Text Classification with Naive Bayes and Logistic Models

Text Analytics (MSIA 414)

Yuri Balasanov

iLykei Teaching Tech

For any inquiries contact the author, Yuri Balasanov, at yuri.balasanov@iLykei.com

Outline of the Session

- Naive Bayesian classifier
  - A generative model
  - A graph model
  - Bag-of-words Bernoulli and multinomial models
  - Laplace smoothing
  - Dealing with negation
  - Sentiment lexicons and other tasks
  - Naive Bayes as a language model
- Generative vs. Discriminative classifiers
- Logistic regression classifier
  - Words polarity analysis
- Potential harms of text classifiers

Text:

Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Third Edition draft. D. Jurafsky, J. H. Martin, © 2023

Naive Bayes Classifier

Definition

Naive Bayesian classifier is a simple probabilistic model predicting class $Y = C_k, k = 1, \ldots, K$ using features $X_1, \ldots, X_n$. The assumption making the classifier naive Bayesian is: features are conditionally independent, given the class C_k

$$p(X_1, \ldots, X_n | C_k) = p(X_{1:n} | C_k) = p(X_1 | C_k) \cdots p(X_n | C_k)$$

- To solve classification problem calculate probability $p(C_k | X_1, \ldots, X_n)$ for each class $k, k = 1, \ldots, K$ and predict the class of the highest probability
- If the number of features is large, and/or if their cardinality is large, calculating $p(C_k | X_1, \ldots, X_n)$ directly may become infeasible
- Then using Bayes theorem with the "naive" assumption can help

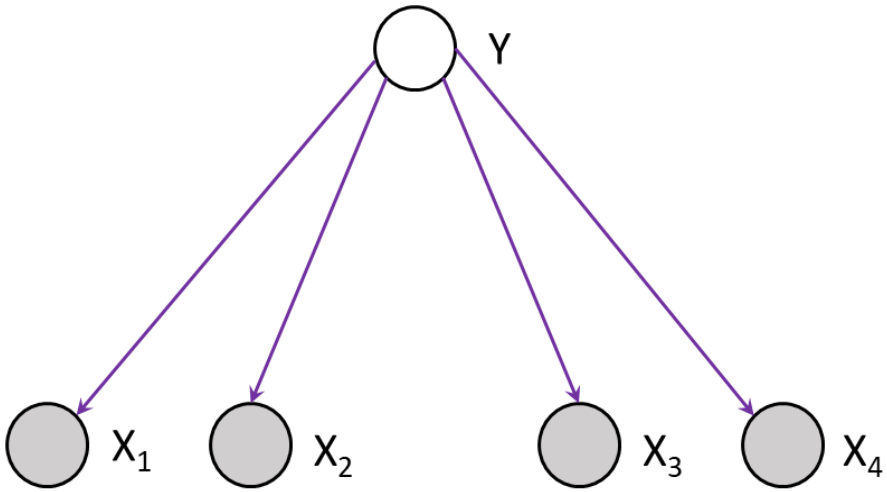
$$p(C_k | X_{1:n}) = \frac{p(C_k)\, p(X_{1:n} | C_k)}{p(X_{1:n})} = \frac{p(C_k)\, p(X_1 | C_k) \cdots p(X_1 | C_k)}{p(X_{1:n})}$$

Naive Bayes Classifier as Generative Model

- If there is a model predicting the direct probability $p\left(C_k \middle| X_{1:n}\right)$, such model can solve the classification problem. But this model is not a **generative model**. It does not explain how the data are generated, i.e. given class C_k, how can the features $X_{1:n}$ be created
- Bayesian classifier is based on probabilities $p\left(X_{1:n} \middle| C_k\right)$, which make the model generative
- Note that in generative classifier the denominator $p\left(X_{1:n}\right)$ does not depend on class C_k. This means that in the process of maximization of $p\left(C_k \middle| X_{1:n}\right)$ the denominator is constant and can be ignored

$$\begin{aligned} p\left(C_k \middle| X_{1:n}\right) &\propto p\left(C_k\right) p\left(X_1 \middle| C_k\right) \cdots p\left(X_1 \middle| C_k\right), \\ \hat{C} &= \arg \max_{k \in \{1,\ldots K\}} p\left(C_k\right) p\left(X_1 \middle| C_k\right) \cdots p\left(X_1 \middle| C_k\right) \end{aligned}$$

