Lecture 7

Text Categorisation
What is Text Categorisation?

• Given:

1. a description of a document \( x \in X \)
2. a fixed set of categories \( C = \{ c_1, c_2, \ldots, c_n \} \)

• Determine:

the category of \( x : c(x) \in C \), where \( c(x) \) is a categorisation function whose domain is \( X \) and whose range is \( C \).

• That is, how do we build categorization functions (classifiers) which can operate over an arbitrary description language and within an arbitrary category space?

Is This Really Something I Want to Read?

Subject: BUSINESS INVESTMENT
From: COLLINS JAMES <collins55ng@yahoo.co.in>

Attention: President/Director,

I am the chairman of the contract award committee of the Gold and Natural resources ministry here in Dakar Senegal, for security reasons, I may not wish to disclose the most important thing for now until I hear from you.

After due deliberation with my partner, I decided to forward to you this business proposal, we want you to assist us receive the sum of Twenty eight million, six hundred thousand united state bills ($28.6M) into your account.

:(
Example Applications of Text Categorisation

- Assign labels to a document, e.g.:
  - Yahoo!-style topic label (e.g. sport, news > world > asia > business)
  - genre (e.g. job listing, news)
  - spam vs. not-spam
  - contains-adult-language vs. conforms-to-Bush-sensibilities

- Determine the authorship of a paper

- Document routing

• Indexing (digital libraries, etc.)

• Sorting

• Identifying e-scams
Methods of Categorisation

- **Manual categorisation**
  - very accurate when the job is done by experts
  - consistent when the problem size and team are small
  - difficult and expensive to scale

- **Handcrafted rule-based systems**
  - e.g., assign a particular category if the document contains a given boolean combination of words
  - accuracy is often very high if a rule has been carefully refined over time by a subject expert
  - building and maintaining these rules is very expensive

• Automatic classifiers
  ★ $k$-Nearest Neighbours ($k$-NN)
  ★ Naive Bayes
  ★ support vector machines
  ★ presuppose hand-classified seed data to generalise from

• In practise, commercial systems tend to use bits and pieces of all of these
  ★ Yahoo!, Google Directory (dmoz.com), Reuters, ...
What are (Supervised) Classifiers?

- Given:
  1. a fixed representation language of attributes
  2. a fixed set of pre-classified training instances
  3. a fixed set of classes $C$
  4. a “learner” algorithm which can identify patterns in the training instances

- Estimate:
  
  the category of a novel input $x : c(x) \in C$

Training data
Learner
Classifier
Test data
Test instance
?
A
B
B
Classification
A

Example Set-up

- **Training data:**

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>FALSE</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>high</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>high</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>normal</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>normal</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>overcast</td>
<td>cool</td>
<td>normal</td>
<td>TRUE</td>
<td>yes</td>
</tr>
</tbody>
</table>

- **Test data:**

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>mild</td>
<td>normal</td>
<td>TRUE</td>
<td>???</td>
</tr>
</tbody>
</table>
Supervision

- **Supervised** methods have prior knowledge of a closed set of classes and instances pre-classified according to those classes, and set out to categorise new instances according to those classes.

- **Unsupervised** methods dynamically discover the classes in the process of categorising the instances [STRONG DEFINITION]

  *OR*

- **Unsupervised** methods categorise instances without the aid of pre-classified data [WEAK DEFINITION]
Discussion: supervised or unsupervised?

- Given a set of web documents, identify obvious outliers for manual inspection

- From a set of web documents, filter out the spam documents based on a sample of manually-classified documents

- Classify a set of web documents as belonging to SCC, IN, OUT or OTHER
Discussion: supervised or unsupervised?

