Concepts and Applications in NLP

Text Classification and Sentiment

Marion Di Marco

November 26, 2024

Outline

Text Classification

- **Text categorization**: assigning a label or category to a text or document
- Sentiment analysis: extraction of sentiment, the positive or negative orientation of a text
- Spam detection: binary classification task of assigning an email to *spam* or *not spam*
- Language id: in what language a text is written, e.g. social media
- Authorship attribution: determine a text's author
- Topic label: determining the subject of a document (e.g. physics vs. biology)

Text Classification: Examples

- Simple lexical features provide useful cues
- Spam detection:
  phrases like "WITHOUT ANY COST" or "Dear Winner" → probably spam
- Sentiment analysis

+	...zany characters and richly applied satire, and some great plot twists
-	It was pathetic. The worst part about it was the boxing scenes...
+	...awesome caramel sauce and sweet toasty almonds. I love this place!
-	...awful pizza and ridiculously overpriced...

  Some informative words
  - great, richly, awesome
  - pathetic, and awful, ridiculously

Basic Classification Method: Manually Written Rules

- Rules based in combinations of words or other features
  - spam: black-list-address
  - detection of the phrase "dollars" or "have been selected"
- High accuracy possible
  - in specific domains
  - if rules are carefully formulated and refined by experts
- Problems
  - building and maintaining rules is expensive
  - too literal and specific: high-precision, low recall

Supervised Machine Learning for Text Classification

- Supervised learning: data set of input observations, each associated with some correct output (the *supervision signal*)
- Learn how to map from a new observation to a correct output

- Input x and a fixed set of output classes $Y = \{y_1, y_2, ..., y_M\}$ return predicted class $y \in Y$

- Text classification
  - d for document as input variable
  - c for class as output variable
  - training set of N documents: $\{(d_1, c_1), ..., (d_N, c_N)\}$
  - learn a classifier that maps a new document into class $c \in C$

- Probabilistic classifier: also gives the probability of the observation being in class c

Classification Algorithms

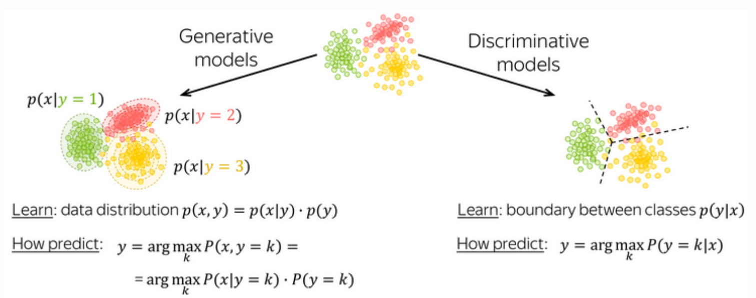
- **Generative classifiers**
  - build a model of how a class could generate some input data
  - given an observation → return the class most likely to have generated the observation

  - Naive Bayes Classifier

- **Discriminative classifiers**
  - learn what features from the input are most useful to discriminate between classes

  - more commonly used
  - for example, logistic regression

Generative and Discriminative Models



- Generative models
  - learn joint probability distribution of data
  - prediction for input x: pick class with the highest joint probability
- Discriminative models
  - look at conditional probability $p(y|x)$: learn border between classes
  - prediction for input x: pick class with the highest conditional probability

Figure from: `https://lena-voita.github.io/nlp_course/text_classification.html`

Outline

Naive Bayes Classifier

- **Multinomial naive Bayes classifier**: probabilistic classifier to predict the category of a text document based on words frequencies

- Naive: simplifying assumption about feature interaction (assumes feature independence, given the target class)
- Multinomial distribution: models the probability of observing a particular set of counts for n trials,

  multinomial distributions work well for text data → word counts

- **Bag of words**: text documents are presented as sets of unordered words, keeping only frequency information

Documents as Bag of Words

Figure 4.1 Intuition of the multinomial naive Bayes classifier applied to a movie review. The position of the words is ignored (the *bag-of-words* assumption) and we make use of the frequency of each word.