Naive Bayes Classifier as Graph Model I



Shaded nodes are observed

Naive Bayes classifier: features X_1, X_2, X_3, X_4 are conditionally independent, given Y, $Y \in \{C_1, \ldots, C_K\}$:

$$p(Y, X_1, X_2, X_3, X_4) = p(Y)\, p(X_1 | Y)\, p(X_2 | Y)\, p(X_3 | Y)\, p(X_4 | Y)$$

Naive Bayes Classifier as Graph Model II

- On the graph model diagram the response random variable Y, $Y \in \{C_1, \ldots, C_K\}$ influences the features through conditional distributions represented by causal edges from Y to X

$$p\left(X_i \middle| C_k\right), i = 1, \ldots, n; k = 1, \ldots, , K$$

- The shaded features X_i are observed
- Information from the observed features X_i to the response variable flows using Bayes Theorem

$$p\left(C_k \middle| X_{1:n}\right) = \frac{p\left(C_k\right) p\left(X_{1:n} \middle| C_k\right)}{p\left(X_{1:n}\right)} = \frac{p\left(C_k\right) p\left(X_1 \middle| C_k\right) \cdots p\left(X_1 \middle| C_k\right)}{p\left(X_{1:n}\right)}$$

- The "naive" assumption simplifies calculation of the n-dimensional distribution $p\left(X_{1:n} \middle| C_k\right) = p\left(X_1 \middle| C_k\right) \cdots p\left(X_1 \middle| C_k\right)$. On the graph it is reflected by no edges going from any feature to other features
- Such graph model is called Bayesian Network

Naive Bayes Classifier and Bag-of-Words Model I

- Naive Bayes Classifiers are used for text classification analysis with **bag-of-words** models
- Bag-of-words is an unordered set of words with their positions ignored, keeping only their frequency in the document
- In naive Bayes bag-of-words models the class C_k is the type of document, for example class "positive movie review"
- Two probabilistic models of documents are common for bag-of-words documents:
  - **Bernoulli Document Model** - a document d is represented by a feature vector with binary elements

$$f_{i,d} = \begin{cases} 1, & \text{if word } w_i \text{ is present in } d, \\ 0, & \text{if } w_i \text{ is not present in } d. \end{cases}$$

  - **Multinomial Document Model** - a document d is represented by a feature vector $f_{i,d}$ with integer elements equal to the frequency of w_i in the document d.

Naive Bayes Classifier and Bag-of-Words Model II

Example

The document d is: "The cat is on the mat." The vocabulary V is: {cat, dog, mat, floor, the, is, on}. The document feature vectors for the 2 models are:

- Bernoulli Document Model: $f_b = \{1, 0, 1, 0, 1, 1, 1\}$.
- Multinomial Document Model: $f_m = \{1, 0, 1, 0, 2, 1, 1\}$.

- Bernoulli model is more suitable for shorter documents, multinomial model can handle longer documents
- For any document d from corpus D and set of classes $C = \{C_1, \ldots, C_K\}$ classification problem is solved by

$$\arg\max_{c \in C} p(c|d) \propto \arg\max_{c \in C} p(d|c)\, p(c),$$

where $p(c)$ is the **prior** probability distribution of classes on C, and $p(d|c)$ is the **likelihood**, or probability of document d given class c

Training the Bernoulli Document Model

- Let N_c be the number of documents in class c, $N_c(w_i)$ be the number of documents of class c containing word w_i, and N_{doc} be the total number of documents in D
- To train binomial model, calculate the prior probability of class c as