- Given a set of web documents, identify obvious outliers for manual inspection
  
  (strongly) unsupervised

- From a set of web documents, filter out the spam documents based on a sample of manually-classified documents
  
  supervised

- Classify a set of web documents as belonging to SCC, IN, OUT or OTHER
  
  (weakly) unsupervised
... But First some Basics

- Document representation
- Basics of probability theory
- Basics of entropy
Document Representation

<table>
<thead>
<tr>
<th>id</th>
<th>word</th>
<th>freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>233</td>
</tr>
<tr>
<td>2</td>
<td>aardvark</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>aback</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>abacus</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>word</th>
<th>freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>abalone</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>abandon</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>abandonment</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>:</td>
</tr>
</tbody>
</table>

\[ \vec{x} = \langle 233, 0, 2, 0, 0, 1, 0, \ldots \rangle \]

- In practice, we tend to weight each term frequency (see later), and also pre-normalise the document vector to unit length
(Very) Basics of Probability Theory

- **Joint probability** \( P(A, B) \): the probability of both \( A \) and \( B \) occurring = \( P(A \cap B) \)

\[
P(\text{ace, heart}) = \frac{1}{52}, \quad P(\text{heart, red}) = \frac{1}{4}
\]
- **Conditional probability** \( P(A|B) \): the probability of \( A \) occurring given the occurrence of \( B \) is computed as:

\[
P(A|B) = \frac{P(A \cap B)}{P(B)}
\]

- Example: 
  - \( P(\text{ace}|\text{heart}) = \frac{1}{13} \),  
  - \( P(\text{heart}|\text{red}) = \frac{1}{2} \)
• **Multiplication rule**: \( P(A \cap B) = P(A|B)P(B) = P(B|A)P(A) \)

• **Chain rule**: \( P(A_1 \cap ... \cap A_n) = P(A_1)P(A_2|A_1)P(A_3|A_2 \cap A_1) ... P(A_n|\cap_{i=1}^{n-1} A_i) \)

• **Prior probability** \( (P(A))\): the probability of \( A \) occurring, given no additional knowledge about \( A \)

• **Posterior probability** \( (P(A|B))\): the probability of \( A \) occurring, given background knowledge about event(s) \( B \) leading up to \( A \)

• **Independence**: \( A \) and \( B \) are independent iff \( P(A \cap B) = P(A)P(B) \)
Binomial Distributions

- A **binomial distribution** results from a series of independent trials with only two outcomes (i.e. **Bernoulli trials**)
  
  e.g. multiple coin tosses (\(\{H, T, H, H, \ldots, T\}\))

- The probability of an event with probability \(p\) occurring exactly \(m\) out of \(n\) times is given by

  \[
  P(m, n, p) = \frac{n!}{m!(n - m)!}p^m(1 - p)^{n-m}
  \]
Binomial Example: $P(m, 10, p = 0.1)$
Multinomial Distributions

- A **multinomial distribution** results from a series of independent trials with more than two outcomes

  e.g. balls in cricket (\(\cdot, \cdot, 1, \text{out}_{LBW}, \ldots, 4\))

- The probability of events \(X_1, X_2, \ldots, X_n\) with probabilities \(p_1, p_2, \ldots, p_n\) occurring exactly \(x_1, x_2, \ldots, x_n\) times, respectively, is given by

  \[
  P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = \left(\sum_i x_i\right)! \prod_i \frac{p_i^{x_i}}{x_i!}
  \]
Entropy

- Given a probability distribution, the information (in bits) required to predict an event is the distribution’s entropy or information value.

- The entropy of a discrete random event $x$ with possible states $1, \ldots, n$ is:

$$H(x) = -\sum_{i=1}^{n} P(i) \log_2 P(i)$$

$$= \frac{\text{freq}(\ast) \log_2(\text{freq}(\ast)) - \sum_{i=1}^{n} \text{freq}(i) \log_2(\text{freq}(i))}{\text{freq}(\ast)}$$

where $0 \log_2 0 = \text{def} \ 0$
Interpreting Entropy Values

- A high entropy value means $x$ is boring (uniform/flat)
- A low entropy value means $x$ is varied ("peaky")
Entropy of Loaded Dice (1)

\[ H(x) = 2.58 \]

