain an example of thing X?
     e.g. Job advertisements (FlipDog).
  2. Is this document/web page of type Y?
     e.g. Course homepages (US government)?
  3. Should we block this page/document?
     e.g. Spam detection, pornography content filtering.
  4. Directed crawling. Active crawling.

Machine Learning Problems for Text/Web Data

• Information Extraction / Entity Tagging / Disambiguation
  1. Get the author/title/company name/course title from this document or web page.
     e.g. Citeseer, WebKB
  2. Find the contact info (phone number/fax/email/etc) for specific people or positions at an organization.
     e.g. MarketIntelligence
  3. Find the salary/location/job title/description for a job posting on the web.
     e.g. FlipDog
  4. Basic idea is documents → databases.
Machine Learning Problems for Text/Web Data

- Searching / Indexing / Collaborative Filtering
  1. Sometimes called “information retrieval”.
  2. Find the most “relevant” document/product/movie/song given these search terms.
     e.g. Google
  3. Find the documents/products most like the list given.
     e.g. Amazon Recommendations
  4. Index collections of crosslinked items into an automatic hierarchy.
     e.g. Citeseer

Machine Learning Techniques for Text/Web Data

- All of the above problems typically involve solving many separate problems in machine learning simultaneously.
  - segmentation
  - classification
  - association
  - clustering
- Why do Machine Learning on Text/Web Data?
  - Classic dream of AI: build a huge knowledge base and use it to reason about the world.
  - Cool machine learning problems.
  - Useful to companies/individuals in the real world.
- How are web documents different?
  They have links and rich formatting.

Document Classification Models

- Basic models for classification:
  1. Naive Bayes
  2. Logistic Regression (MaxEnt)
  3. Support Vector Machines
  4. Decision Trees
  5. Winnow
- More sophisticated models:
  1. Mixtures of Naive Bayes
  2. Latent Probabilistic Semantic Indexing
  3. Latent Dirichlet Allocation
  4. Boosted Decision Trees
### Document Features for Classification

- Binary word occurrence
- Word counts / log counts
- Binary presence on one or more lists of names, cities, companies, states, countries, products, television shows, etc.
- TF-IDF: Term Frequency * Inverse Document Frequency
- Binary indication of “Trigger Phrases”.

### TF-IDF

- The TF-IDF measure counts how many times a word occurs (term-frequency), but normalizes that count by the proportion of documents containing a particular word (inverse-document-frequency).
- A typical measure is:
  \[
  \text{TFIDF}(\text{word}; \text{document}) = n_{\text{wd}} \log \frac{N_{\text{documents}}}{\sum_{d}[n_{\text{wd}} > 0]}
  \]
- Very unusual words have their counts amplified, very common words have their counts multiplied by a very small number.
- Problem: hard to define this measure on a new test case or small test set. Can use IDF from training set, or pooled IDF from training plus testing sets.

### Some Practical Considerations

- Must use stop word lists and stemming.
- Naive Bayes is an excellent model, and often very hard to beat. Always add one to your counts.
- MaxEnt/Logistic Regression: don’t use iterative scaling to update parameters, use conjugate gradient instead. Always use a quadratic prior on the weights.
- Feature selection is key. Some common approaches: max mutual info with class label; most frequent non-stopwords, non-stopwords appearing in most documents.

### Information Extraction Models

- Simple models: sliding windows, boundary finders.
- Use latent variable models where latent variable indicates entity groupings and observables are words or other document features.
- Hidden Markov Models
- Maximum Entropy Markov Models
- Conditional Random Fields
- Voted Perceptron
- Local-Global Models
- More sophisticated tree-based models...
Any of these models can be used to capture words, formatting or both.

**Information Extraction Models**

**Lexicons**
- Abraham Lincoln was born in Kentucky.

**Classify Pre-segmented Candidates**
- Abraham Lincoln was born in Kentucky.

**Sliding Window**
- Abraham Lincoln was born in Kentucky.

**Boundary Models**
- Abraham Lincoln was born in Kentucky.

**Finite State Machines**
- Abraham Lincoln was born in Kentucky.

**Context Free Grammars**
- Abraham Lincoln was born in Kentucky.

**Maximum Entropy Markov Models**
- Abraham Lincoln was born in Kentucky.

**Conditional Random Fields**

- How is this different than the MEMM?
- Normalization is global and not local.

**Labelled vs. Unlabelled Data**
- For information extraction / named-entity tagging, most models require labelled data, which can be very difficult to get in large quantities.
- Such data is often generated by a hand labelling pages and documents.
- Sometimes it is possible to “bootstrap” up from a small amount of labelled data to a larger amount.
  e.g. word sense disambiguation.

- Also known as “logistic regression through time”.

- What are the cliques here?
**Web Searching Models**

- Most web search engines work by using a combination of two technologies:
  - 1. Highly efficient index and search for retrieving a list of pages that contain the keywords you search for, or a set of words that mean roughly the same thing. (This is a databases problem.)
  - 2. A method of ranking the matching results so that the “best” or “most relevant” pages come earlier on the list. (This is a machine learning problem.)
- Virtually all ranking algorithms are eigenvector methods applied to the link matrix of the web.
  - e.g. Google, hubs and authorities
- “Bibliometrics” treats citations in documents like links on the web.

**Recommendation/Collaborative Filtering**

- The most basic recommendation system is table-lookup into the past: given what you already like, recommend things that other people who liked what you do also liked.
- This only works if you have an enormous amount of data (e.g. weather prediction, Amazon).
- In general, we must group documents/products together based on co-occurrence and then extrapolated from our limited database to discover which items to recommend.
  - e.g. Aspect Model

**Summarization**

- Generate a small amount of text that summarizes a larger document.
  - e.g. Google news.
- Very hard problem because the computer has to generate some believable content.
- Easier problem: excerpt a small amount of original text or audio or video that best captures the entire document.

**Evaluation Metrics**

- Perplexity: average number of plausible alternatives on test set.
- Precision vs. Recall curves.
- F-score: \( (\beta^2 + 1)PR/(\beta^2 P + R) \)
- N-best performance.
- Ranking performance.
Example: Person Name Extraction

Person name Extraction

McCallum 2001, unpublished

Example: Features Used

- Capitalized: X
- Mixed Caps: Xx
- All Caps: XXXX
- Initial: X
- Contains Digit: x
- All lowercase: x
- Initial Cap: X
- Punctuation: ..., etc
- Period: .
- Comma: ,
- Apostrophe: '
- Dash: -
- Preceded by HTML tag

Total number of features = ~200k

More Resources

- Data
  - Linguistic Data Consortium (LDC)
    - Penn Treebank, Named Entities, Relations, etc.
      - http://www.biostat.wisc.edu/~craven/ie
      - http://www.cs.umass.edu/~mccallum/data

- Code

- Both
  - http://www.cis.upenn.edu/~adwait/penntools.html
  - http://www.cs.umass.edu/~mccallum/ie