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

- Let $p(w_i|c)$ be probability that word $w_i \in V$ is present in a document of class c. Then "naive" assumption means that occurrences of words in the document are conditionally independent given class c, and

$$p(d|c) = \prod_{w_i \in V} \left(f_{i,d}\, p(w_i|c) + (1 - f_{i,d})\,(1 - p(w_i|c))\right),$$

where $f_{i,d}$ is the binary feature of $w_i \in d$

- The probability $\hat{p}(w_i|c)$, necessary for the likelihood is

$$\hat{p}(w_i|c) = \frac{N_c(w_i)}{N_{doc}}$$

Training the Multinomial Document Model

- In multinomial model the feature vector of document d is $f_d = \{f_{i,d}, i = 1, \ldots, |V|\}$, containing frequencies of words w_i from the vocabulary in the document d.
- Let $n_d = \sum_{i=1}^{|V|} f_{i,d}$ be the total number of words in document d. To find $p(d|c)$ use again the "naive" assumption that words in the document occur independently, and use the multinomial model

$$p(d|c) = p(f_d|c) = \frac{n_d!}{\prod_{i=1}^{|V|} f_{i,d}!} \prod_{i=1}^{|V|} p(w_i|c)^{f_{i,d}}$$

- Let

$$\delta_{d,c} = \begin{cases} 1, & \text{if document } d \text{ belongs to class } c, \\ 0, & \text{otherwise.} \end{cases}$$

  Then

$$\hat{p}(w_i|c) = \frac{\sum_{d \in D} f_{i,d} \delta_{d,c}}{\sum_{w \in V} \sum_{d \in D} f_{w,d} \delta_{d,c}}$$

- Calculate the prior probability $\hat{p}(c)$ of class c as before

Laplace Smoothing

- There is a problem with the multinomial model when a word h does not appear in the class c. In such case $f_{h,d} = 0$ for each $d \in c$, and

$$\hat{p}\left(h \middle| c\right) = \frac{\sum_{d \in D} f_{h,d} \delta_{d,c}}{\sum_{w \in V} \sum_{d \in D} f_{w,d} \delta_{d,c}} = 0$$

  This causes

$$p\left(d \middle| c\right) = \frac{n_d!}{\prod_{i=1}^{|V|} f_{i,d}!} \prod_{i=1}^{|V|} p\left(w_i \middle| c\right)^{f_{i,d}} = 0$$

- If one word does not appear in the class, posterior distribution of that class is zero, regardless of all other evidence
- To solve this problem apply add-one smoothing (a.k.a. Laplace smoothing, Laplace law of succession) by adding 1 to the numerator of $\hat{p}\left(w_i \middle| c\right)$:

$$\hat{p}\left(w_i \middle| c\right) = \frac{1 + \sum_{d \in D} f_{i,d} \delta_{d,c}}{\sum_{w \in V} \sum_{d \in D} f_{w,d} \delta_{d,c}}$$

Steps of Training a Naive Bayes Classifier

1. Create a vocabulary V. Selection of words in V is important because the size of V is the dimension of the feature vector. It should include all important words while being as short as possible
2. Collect the necessary metrics from the training corpus:
   - N_{doc} - the total number of documents in D
   - N_c - the number of documents in each class $c \in C$
   - Collect statistics necessary for calculation of likelihood:
     - For Bernoulli model: $N_c(w_i)$ - the number of documents of class c containing word w_i, for each $w_i \in V$
     - For multinomial model: $n_d = \sum_{i=1}^{|V|} f_{i,d}$ - the total number of words in each document $d \in D$, where $f_{i,d}$ is the element of feature vector $f_d = \{f_{i,d}, i = 1, \ldots, |V|\}$, containing frequencies of words w_i from the vocabulary V in the document d
3. To classify an unlabeled document $\tilde{d}$, estimate the prior probabilities $\hat{p}(c)$, $c \in C$, and likelihoods of words $\hat{p}(w_i|c)$ necessary for likelihood of the document $p(\tilde{d}|c)$. Then predict the class by

$$\arg\max_{c \in C} p(c|\tilde{d}) \propto \arg\max_{c \in C} p(\tilde{d}|c)\, p(c)$$