[Graph showing distribution of probabilities for loaded dice]
Entropy of Loaded Dice (2)

\[ H(x) = 2.16 \]
Entropy of Loaded Dice (3)

\[ H(x) = 2.32 \]
Entropy of Loaded Dice (4)

\[ H(x) = 1.56 \]
Entropy of Loaded Dice (5)

$$H(x) = 0$$
Estimating the Probabilities

- The most obvious way of generating the probabilities is via **maximum likelihood estimation** (MLE), using the frequency counts in the training data:

\[
\hat{P}(c_j) = \frac{\text{freq}(c_j)}{\sum_k \text{freq}(c_k)}
\]

\[
\hat{P}(x_i|c_j) = \frac{\text{freq}(x_i, c_j)}{\text{freq}(c_j)}
\]
• Based on this, our document representation would look something like:

\[ \bar{x} = \langle 0.1, 0, 0.0002, 0, 0, 0.0001, 0, \ldots \rangle \]
Modelling Document Similarity: Cosine Similarity

- Given two documents $x$ and $y$, and their corresponding feature vectors $\vec{x}$ and $\vec{y}$, respectively, we can calculate their similarity via their vector cosine:

$$\text{sim}(x, y) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| |\vec{y}|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i y_i^2}}$$

Cosine Similarity Example

- Calculate the cosine similarity of the following documents:

$A = \begin{bmatrix} \text{aardvark} & \text{back} & \text{abandon} \\ \text{abandon} & \text{abandon} \end{bmatrix}$

$B = \begin{bmatrix} \text{aardvark} & \text{abandonment} \\ \text{back} & \text{back} & \text{back} & \text{back} \end{bmatrix}$

$\vec{A} = \langle 1, 3, 0, 1 \rangle$

$= \langle \frac{1}{\sqrt{11}}, \frac{3}{\sqrt{11}}, 0, \frac{1}{\sqrt{11}} \rangle$

$\vec{B} = \langle 1, 0, 1, 3 \rangle$

$= \langle \frac{1}{\sqrt{11}}, 0, \frac{1}{\sqrt{11}}, \frac{3}{\sqrt{11}} \rangle$

$\vec{A} \cdot \vec{B} = \frac{1}{\sqrt{11}} \times \frac{1}{\sqrt{11}} + \frac{3}{\sqrt{11}} \times 0 + 0 \times \frac{1}{\sqrt{11}} + \frac{1}{\sqrt{11}} \times \frac{3}{\sqrt{11}} = \frac{4}{11}$
Modelling Document Distance: Relative Entropy

- Given two documents $x$ and $y$, and their corresponding feature unit-length vectors $\vec{x}$ and $\vec{y}$, respectively, we can interpret the feature vector as a probability distribution and calculate the relative entropy (or KL divergence):

$$D(x \mid\mid y) = \sum_i x_i \left(\log_2 x_i - \log_2 y_i\right)$$

or alternatively skew divergence:

$$s_{\alpha}(x, y) = D(x \mid\mid \alpha y + (1 - \alpha)x)$$
• This causes considerable grief for our MLE-based probabilities: why?

• A simplistic way of getting around this is via \textbf{Laplacian smoothing}:

\[
\hat{P}(c_j) = \frac{freq(c_j) + 1}{k + \sum_k freq(c_k)}
\]

\[
\hat{P}(x_i|c_j) = \frac{freq(x_i, c_j) + 1}{freq(c_j) + l}
\]
Relative Entropy Example

- Calculate the relative entropy and skew divergence of the following documents:

<table>
<thead>
<tr>
<th>aardvark back abandon</th>
<th>abandon abandon</th>
</tr>
</thead>
<tbody>
<tr>
<td>abandon abandon</td>
<td>aardvark abandonment</td>
</tr>
<tr>
<td>back back back</td>
<td>back back back</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
A &= \begin{pmatrix} 2 & 4 & 1 & 2 \\ 9 & 9 & 9 & 9 \end{pmatrix} \\
B &= \begin{pmatrix} 2 & 1 & 2 & 4 \\ 9 & 9 & 9 & 9 \end{pmatrix}
\end{align*}
\]

\[
D(A \| B) = \sum_i a_i (\log_2 a_i - \log_2 b_i)
\]
\[
\begin{align*}
&= \frac{2}{9}(\log \frac{2}{9} - \log \frac{2}{9}) + \frac{4}{9}(\log \frac{4}{9} - \log \frac{1}{9}) + \\
&\quad \frac{1}{9}(\log \frac{1}{9} - \log \frac{2}{9}) + \frac{2}{9}(\log \frac{2}{9} - \log \frac{4}{9}) \\
&\approx 0.56
\end{align*}
\]
Skew Divergence Example

