Bag of Word Models

Bonan Min
bonanmin@gmail.com

Some slides are based on class materials from Ralph Grishman, Thien Huu Nguyen
Bag of Words Models

When do we need elaborate linguistic analysis?

Look at NLP applications

- document retrieval (a.k.a., information retrieval)
- opinion mining
- association mining

See how far we can get with document-level bag-of-words models

- and introduce some of our mathematical approaches
Application 1: Information Retrieval

Task: given query = list of keywords, identify and rank relevant documents from collection

Basic idea:
  ◦ Find documents whose set of words most closely matches words in query
Vector Space Model

Suppose the document collection has $n$ distinct words, $w_1, \ldots, w_n$

Each document is characterized by an $n$-dimensional vector whose $i^{th}$ component is the frequency of word $w_i$ in the document

Example

- $D1 = [\text{The cat chased the mouse.}]$
- $D2 = [\text{The dog chased the cat.}]$
- $W = [\text{The, chased, dog, cat, mouse}]$  (n = 5)
- $V1 = [2, 1, 0, 1, 1]$
- $V2 = [2, 1, 1, 1, 0]$
Weighting the Components

Unusual words like *elephant* determine the topic much more than common words such as “the” or “have”

- can ignore words on a *stop list* or
- weight each term frequency $t_{fi}$ by its inverse document frequency $idf_i$

\[
idf_i = \log\left( \frac{N}{n_i} \right)
\]

\[
w_i = tf_i \times idf_i
\]

where $N = \text{size of collection}$ and $n_i = \text{number of documents containing term } i$
Cosine Similarity

Define a similarity metric between topic vectors.

A common choice is *cosine similarity* (normalized dot product):

$$\text{sim}(A, B) = \frac{\sum_i a_i \times b_i}{\sqrt{\sum_i a_i^2} \times \sqrt{\sum_i b_i^2}}$$

The cosine similarity metric is the cosine of the angle between the term vectors.
Verdict: a Success

For heterogeneous text collections, the vector space model, tf-idf weighting, and cosine similarity have been the basis for successful document retrieval for over 50 years

- stemming required for some languages
- limited resolution: returns documents, not answers
Application 2: Opinion Mining

Task: judge whether a document expresses a positive or negative opinion (or no opinion) about an object or topic
- *classification* task
- valuable for producers/marketers of all sorts of products

Simple strategy: rule-based approach
- make lists of positive and negative words
- see which predominate in a given document (and mark as ‘no opinion’ if there are few words of either type)
- problem: hard to make such lists
- hard to switch to different domains/labels/languages

<table>
<thead>
<tr>
<th>Training Examples</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simply loved it</td>
<td>Positive</td>
</tr>
<tr>
<td>Most disgusting food I have ever had</td>
<td>Negative</td>
</tr>
<tr>
<td>Stay away, very disgusting food!</td>
<td>Negative</td>
</tr>
<tr>
<td>Menu is absolutely perfect, loved it!</td>
<td>Positive</td>
</tr>
<tr>
<td>A really good value for money</td>
<td>Positive</td>
</tr>
<tr>
<td>This is a very good restaurant</td>
<td>Positive</td>
</tr>
<tr>
<td>Terrible experience!</td>
<td>Negative</td>
</tr>
<tr>
<td>This place has best food</td>
<td>Positive</td>
</tr>
<tr>
<td>This place has most pathetic serving food!</td>
<td>Negative</td>
</tr>
</tbody>
</table>
Training a Classification Model

Supersede the rule-based approach

- A generic (task-independent) learning algorithm to train a classifier/function/model from a set of labeled examples
- The classifier learns, from these labeled examples, the characteristics of a new text should have in order to be assigned to some label

Advantages

- Annotating/locating training examples is cheaper than writing rules
- Easier updates to changing conditions (annotate more data with new labels for new domains)
Naive Bayes Classification

Identify most likely class

\[ s = \arg\max_t P(t | W) \]
\[ t \in \{\text{pos, neg}\} \]

Use Bayes’ rule

\[
\arg\max_t P(t | W) = \frac{\arg\max_t P(W | t)P(t)}{P(W)}
\]
\[ = \arg\max_t P(W | t)P(t) \]
\[ = \arg\max_t P(w_1, ..., w_n | t)P(t) \]
\[ = \arg\max_t \prod_i P(w_i | t)P(t) \]

Doesn’t change if changing \( t \), so we’re going to drop it

Based on the naïve assumption of independence of the word probabilities
Training

Estimate probabilities from the training corpus (N documents) using maximum likelihood estimators

\[
P(t) = \frac{\text{count (docs labeled } t\text{)}}{N}
\]

\[
P(w_i | t) = \frac{\text{count (docs labeled } t\text{ containing } w_i\text{)}}{\text{count (docs labeled } t\text{)}}
\]
Text Classification: Flavors

Bernoulli model: use presence (/ absence) of a term in a document as feature
- *formulas on previous slide*

Multinomial model: based on frequency of terms in documents:
- \[ P(t) = \frac{\text{total length of docs labeled } t}{\text{total size of corpus}} \]
- \[ P(w_i | t) = \frac{\text{count (instances of } w_i \text{ in docs labeled } t)}{\text{total length of docs labeled } t} \]

Better performance on long documents
Text Classification: Flavors

Bernoulli model: use presence (/ absence) of a term in a document as feature
  ◦ formulas on previous slide