Dealing with Negation

- Negation alters the inferences drawn from predicates

Example

Consider the difference between "I really like this movie" (positive) and "I didn't like this movie" (negative). The negation expressed by "didn't" completely alters the inferences drawn from the predicate "like".

- A simple method used in sentiment analysis is prepending the prefix "NOT_" to every word after a token of logical negation ("n't", "not", "no", "never") during word normalization

Example

The phrase "didn't like this movie , but I" becomes "didn't NOT_like NOT_this NOT_movie , but I".
New tokens in the vocabulary like "NOT_like" will occur more often in negative document and act as cue for negative sentiment.

Sentiment Lexicons

- Sometimes there is not enough labeled training data to reliably train naive Bayes sentiment classifier. In such cases improvement can be achieved by using **sentiment lexicons,** lists of words that are pre-annotated with positive or negative sentiment
  - General Inquirer
  - LIWC
  - Opinion Lexicon
  - MPQA Subjectivity Lexicon
- A common way to use lexicons in a naive Bayes classifier is to add a feature that is counted whenever a word from that lexicon occurs.

Example

The MPQA subjectivity lexicon has 6885 words each marked for whether it is strongly or weakly biased positive or negative. Some examples:

+: admirable, beautiful, confident, dazzling, ecstatic, favor, glee, great

-: awful, bad, bias, catastrophe, cheat, deny, envious, foul, harsh, hate

Other Tasks

Spam detection

- A common solution for spam detection, rather than using all the words as individual features, is to predefine likely sets of words or phrases as features, combined with features that are not purely linguistic
- For example, SpamAssassin predefines features like:
  - The phrase “one hundred percent guaranteed”
  - The feature "mentions millions of dollars", which is a regular expression that matches suspiciously large sums of money
  - The feature "HTML has a low ratio of text to image area", that aren't purely linguistic and might require some sophisticated computation
  - Totally non-linguistic features like the path that the email took to arrive
  - Email subject line is all capital letters
  - Contains phrases of urgency like “urgent reply”
  - Email subject line contains “online pharmaceutical”
  - HTML has unbalanced “head” tags
  - Claims you can be removed from the list

Other Tasks

Language identification

- Language Identification is determining what language a given piece of text is written in
- The most effective naive Bayes features for language id are not words at all, but **character n-grams**, like
  - 2-grams ("zw")
  - 3-grams ("nya", " Vo")
  - 4-grams ("ie z", "thei")
  - Byte n-grams, where instead of using the multibyte Unicode character representations called codepoints, we just pretend everything is a string of raw bytes. Because spaces count as a byte, byte n-grams can model statistics about the beginning or ending of words
- A widely used naive Bayes system, langid.py begins with all possible n-grams of lengths 1-4, using feature selection to select the most informative 7000 final features

Naive Bayes as a Language Model

- Naive Bayes classifiers can be viewed as language models assigning probabilities to sequences of words, like sentences s:

$$p(s|c) = \prod_{w_i \in s} p(w_i|c)$$

Example

Consider a model for 2 classes ($\pm$) and distributions on the dictionary

w	I	love	this	fun	film
$p(w\|+)$	0.1	0.1	0.01	0.05	0.1
$p(w\|-)$	0.2	0.001	0.01	0.005	0.1

Then for $s =$ "I love this film"

$$\begin{aligned} p(s|+) &= 0.1 \times 0.1 \times 0.01 \times 0.05 \times 0.1 = 5.0 \times 10^{-7}, \\ p(s|-) &= 0.2 \times 0.001 \times 0.01 \times 0.005 \times 0.1 = 1.0 \times 10^{-9} \end{aligned}$$