- Calculate the relative entropy and skew divergence of the following documents:

<table>
<thead>
<tr>
<th>aardvark back abandon</th>
<th>aardvark abandonment back back back</th>
</tr>
</thead>
<tbody>
<tr>
<td>abandon</td>
<td>back back back</td>
</tr>
</tbody>
</table>

\[
A = \langle 0.2, 0.6, 0.0, 0.2 \rangle \quad B = \langle 0.2, 0.0, 0.2, 0.6 \rangle
\]

\[
s_{0.99}(A, B) = D(A \mid 0.99A + 0.01B) = \sum_i a_i (\log a_i - \log(0.99b_i + 0.01a_i))
\]
\[
\begin{align*}
\approx & \quad 0.2(\log 0.2 - \log(0.99 \times 0.2 + 0.01 \times 0.2)) + \\
& \quad 0.6(\log 0.6 - \log(0.99 \times 0.0 + 0.01 \times 0.6)) + \\
& \quad 0.0(\log 0.0 - \log(0.99 \times 0.2 + 0.01 \times 0.0)) + \\
& \quad 0.2(\log 0.2 - \log(0.99 \times 0.6 + 0.01 \times 0.2)) \\
\approx & \quad 3.67
\end{align*}
\]
Nearest Neighbour Classifiers

- There are various ways to combine these document–document scores to form an overall categorisation function, e.g.:

- **Method 1:** index all training documents, and query the training document set with each test document; classify the test document according to the class of the top-ranked training document [1-NN]

- **Method 2:** index all training documents, and query the training document set with each test document; classify the test document according to the majority class within the \(k\) top-ranked training documents [k-NN]

• **Method 3:** index all training documents, and query the training document set with each test document; classify the test document according to the class with the best accumulative score *[weighted k-NN]*

• **Method 4:** index all training documents, and query the training document set with each test document; classify the test document according to the class with the best accumulative score based on scores, factoring in an offset to indicate the prior expectation of a test document being classified as being a member of that class *[offset weighted k-NN]*
• Overall advantages of the nearest neighbour approach:
  ★ simple

• Overall disadvantages of the nearest neighbour approach:
  ★ expensive (in terms of index accesses)
  ★ everything is done at run time (lazy learner)
  ★ prone to bias
  ★ arbitrary $k$ value

Feature Selection
Feature Selection

- Classes will tend to have medium-frequency membership, suggesting that we need some extra mechanism of identifying the terms which best discriminate the classes
  
  → enter feature selection

- We will focus on greedy inclusion algorithms for feature selection:
  
  rank terms in descending order of class discrimination, and select the top $N$ features

Chakrabarti (2003:pp136–47)
Feature Selection via MI

- **Mutual information** (MI) provides an information-based estimate of the (in)dependence of two discrete random variables $T$ (term) and $C$ (class):

\[
MI(T, C) = \sum_{t \in \{0, 1\}} \sum_{c} P(t, c) \log \frac{P(t, c)}{P(t)P(c)}
\]

- If $T$ and $C$ are independent, $MI(T, C) = 0$

- If $T$ and $C$ are positively correlated, $MI(T, C) > 0$

- If $T$ and $C$ are negatively correlated, $MI(T, C) < 0$

Chakrabarti (2003:pp139–40)
- We select our $N$ “best” features by taking the terms $T$ with the highest MI value

- The method is greedy in that it doesn’t take term inter-dependence into consideration

- Bias towards rare uninformative terms

- Example features (on 20 Newsgroups):
  
  sci.electronics: circuit, voltage, amp, ground, copy, battery, electronics, cooling, ...
  rec.autos: car, cars, engine, ford, dealer, mustang, oil, collision, autos, tires, toyota, ...