Naive Bayes: Intuition

- Naive Bayes is probabilistic classifier: for a document d, it returns the class $\hat{c}$ (of all $c \in C$) with the maximum posterior probability given d

$$\hat{c} = \operatorname*{argmax}_{c \in C} P(c|d)$$

- hat notation ˆ : our estimate of the correct class
- argmax: operation that selects the argument (c) that maximizes the function $P(c|d)$

- Intuition: use Bayes' rule to transform the equation above into other probabilities that have useful properties
- Bayes' rule:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

Naive Bayes: Intuition

- Apply Bayes' rule:

$$\hat{c} = \operatorname*{argmax}_{c \in C} P(c|d) = \operatorname*{argmax}_{c \in C} \frac{P(d|c)P(c)}{P(d)}$$

- Drop denominator (the document is always the same):

$$\hat{c} = \operatorname*{argmax}_{c \in C} P(c|d) = \operatorname*{argmax}_{c \in C} P(d|c)P(c)$$

- Generative model: expresses implicit assumption about how a document is generated
  - a class is sampled from $P(c)$
  - words are generated by sampling from $P(d|c)$
  - → imaging generating documents, i.e. their word counts

Naive Bayes Classification

- Product of the prior probability of class $P(c)$ and likelihood of document $P(d|c)$

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \; \overbrace{P(d|c)}^{\text{likelihood}} \; \overbrace{P(c)}^{\text{prior}}$$

- Represent document d as a set of features $f_1, f_2, ..., f_n$

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \; \overbrace{P(f_1, f_2, ..., f_n|c)}^{\text{likelihood}} \; \overbrace{P(c)}^{\text{prior}}$$

prior probability:
before looking at the data

- Introduce simplifying assumptions

Naive Bayes: Simplifying Assumptions

- **Bag-of-words assumption**
  assume that position of a word in d doesn't matter
- Features $f_1, f_2, ..., f_n$ only encode word identity and not position

- **Naive Bayes Assumption**
  conditional independence assumption that the probabilities $P(f_i|c)$ are independent given the class c
- Probabilities can be "naively" multiplied

$$P(f_1, f_2, ..., f_n|c) = P(f_1|c) \cdot P(f_2|c) \cdot ... \cdot P(f_n|c)$$

- Plug in simplifying assumptions:

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{f \in F} P(f|c)$$

Naive Bayes

- Features: words in document

positions $\leftarrow$ all word positions in test document

$$c_{NB} = \operatorname*{argmax}_{c \in C} P(c) \prod_{i \in positions} P(w_i|c)$$

- Calculate in log-space to avoid problems with very small numbers

$$c_{NB} = \operatorname*{argmax}_{c \in C} \log P(c) + \sum_{i \in positions} \log P(w_i|c)$$

- Sum logs of probabilities instead of multiplying probabilities

$log(ab) = log(a) + log(b)$

Outline

Training Naive Bayes

- Need to learn $P(c)$ and $P(f_i|c)$
- Maximum likelihood estimates based on frequencies in data

- Class prior $P(c)$: percentage of documents in each class c

$$\hat{P}(c) = \frac{N_c}{N_{doc}}$$

- For $P(f_i|c)$: feature as existence of a word $\rightarrow P(w_i|c)$

  fraction of times w_i appears among all words in all documents of class c

  concatenate all documents of class c into one big "class c" text

$$\hat{P}(w_i|c) = \frac{count(w_i, c)}{\sum_{w \in V} count(w, c)}$$

- Vocabulary V: union of words in all classes

Training Naive Bayes: Smoothing

- Problem: estimating the likelihood of a word that we have not seen in a particular class