Multinomial model: based on frequency of terms in documents:
  ◦ \( P( t ) = \frac{\text{total length of docs labeled } t}{\text{total size of corpus}} \)

  ◦ \( P( w_i | t ) = \frac{\text{count (instances of } w_i \text{ in docs labeled } t)}{\text{total length of docs labeled } t} \)

  Better performance on long documents
The Importance of Smoothing

Suppose a glowing review SLP2 (with lots of positive words) includes one word, “mathematical”, previously seen only in negative reviews

\[
P(\text{positive} | \text{SLP2}) = 0
\]

because \(P(\text{“mathematical”} | \text{positive}) = 0\)

The maximum likelihood estimate is poor when there is very little data

We need to ‘smooth’ the probabilities to avoid this problem
Add-One (Laplace) Smoothing

A simple remedy is to add 1 to each count
- for the conditional probabilities $P( w \mid t )$: Add 1 to each $c(w, t)$
- Increase the denominator by number of unique words ($|V|$). That is, add $|V|$ to $c(t)$ to keep them as probabilities (sum up to 1)

$$\sum_{w \in V} p(w|t) = 1$$
An Example

$$P(t) = \frac{N_t}{N}$$

$$P(w \mid t) = \frac{\text{count}(w, t) + 1}{\text{count}(t) + |V|}$$

<table>
<thead>
<tr>
<th>Doc</th>
<th>Words</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>1 Chinese Beijing Chinese</td>
<td>c</td>
</tr>
<tr>
<td>2</td>
<td>Chinese Chinese Shanghai</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>Chinese Macao</td>
<td>c</td>
</tr>
<tr>
<td>4</td>
<td>Tokyo Japan Chinese</td>
<td>j</td>
</tr>
<tr>
<td>Test</td>
<td>5 Chinese Chinese Chinese Tokyo Japan</td>
<td>?</td>
</tr>
</tbody>
</table>

Priors:
$$P(t) = \frac{3}{4}$$
$$P(j) = \frac{1}{4}$$

Conditional Probabilities:
$$P(\text{Chinese} \mid c) = \frac{5+1}{8+6} = \frac{6}{14} = \frac{3}{7}$$
$$P(\text{Tokyo} \mid c) = \frac{0+1}{8+6} = \frac{1}{14} \leftarrow 0$$
$$P(\text{Japan} \mid c) = \frac{0+1}{8+6} = \frac{1}{14} \leftarrow 0$$
$$P(\text{Chinese} \mid j) = \frac{1+1}{3+6} = \frac{2}{9}$$
$$P(\text{Tokyo} \mid j) = \frac{1+1}{3+6} = \frac{2}{9}$$
$$P(\text{Japan} \mid j) = \frac{1+1}{3+6} = \frac{2}{9}$$

Choosing a class:
$$P(c \mid d5) \propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003 \leftarrow 0$$
$$P(j \mid d5) \propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001$$

$$P(t)$$  $$P(w \mid t)$$
Some Useful Resources Using NLTK

Sentiment Analysis with Python NLTK Text Classification

- [http://text-processing.com/demo/sentiment/](http://text-processing.com/demo/sentiment/)

NLTK Code (simplified classifier)

Problems with Bag-of-Words Models

Ambiguous terms: is “low” a positive or a negative term?
- “low” can be positive: “low price”
- or negative: “low quality”

Negation: How to handle “the equipment never failed”? A trick:
- modify words following negation
  “the equipment never NOT_failed”
- treat them as a separate ‘negated’ vocabulary

How far can this trick go?
e.g., “the equipment never failed and was cheap to run”
  → “the equipment never NOT_failed NOT_and NOT_was NOT_cheap NOT_to
     NOT_run”
  have to determine scope of negation
Verdict: Mixed

A simple bag-of-words strategy with a NB model works quite well for simple reviews referring to a single item

- Very fast, low storage requirements
- Robust to irrelevant features
  - Irrelevant features cancel each other without affecting results
- Very good in domains with many equally important features
- Optimal if the independence assumptions hold
  - If assumed independence is correct, then it is the Bayes Optimal Classifier for problem

but fails

- for ambiguous terms
- for negation
- for comparative reviews
- to reveal aspects of an opinion
  - *the car looked great and handled well, but the wheels kept falling off*
Application 3: Association Mining

Goal: find interesting relationships among attributes of an object in a large collection ...

Objects with attribute A also have attribute B
  ◦ e.g., “people who bought A also bought B”

For text: documents with term A also have term B
  ◦ widely used in scientific and medical literature
Bag-of-Words

Simplest approach
- look for words $x$ and $y$ for which frequency ($x$ and $y$ in same document) $>>$ frequency of $x$ * frequency of $y$
- Or use Mutual Information:

$$pmi(x; y) \equiv \log \frac{p(x, y)}{p(x)p(y)}$$

Doesn’t work well
- want to find names (of companies, products, genes), not individual words
- interested in specific types of terms
- want to learn from a few examples
  - need contexts to avoid noise
Beyond Bag-of-Words Models

Effective Text Association Mining Needs
- Name recognition
- Term classification
- Ability to learn patterns (lexical sequence or syntactic)

Semantic and syntactic analyzers at varying levels can help

*the duration of* diabetes mellitus *was the significant risk factor for* cataracts
Conclusion

We have reviewed bag-of-words models in the context of three tasks

- Document retrieval
- Opinion mining
- Association mining

Some tasks can be handled effectively (and very simply) by bag-of-words models,

but most benefit from an analysis of language structure