Information Retrieval and Organisation

Chapter 13

Text Classification and Naïve Bayes

Dell Zhang
Birkbeck, University of London
Motivation

- Relevance Feedback revisited
  - The user marks a number of documents as relevant/nonrelevant
  - We then try to use this information to return better search results.
  - Suppose we just tried to learn a filter for nonrelevant documents
  - This is an instance of a text classification problem:
    - Two “classes”: relevant, nonrelevant
    - For each document, decide whether it is relevant or nonrelevant
Motivation

- The path from information retrieval to text classification:
  - You have an information need, say:
    - Unrest in the Niger delta region
  - You want to rerun an appropriate query periodically to find new news items on this topic
  - You will be sent new documents that are found
    - I.e., it’s classification not ranking
- Such queries are called **standing queries**
  - Long used by “information professionals”
  - A modern mass instantiation is
“It’s not the most sophisticated Spam blocker I’ve tried, but it’s the only one that works!”
Motivation

- Many search engine functionalities use classification
- The notion of classification is very general and has many applications within and beyond IR
Text Classification/Categorization

- **Given:**
  - A document, \( d \in D \).
  - A set of classes \( C = \{ c_1, c_2, \ldots, c_n \} \).

- **Determine:**
  - The class of \( d \): \( c(d) \in C \), where \( c(d) \) is a **classification function** (“classifier”).
Text Classification Examples

- Classes are most often topics such as Yahoo-categories
  - e.g., “finance”, “sports”, “news>world>asia>business”
- Classes may be genres
  - e.g., “editorials”, “movie-reviews”, “news”
- Classes may be opinion on a person/product
  - e.g., “like”, “hate”, “neutral”
Text Classification Examples

- Classes may be domain-specific
  - e.g., “interesting-to-me” vs. “not-interesting-to-me”
  - e.g., “contains-adult-language” vs. “doesn’t”
  - e.g., English, French, Chinese, … (language identification)
  - e.g., “about-Linux” vs “not-about-Linux” (vertical search)
  - e.g., “link-spam” vs. “not-link-spam”
Classification Methods (1)

- Manual Classification
  - Used by
    - Yahoo! (originally; now present but downplayed), Looksmart, about.com, ODP, PubMed, …
  - Very accurate when job is done by experts
  - Consistent when the problem size and team is small
  - Difficult and expensive to scale
    - We need *automatic* classification methods for big problems.
Classification Methods (2)

- Hand-Coded Rules
  - Used by
    - CIA, Reuters, CS dept’s spam filter, …
    - Commercial systems for standing queries have complex query languages (everything in IR query languages plus accumulators)
  - Accuracy is often quite high, if the rules have been carefully refined over time by experts.
  - Expensive to build/maintain the rules.
- Companies (such as Verity) provide "IDE" for writing such complex classification rules
- Hand-weighting of terms
- Maintenance issues (author, etc.)
Classification Methods (3)

- (Supervised) Machine Learning
  - Used by
    - Google, Yahoo!, MSN, Autonomy, Verity, Enkata, …
    - Note that many commercial systems use a mixture of methods
  - There is no free lunch: hand-classified training data are required.
  - But the training data can be built up (and refined) easily by amateurs.
    - Such as graduate students 😊
Text Classification via ML

Training Documents \( \rightarrow \) Learning \( \rightarrow \) Classifier \( \rightarrow \) Predicting \( \rightarrow \) Test Documents
Text Classification via ML

Test Data:

Classes:
- ML
- Planning
- Semantics
- Garb.Coll.
- Multimedia
- GUI

Training Data:
- learning
- intelligence
- algorithm
- reinforcement
- network...
- planning
- temporal
- reasoning
- plan
- language...
- programming
- semantics
- language
- proof...
- garbage
- collection
- memory
- optimization
- region...