- Estimate likelihood for *fantastic* given class *positive*;
  suppose there is no occurrence of *fantastic* documents of class *positive*

$$\hat{P}(\text{“fantastic”}|\text{positive}) = \frac{count(\text{“fantastic”}, \text{positive})}{\sum_{w \in V} count(w, \text{positive})} = 0$$

- Multiplication of all feature likelihoods → zero probability for class

- Add-one smoothing (Laplace smoothing)

$$\hat{P}(w_i|c) = \frac{count(w_i, c) + 1}{\sum_{w \in V}(count(w, c) + 1)} = \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c)\right) + |V|}$$

Training Naive Bayes: Unknown Words and Stop Words

- **Unknown words**
  words in the test data not occurring in the training data: ignore and don't include any probability
  - just remove from test input
  - knowing which class has more unknown words: not helpful

- **Stop words**
  very frequent words like *the* and *a*, to be determined via frequency count or stop-word list: can be ignored

  Often does not make much difference in practice

Outline

Naive Bayes: Example

- Sentiment analysis with 2 classes: positive (+) and negative (-)

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

- 5 training sentences
  - vocabulary: 20
- Test sentence: drop *with*

- Class priors: $P(-) = \frac{3}{5}$ and $P(+) = \frac{2}{5}$

Naive Bayes: Example

- Likelihoods for 3 words in the test sentence (with Laplace smoothing):

$$P(\text{"predictable"}|-) = \frac{1+1}{14+20} \quad P(\text{"predictable"}|+) = \frac{0+1}{9+20}$$

$$P(\text{"no"}|-) = \frac{1+1}{14+20} \quad P(\text{"no"}|+) = \frac{0+1}{9+20}$$

$$P(\text{"fun"}|-) = \frac{0+1}{14+20} \quad P(\text{"fun"}|+) = \frac{1+1}{9+20}$$

- Test sentence S = "predictable with no fun"

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in positions} P(w_i|c)$$

$$P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

- Predicted class? *negative* (-)

Outline

Naive Bayes: Variants

- Standard naive Bayes classification can work well for sentiment analysis
- Some changes can improve performance

Binary naive Bayes

- For sentiment classification and some other tasks:
  occurrence of a word matters more than frequency
  - the occurrence of *fantastic* tells us a lot
  - the fact that *fantastic* occurs 4 times does not tell much more
- Clip word counts in documents at 1

- In each document, duplicates are removed in the training and test data

Binary Naive Bayes

Four original documents:

- − it was pathetic the worst part was the boxing scenes
- − no plot twists or great scenes
- + and satire and great plot twists
- + great scenes great film

After per-document binarization:

- − it was pathetic the worst part boxing scenes
- − no plot twists or great scenes
- + and satire great plot twists
- + great scenes film

	NB Counts		Binary Counts	
	+	−	+	−
and	2	0	1	0
boxing	0	1	0	1
film	1	0	1	0
great	3	1	2	1
it	0	1	0	1
no	0	1	0	1
or	0	1	0	1
part	0	1	0	1
pathetic	0	1	0	1
plot	1	1	1	1
satire	1	0	1	0
scenes	1	2	1	2
the	0	2	0	1
twists	1	1	1	1
was	0	2	0	1
worst	0	1	0	1

Figure 4.3 An example of binarization for the binary naive Bayes algorithm.

Naive Bayes: Handling Negation

- *I really like this movie.* (positive)
  I didn't like this movie. (negative)
- Negation completely alters the meaning of the sentence
- Modify a negative word to produce a positive review:
  don't dismiss this film

- Mark negative context add negation marker to every word after a negation (*n't, not, no, never*)
  until next punctuation mark

  `didn't like this movie , but I ...`
  `didn't NOT_like NOT_this NOT_movie , but I ...`

- Words like `NOT_like`, `NOT_recommend` → cues for negative sentiment
- Words like `NOT_bored`, `NOT_dismiss` → cues for positive sentiment

Sentiment Lexicons

- What to do when we have insufficient labeled training data?
- **Sentiment lexicon**: lists of words that are pre-annotated with positive or negative sentiment

+	admirable, beautiful, confident, dazzling, ecstatic, favor, glee, great
-	awful, bad, bias, catastrophe, cheat, deny, envious, foul, harsh, hate

- Add feature that is counted when a word from the lexicon occurs
  - feature "*w* occurs in the positive lexicon": all instances of words in the lexicon as counts for that feature
  - feature "*w* occurs in the negative lexicon": ...