Generative vs. Discriminative Classifiers for NLP I

- Logistic regression, and softmax regression, its extension to multi-class classification, are among the most important analytic tools in the social and natural sciences
- In natural language processing, logistic regression is the baseline supervised machine learning algorithm for classification, and also has a very close relationship with neural networks
- The most important difference between naive Bayes classifier and logistic regression classifier is that the latter is a discriminative classifier while the former is a generative classifier
- While a generative model has the goal of "understanding" the classes (one could literally ask the model to generate observations from each class), a discriminative model only tries to distinguish them based on some features

Generative vs. Discriminative Classifiers for NLP II

Example

Imagine a classifier learning to classify images of cats and dogs with one of the features available beeing "wearing a collar". While the collar feature is not "generative" (generated dogs images may or may not wear a collar), it may well separate the classes in the sample if used by a discriminative model

- While a generative model classification is based on the Bayes rule

$$p(c|d) = \frac{p(d|c)\,p(c)}{p(d)},$$

  a discriminative model calculates $p(c|d)$ directly from the sample

Logistic Regression I

Assume categorical response being a discrete variable with values 0 and 1 and one quantitative predictor. The left-hand side of the model then is the probability of success (1):

$$y = \beta_0 + \beta_1 \mathbf{x} + \varepsilon$$

Such type of linear model is a case of **logistic regression**.
In order to make the ranges of both sides of the model comparable we will use the **logit transformation**:

$$\text{logit}\left(\pi\left(x\right)\right) = \log\left(\frac{\pi\left(x\right)}{1-\pi\left(x\right)}\right), \pi\left(x\right) = \mathbb{P}\left\{y = 1 | \, x\right\}$$

Then the model of logistic regression is

$$\begin{aligned} \log\left(\frac{\pi\left(\mathbf{x}\right)}{1-\pi\left(\mathbf{x}\right)}\right) &= \beta_0 + \beta_1 \mathbf{x}, \\ \pi\left(\mathbf{x}\right) &= \frac{e^{\beta_0+\beta_1 \mathbf{x}}}{1+e^{\beta_0+\beta_1 \mathbf{x}}} = \frac{1}{e^{-(\beta_0+\beta_1 \mathbf{x})}+1} \end{aligned}$$

Logistic Regression II

Joseph Berkson

- Logistic regression fits **logistic function** to approximate probability p of success in binomial experiment
- Logistic function $f(x) = \frac{e^x}{1+e^x}$ has been used since 1830-1840 thanks to works of Pierre François Verhulst and Adolphe Quetelet
- They applied the function to describe process of population growth. The sigmoid shape explains how the population initially grows exponentially (geometric), then, as saturation begins, it slows down to linear and eventually stops
- As a regression model with binary response logistic regression was introduced by Joseph Berkson since 1944
- Joseph Berkson, 5/14/1899-9/12/1982, born in New York City, trained as a physicist, physician and statistician
- Head of Division of Biometry and Medical Statistics of the Mayo Clinic, Rochester, MN. Remembered for opposition to the idea that cigarette smoking causes cancer

Logistic Regression III

Maximum Likelihood

Logistic regression uses method of maximum likelihood for fitting.
The assumption for two-level model is: Y_i are i.i.d. random variables and

$$Y \sim Binom\left(1, \frac{e^{\beta_0+\beta_1 \mathbf{x}}}{1+e^{\beta_0+\beta_1 \mathbf{x}}}\right).$$

Solving the model for β_1 we define $e^{\beta_1} =$ **odds ratio**;

$$\begin{aligned}\beta_1 &= \log\left(\frac{\pi(x+1)}{1-\pi(x+1)}\right) - \log\left(\frac{\pi(x)}{1-\pi(x)}\right) \\ &= \log\left(\frac{\frac{\pi(x+1)}{1-\pi(x+1)}}{\frac{\pi(x)}{1-\pi(x)}}\right) = \textbf{log odds ratio}\end{aligned}$$

.