“planning language proof intelligence”

(Note: in real life there is often a hierarchy, not present in the above problem statement; and also, you may get multi-topic papers for example on ML approaches to Garb. Coll.)
Evaluating Classification

- Classification Accuracy \((\#\text{correct} / \#\text{total})\)
  - The proportion of correct predictions
  - Adequate if one class per document
- Precision, Recall \(\Rightarrow F_1\) measure (for each class)
  - Macro-averaging: computes performance measure for each class, and then computes a simple average over classes.
  - Micro-averaging: pools per-document predictions across classes, and then computes performance measure on the pooled contingency table.
Evaluating Classification

<table>
<thead>
<tr>
<th></th>
<th>class 1 truth: yes</th>
<th>class 1 truth: no</th>
<th>class 2 truth: yes</th>
<th>class 2 truth: no</th>
</tr>
</thead>
<tbody>
<tr>
<td>call: yes</td>
<td>10</td>
<td>10</td>
<td>90</td>
<td>10</td>
</tr>
<tr>
<td>call: no</td>
<td>10</td>
<td>970</td>
<td>10</td>
<td>890</td>
</tr>
</tbody>
</table>

macro-averaged precision is \[
\frac{10/(10 + 10) + 90/(10 + 90)}{2} = (0.5 + 0.9)/2 = 0.7
\]

<table>
<thead>
<tr>
<th></th>
<th>pooled table truth: yes</th>
<th>pooled table truth: no</th>
</tr>
</thead>
<tbody>
<tr>
<td>call: yes</td>
<td>100</td>
<td>20</td>
</tr>
<tr>
<td>call: no</td>
<td>20</td>
<td>1860</td>
</tr>
</tbody>
</table>

micro-averaged precision is \[
100/(100 + 20) \approx 0.83
\]
Evaluating Classification

- Evaluation must be done on test data that are independent of the training data (usually a disjoint set of instances).
- Results can vary based on sampling error due to different training and test sets.
- Average results over multiple training and test sets (splits of the overall data) for the best results.

Reuters-21578
Learning Curve

NaiveBayes: 10-fold CV Learning Curve

Yahoo Science Data
Naïve Bayes

- Before seeing the content of document $d$
  - Classify $d$ to the class with maximum prior probability $P(c)$.
- After seeing the content of document $d$
  - Classify $d$ to the class with maximum posteriori probability $P(c|d)$.
  - For each class $c_j \in C$, $P(c_j|d)$ can be estimated using the Bayes’ Rule.
Naïve Bayes

- Bayes’ Rule, Again!

\[
P(c, d) = P(c \mid d)P(d) = P(d \mid c)P(c)
\]

\[
P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}
\]
Naïve Bayes

\[ c(d) = \arg\max_{c_j \in C} P(c_j | d) \]

\[ = \arg\max_{c_j \in C} \frac{P(d | c_j) P(c_j)}{P(d)} \]

\[ = \arg\max_{c_j \in C} P(d | c_j) P(c_j) \]

How can we estimate?
Naïve Bayes

For each class \( c_j \in C \), \( P(c_j) \) can be estimated from the frequency of classes in the training data.

\[
P(c_j) = \frac{N_j}{\sum_j N_j}
\]

where \( N_j \): the number of documents in the class \( c_j \)
Naïve Bayes

- \[ P(d|c_j) = P(t_1, t_2, ..., t_n|c_j) \]
  - There are \( O(|X|^n \cdot |C|) \) parameters.
  - Could only be estimated if a very, very large number of training examples was available.

To facilitate the estimation of \( P(d|c_j) \), two simplifying assumptions are made.

- **Conditional Independence Assumption**
  - The term occurrences are independent of each other given the class.

- **Positional Independence Assumption**
  - The conditional probabilities for a term are the same independent of position in the document.
Naïve Bayes

- Multinomial NB: effectively, the probability of each doc $P(d|c_j)$ is given by a class-specific unigram language model.

$$P(d|c_j) = \prod_{t_i \in d} P(t_i|c_j)$$
Smoothing for NB

- Why not just use MLE?
  - If a term $t$ (in a test doc $d$) did not occur in the training data, $P(t|c_j)$ would be 0, and then $P(d|c_j)$ would be 0 no matter how strongly other terms in $d$ are associated with class $c_j$.

- Add-One (Laplace) Smoothing

$$P(t_i | c_j) = \frac{T_{ji}}{\sum_i T_{ji}}$$

$$P(t_i | c_j) = \frac{(T_{ji} + 1)}{\sum_i (T_{ji} + 1)}$$

$T_{ji}$: the number of occurrences of term $i$ in documents of class $c_j$
Underflow Prevention

- Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow.
- Since $\log(xy) = \log(x) + \log(y)$, it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities.
- Class with highest final un-normalized log probability score is still the most probable.

$$c_{NB} = \arg\max_{c_j \in C} \left\{ \log P(c_j) + \sum_{i \in \text{positions}} \log P(t_i | c_j) \right\}$$