- Lots of training data: using words better than just two features
- Sparse training data or not representative of test data: dense lexicon features might be better than sparse word features

Other Text Classification Tasks

- Naive Bayes can express any property of the input text
- Spam detection
  - one of the first applications of naive Bayes (1998)
  - pre-define likely sets of words and phrases: *one hundred percent guaranteed*, *urgent reply*, *millions of dollars*
  - other features, such as "email subject line is all capital letters"
- Language id: determine the language of a text
  - most effective naive Bayes features are character n-grams
  - trained on multilingual text (e.g. Wikipedia)
  - plus other data resources to capture as many varieties as possible

Outline

Accuracy

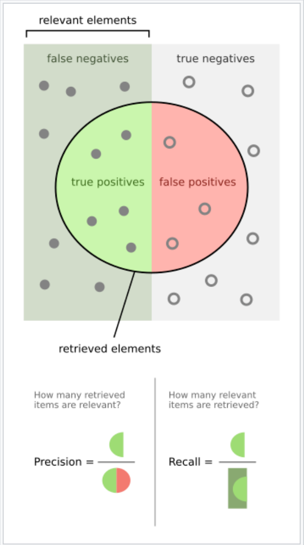
- Accuracy: percentage of all observations the system labeled correctly

- Example: consider 1 million tweets
  - 100 are on the topic of pie
  - 999,900 are about other topics
- Distinguish between tweets *about pie* and *not about pie*
- Simple classifier: labels every tweet as “not about pies”
  - 999,900 true negatives
  - only 100 false negatives
  - accuracy $= 999{,}900/1{,}000{,}000 = 99{,}99\,\%$
- Still a useless classifier: none of the relevant tweets are identified
- Accuracy doesn't work well when classes are unbalanced (most tweets are not about pies)

Precision and Recall

- **Precision**: percentage of retrieved documents relevant to the query
- **Recall**: percentage of relevant documents that were retrieved

- Originally from information retrieval



Precision and recall

Figure from https://en.wikipedia.org/wiki/Precision_and_recall

Evaluation

- Consider binary detection tasks
  - spam detection: is spam – is not spam
  - tweets about particular topic (e.g. pies): yes – no
- **Gold labels**: manually annotated labels in data set

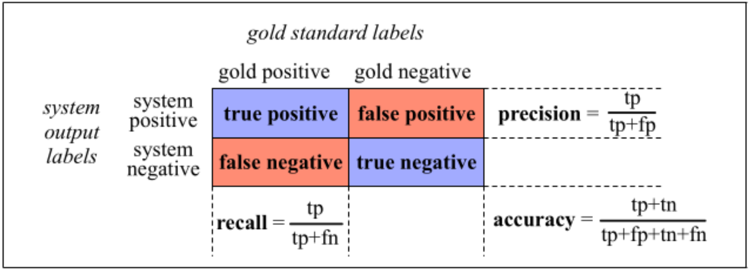


Figure 4.4 A confusion matrix for visualizing how well a binary classification system performs against gold standard labels.