Logistic Regression IV

Estimation of Parameters

For binomial random variable the likelihood function is

$$L\left(\vartheta\right)=\prod_{i=1}^{n}p\left(x_i;\vartheta\right)^{y_i}\left(1-p\left(x_i;\vartheta\right)\right)^{1-y_y}$$

After maximizing the log-likelihood function we will predict the response

$$Y=\begin{cases}1, \text{ if } p\left(x\right)\geq 0.5,\\ 0, \text{if } p\left(x\right)<0.5,\end{cases}$$

This means that we predict $Y=1$ when

$$\begin{aligned}p\left(x\right) &= \frac{1}{e^{-\left(\beta_0+\beta_1\mathbf{x}\right)}+1}\geq 0.5\\ \beta_0+\beta_1 x &\geq 0\end{aligned}$$

This is a linear classifier which not only tells where is he boundary, but also how far we are from the boundary.

Polarity Analysis I

- Besides the straightforward application of logistic regression to predicting probabilities of classes (i.e. $p(+|d)$, $p(-|d)$), consider a simple method, usually useful for bag-of-words models, called **words polarity analysis**
- The steps of word polarity analysis are:
  - Create tf-idf feature for each word of the vocabulary based on the train sample
  - Fit logistic regression with response equal to the class of the document and the feature column for each word in the dictionary
  - Logistic regression coefficients for all words are interpreted as polarity measure for the given class
  - Plot the the words corresponding to the largest positive coefficients (polarity for class 1), and the words corresponding to the most negative coefficients (class 0) to analyze the classes

