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 = [The cat chased the mouse.]
- D2 = [The dog chased the cat.]
- W = [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 tf_i by its inverse document frequency idf_i

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

$$w_i = tf_i \times idf_i$$

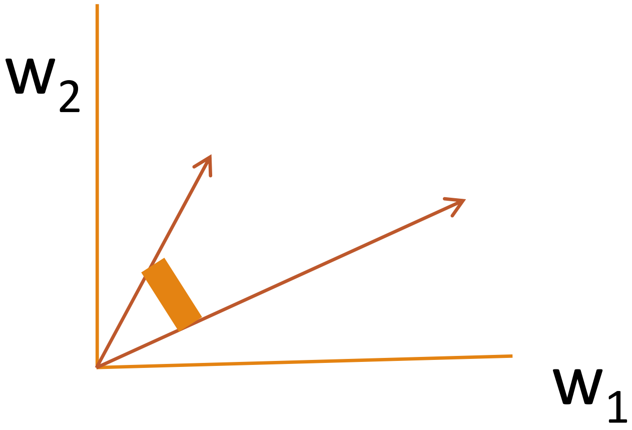
where N = size of collection and n_i = number of documents containing term i

Cosine Similarity

Define a similarity metric between topic vectors

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

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



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

Training Examples	Labels
Simply loved it	Positive
Most disgusting food I have ever had	Negative
Stay away, very disgusting food!	Negative
Menu is absolutely perfect, loved it!	Positive
A really good value for money	Positive
This is a very good restaurant	Positive
Terrible experience!	Negative
This place has best food	Positive
This place has most pathetic serving food!	Negative

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 assign 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

$$s = \underset{t \in \{pos, neg\}}{\text{argmax}}\ P(t \mid W)$$

Use Bayes' rule

$$\text{argmax}_t P(t|W) = \frac{\text{argmax}_t P(W|t)P(t)}{P(W)}$$

$$= \text{argmax}_t P(W|t)P(t)$$

$$= \text{argmax}_t P(w_1,\ldots,w_n|t)P(t)$$

$$= \text{argmax}_t \prod_i P(w_i|t)P(t)$$

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*

$$P(t) = \text{count (docs labeled t)} / N$$

$$P(w_i \mid t) = \frac{\text{count (docs labeled t containing } w_i\text{)}}{\text{count (docs labeled t)}}$$

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:

- $P(t) = \dfrac{\text{total length of docs labeled } t}{\text{total size of corpus}}$

- $P(w_i \mid t) = \dfrac{\text{count (instances of } w_i \text{ in docs labeled } t)}{\text{total length of docs labeled } t}$

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:

- $P(t) = \dfrac{\text{total length of docs labeled } t}{\text{total size of corpus}}$

- $P(w_i \mid t) = \dfrac{\text{count (instances of } w_i \text{ in docs labeled } t)}{\text{total length of docs labeled } t}$

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

$$P(\text{positive} \mid \text{SLP2}) = 0$$

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

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 | 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

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

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

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Priors:

$P(t) = \frac{3}{4}$

$P(j) = \frac{1}{4}$

Conditional Probabilities:

P(Chinese|*c*) = (5+1) / (8+6) = 6/14 = 3/7

P(Tokyo|*c*) = (0+1) / (8+6) = 1/14 ←0

P(Japan|*c*) = (0+1) / (8+6) = 1/14 ←0

P(Chinese|*j*) = (1+1) / (3+6) = 2/9

P(Tokyo|*j*) = (1+1) / (3+6) = 2/9

P(Japan|*j*) = (1+1) / (3+6) = 2/9

Choosing a class:

$P(c|d5) \propto 3/4 * (3/7)^3 * 1/14 * 1/14$

≈ 0.0003

←0

$P(j|d5) \propto \underbrace{1/4}_{P(t)} * \underbrace{(2/9)^3 * 2/9 * 2/9}_{P(w \mid t)}$

≈ 0.0001

Some Useful Resources Using NLTK

Sentiment Analysis with Python NLTK Text Classification

- http://text-processing.com/demo/sentiment/

NLTK Code (simplified classifier)

- http://streamhacker.com/2010/05/10/text-classification-sentiment-analysis-naive-bayes-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:

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

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 <u>contexts</u> 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