  - true positive: spam documents classified as spam
  - false negative: spam documents classified as non-spam

Precision, Recall and F-measure

- **Precision**: percentage of items labeled as X that are in fact X

$$Precision = \frac{true\ \ positives}{true\ \ positives + false\ \ positives}$$

- **Recall**: percentage of items having label X in the test set that were correctly identified by the system as X

$$Recall = \frac{true\ \ positives}{true\ \ positives + false\ \ negatives}$$

- Precision and recall emphasize true positives
- Useless "nothing is pie" classifier: no true positives

- **F-Measure**: combines precision and recall into one metric

$$F_1 = \frac{2PR}{P+R}$$

The F-measure is the (weighted) harmonic mean of precision and recall

Evaluating More Classes

- Many classification tasks have more than two classes

		gold labels urgent	*gold labels* normal	*gold labels* spam	
system output	urgent	8	10	1	**precision**$_u = \frac{8}{8+10+1}$
system output	normal	5	60	50	**precision**$_n = \frac{60}{5+60+50}$
system output	spam	3	30	200	**precision**$_s = \frac{200}{3+30+200}$
		recall$_u = \frac{8}{8+5+3}$	**recall**$_n = \frac{60}{10+60+30}$	**recall**$_s = \frac{200}{1+50+200}$	

Figure 4.5 Confusion matrix for a three-class categorization task, showing for each pair of classes (c_1, c_2), how many documents from c_1 were (in)correctly assigned to c_2.

Evaluating More Classes

- **Microaveraging**:
  collect the decisions for all classes into a single confusion matrix,
  then compute precision and recall from that table

- **Macroaveraging**:
  compute performance for each class, then average over classes

- Microaverage (average of all items)
  dominated by the more frequent class since the counts are pooled
- Macroaverage (average of all classes)
  better reflects statistics of smaller classes;
  more appropriate when performance on all classes is equally important

Microaveraging and Macroaveraging

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

$$\text{microaverage precision} = \frac{268}{268+99} = \mathbf{.73}$$

$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = \mathbf{.60}$$

Figure 4.6 Separate confusion matrices for the 3 classes from the previous figure, showing the pooled confusion matrix and the microaveraged and macroaveraged precision.

Outline

Data Sets and Cross-Validation

- Three distint data sets
  - training data: train the model
  - development data: tune parameters, decide on model variants
  - test data: test the model on held-out unseen data
- How to best manage splitting of data?
- **Cross-validation**: partition data into k disjoint subsets (folds)
  - train on $k-1$ folds, test on the remaining one
  - repeat sampling process k times
  - average error rate
- $k = 10$: 10-fold cross-validation

- Potential problem: all data needs to be blind → no dev set (that would be peeking at the data)
- Create fixed training and test set, do cross-validation inside the training set

Cross-Validation

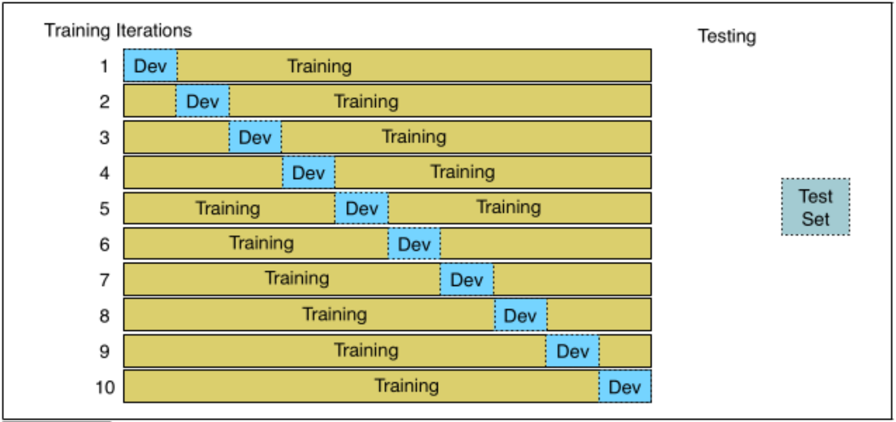


Figure 4.7 10-fold cross-validation

END

Credits

The slides contain content and examples from

- *Speech and Language Processing* (Jurafsky and Martin): Chapter 4
- Slides for Chapter 4: `https://web.stanford.edu/~jurafsky/slp3/slides/nb24aug.pdf`