Polarity Analysis II

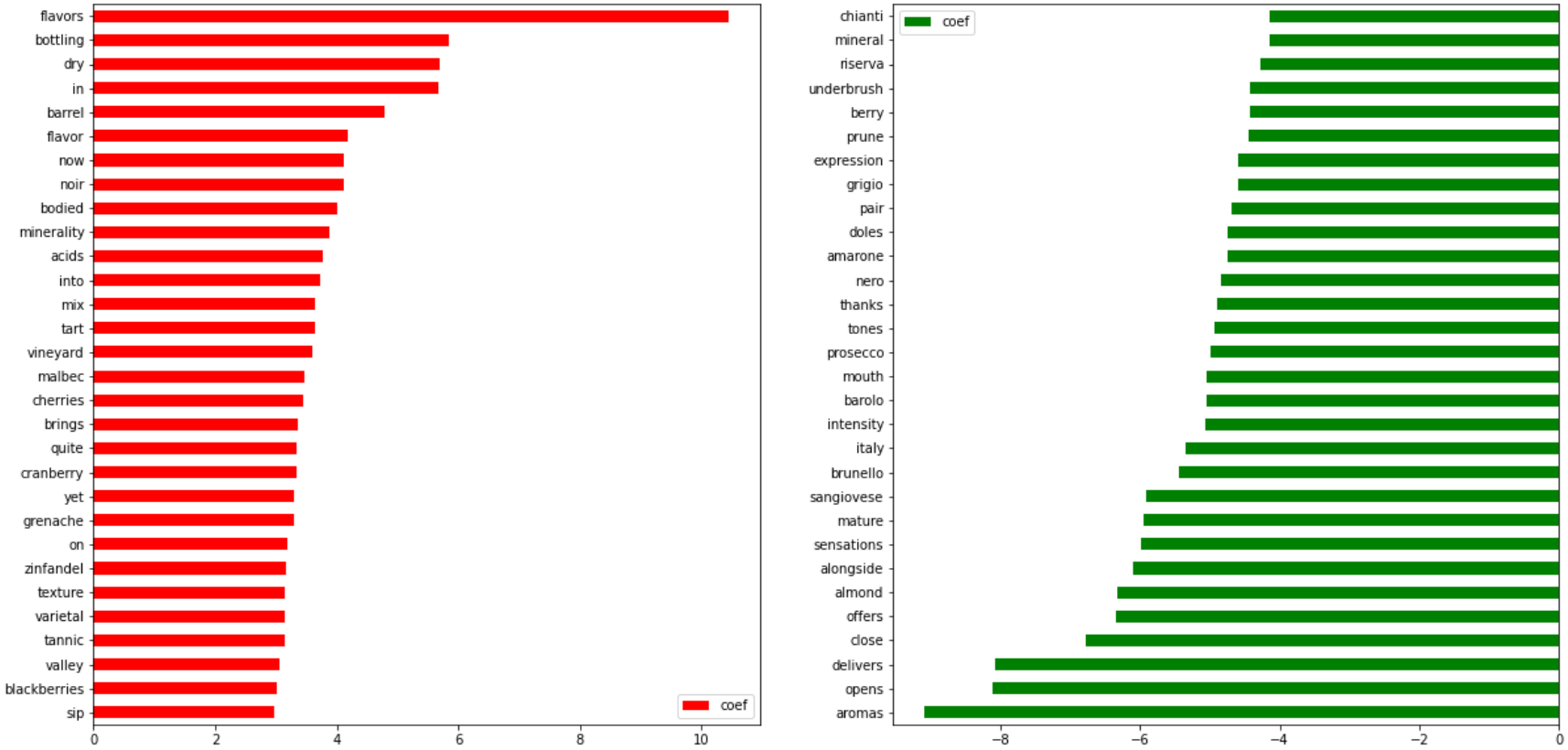


Figure: Polarity analysis of wine reviews

Potential Harms of Text Classifiers I

Representational harms

- It is important to avoid potential harms that may result from text classifiers, harms that exist for naive Bayes classifiers, Logistic regression, as well as for the other classification algorithms
- One class of harms is **representational harms** caused by a system that demeans a social group, for example by perpetuating negative stereotypes about them

Example

One study (see Text, pp.75-76) examined the performance of 200 sentiment analysis systems on pairs of sentences that were identical except for containing either a common African American first name (like Shaniqua) or a common European American first name (like Stephanie). Most systems assigned lower sentiment and more negative emotion to sentences with African American names

Potential Harms of Text Classifiers II

Censorship

- Another class is **censorship**. For example, toxicity detection is the task of detecting hate speech, abuse, harassment, or other kinds of toxic language. While the goal of such classifiers is to help reduce societal harm, toxicity classifiers can themselves cause harms

Example

For example, researchers have shown that some widely used toxicity classifiers incorrectly flag as being toxic sentences that are non-toxic but simply mention minority identities like women, blind people, gay people, or simply use linguistic features characteristic of varieties like African-American Vernacular English

- Such false positive errors, if employed by toxicity detection systems without human oversight, could lead to the censoring of discourse by or about these groups

Potential Harms of Text Classifiers III

- These model problems can be caused by biases or other problems in the training data; in general, machine learning systems replicate and even amplify the biases in their training data
- But these problems can also be caused:
  - By the labels (for example, due to biases in the human labelers)
  - By the resources used (like lexicons, or model components like pretrained embeddings)
  - By model architecture (like what the model is trained to optimize)
- While the mitigation of these biases (for example by carefully considering the training data sources) is an important area of research, we currently don't have general solutions
- For this reason it is important, when introducing any NLP model, to study these kinds of factors and make them clear

Potential Harms of Text Classifiers IV

Model card

- One way to do this is by releasing a **model card** for each version of a model
- A model card documents a machine learning model with information like:
  - Training algorithms and parameters
  - Training data sources, motivation, and preprocessing
  - Evaluation of data sources, motivation, and preprocessing
  - Intended use and users
  - Model performance across different demographic or other groups and environmental situations