Chakrabarti (2003:pp139–40)
Mutual Information Example

- Perform feature selection over a document collection with the following characteristics:

<table>
<thead>
<tr>
<th>Term</th>
<th>Class A</th>
<th>Class B</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1,000</td>
<td>1,000</td>
<td>2,000</td>
</tr>
<tr>
<td>aardvark</td>
<td>30</td>
<td>50</td>
<td>80</td>
</tr>
<tr>
<td>aback</td>
<td>10</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>abacus</td>
<td>100</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>TOTAL</td>
<td>1,000</td>
<td>1,000</td>
<td>2,000</td>
</tr>
</tbody>
</table>

Chakrabarti (2003:pp139–40)
\[ P(A) = \frac{1000}{2000} \]
\[ P(\text{aback}) = \frac{30}{2000} \]
\[ P(\text{aback}, A) = \frac{10}{2000} \]
\[ P(\text{aback}, B) = \frac{20}{2000} \]
\[ P(B) = \frac{1000}{2000} \]
\[ P(\text{aback}) = \frac{1970}{2000} \]
\[ P(\text{aback}, A) = \frac{990}{2000} \]
\[ P(\text{aback}, B) = \frac{980}{2000} \]

\[ MI(\text{aback}) = \frac{990}{2000} \log \frac{990}{1970\ 1000} + \frac{980}{2000} \log \frac{980}{1970\ 1000} \]
\[ + \frac{10}{2000} \log \frac{10}{30\ 1000} + \frac{20}{2000} \log \frac{20}{30\ 1000} \]

\[ = 0.0009 \]

Chakrabarti (2003: pp139–40)
Feature Selection via $\chi^2$

- $\chi^2$ ("kai-square") provides an estimate of the level of statistical significance of the correlation between two discrete random variables $T$ (term) and $C$ (class):

<table>
<thead>
<tr>
<th></th>
<th>Term present</th>
<th>Term absent</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{class} = c_i$</td>
<td>$W$</td>
<td>$X$</td>
</tr>
<tr>
<td>$\text{class} \neq c_i$</td>
<td>$Y$</td>
<td>$Z$</td>
</tr>
</tbody>
</table>

$$
\chi^2 = \frac{N(WZ - XY)^2}{(W + X)(W + Y)(X + Z)(Y + Z)}
$$

Chakrabarti (2003: pp138–9)
• The higher the value of $\chi^2$, the less confident we are of $T$ and $c_i$ being independent

• We select our $N$ “best” features by taking the terms $T$ with the highest $\chi^2$ value

• Bias towards frequent uninformative terms

Chakrabarti (2003:pp138–9)
**\( \chi^2 \) Example**

- Perform feature selection over a document collection with the following characteristics:

<table>
<thead>
<tr>
<th>Term</th>
<th>Class A</th>
<th>Class B</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1,000</td>
<td>1,000</td>
<td>2,000</td>
</tr>
<tr>
<td>aardvark</td>
<td>30</td>
<td>50</td>
<td>80</td>
</tr>
<tr>
<td>aback</td>
<td>10</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>abacus</td>
<td>100</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>TOTAL</td>
<td>1,000</td>
<td>1,000</td>
<td>2,000</td>
</tr>
</tbody>
</table>

Chakrabarti (2003:pp138–9)
\[
\begin{array}{cc}
aback & \overline{aback} \\
A & 10 & 990 \\
\overline{A} & 20 & 980 \\
\end{array}
\]

\[
\chi^2 = \frac{2000(10 \times 980 - 990 \times 20)^2}{(10 + 990)(10 + 20)(990 + 980)(20 + 980)}
= 3.38
\]
Summary

- What are the basic methods of text categorisation?
- What is the different between supervised and unsupervised classifiers?
- How does the $k$-nearest neighbour method operate, and what are some of the variants on the original algorithm?
- What different methods are used to calculate document similarity?
- How can we perform feature selection?
References