Note that the model is now just max of sum of weights…
**NB Algorithm: Training**

\[
\text{TrainMultinomialNB}(\mathcal{C}, \mathcal{D})
\]

1. \( V \leftarrow \text{ExtractVocabulary}(\mathcal{D}) \)
2. \( N \leftarrow \text{CountDocs}(\mathcal{D}) \)
3. For each \( c \in \mathcal{C} \) do
   4. \( N_c \leftarrow \text{CountDocsInClass}(\mathcal{D}, c) \)
   5. \( \text{prior}[c] \leftarrow N_c/N \)
   6. \( \text{text}_c \leftarrow \text{ConcatenateTextOfAllDocsInClass}(\mathcal{D}, c) \)
4. For each \( t \in V \) do
   8. \( T_{ct} \leftarrow \text{CountTokensOfTerm}(\text{text}_c, t) \)
   9. For each \( t \in V \) do
   10. \( \text{condprob}[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)} \)
5. Return \( V, \text{prior}, \text{condprob} \)
NB Algorithm: Testing

\begin{algorithm}
\textbf{ApplyMultinomialNB}(\mathcal{C}, V, prior, cond\,prob, d)
1. $W \leftarrow \text{ExtractTokensFromDoc}(V, d)$
2. \textbf{for each} $c \in \mathcal{C}$
3. \hspace{1em} \textbf{do} $score[c] \leftarrow \log prior[c]$
4. \hspace{1em} \textbf{for each} $t \in W$
5. \hspace{2em} \textbf{do} $score[c] \leftarrow score[c] + \log cond\,prob[t][c]$
6. \textbf{return} $\arg \max_{c \in \mathcal{C}} score[c]$
\end{algorithm}
Time Complexity

- Training Time: $O(|D|L_d + |C||V|)$
  where $L_d$ is the average length of a document in $D$.
  - Assumes $V$ and all $D_i$, $n_i$, and $n_{ij}$ pre-computed in $O(|D|L_d)$ time during one pass through all of the data.
  - Generally just $O(|D|L_d)$ since usually $|C||V| < |D|L_d$

- Testing Time: $O(|C|L_t)$
  where $L_t$ is the average length of a test document.

Very efficient overall, linearly proportional to the time needed to just read in all the data.
Naïve Bayes is Not So Naïve

- Effectiveness
  - The Bayes *optimal* classifier if the independence assumptions do hold.
  - Often performs well even if the independence assumptions are badly violated.
  - Robust to irrelevant features.
  - Good in domains with many *equally important* features.
  - A good dependable baseline for text classification (though may not be the best).
Naïve Bayes is Not So Naïve

- **Efficiency**
  - Very fast
    - Linear training/testing time complexity
    - One pass of counting over the data
  - Low storage requirements.
Application: Web Page Cat.

- **WebKB Experiment (1998)**
  - Classify webpages from CS departments into:
    - student, faculty, course, project
  - Train on ~5,000 hand-labeled web pages
    - Cornell, Washington, U.Texas, Wisconsin
  - Crawl and classify a new site (CMU)

<table>
<thead>
<tr>
<th></th>
<th>Student</th>
<th>Faculty</th>
<th>Person</th>
<th>Project</th>
<th>Course</th>
<th>Departmt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extracted</td>
<td>180</td>
<td>66</td>
<td>246</td>
<td>99</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>Correct</td>
<td>130</td>
<td>28</td>
<td>194</td>
<td>72</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>Accuracy:</td>
<td>72%</td>
<td>42%</td>
<td>79%</td>
<td>73%</td>
<td>89%</td>
<td>100%</td>
</tr>
</tbody>
</table>
Application: Email Filtering

- Naïve Bayes has found a home in spam filtering
  - Paul Graham’s *A Plan for Spam*
    - A mutant with more mutant offspring ...
    - Naive Bayes-like classifier with weird parameter estimation
  - Widely used in spam filters
    - Classic Naive Bayes superior when appropriately used (According to David D. Lewis)
    - But also many other things: black hole lists, etc.
- Many email topic filters also use NB classifiers
Application: Direct Marketing

- **KDD-CUP** 97 competition
  - Task: to predict if the recipient of mail will actually respond to the advertisement
    - Financial services industry
    - 750,000 records
  - Naïve Bayes: the 1st & 2nd place in among 16 (then) state-of-the-art algorithms.