Text Classification

Classification Methods: Supervised Machine Learning

- Any kind of classifier
  - Naïve Bayes
  - Random forests
  - Logistic regression
  - Perceptrons
- **Generative classifiers** Generative classifiers model the joint distribution of features and labels, estimating both $P(X|Y)$ and $P(Y)$, and then use Bayes' theorem to compute posterior probabilities.
- like Naïve Bayes build a model for each class.
- **Discriminative classifiers** Rather than modeling the joint distribution of features and labels, discriminative classifiers focus on learning the decision boundary between classes based on the observed features.
- They aim to find the function that separates the feature space into regions corresponding to different classes. During training, the classifier learns to distinguish between spam and non-spam emails by finding the optimal decision boundary in the feature space.
- Directly predict outputs Y from inputs X

Sentiment Analysis: Optimization

- Whether a word occurs or not seems to matter more than its frequency
- Improves performance by clipping word counts in **each document** at 1
  - called **binary multinomial naive Bayes** or **binary NB**
  - for each document remove all duplicate words before concatenating them into the single big document
  - the word *great* has a count of 2 even for Binary NB, because it appears in multiple documents.

Sentiment Analysis: Optimization

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

Sentiment Analysis: Optimization

- when a negation is present, the sentiment of the subsequent words may be reversed or altered.
- **Negation**
  - *I really like this movie* (positive)
  - *I didn’t like this movie* (negative)
- Prepend the prefix *NOT* to every word after a token of logical negation (*n’t, not, no, never*) until the 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* will occur more often in negative documents, while words like *NOT_bored*, *NOT_dismiss* will acquire positive associations

Sentiment Analysis: Insufficient data

- Insufficient labeled training data to train accurate naive Bayes classifiers
- **Sentiment lexicons**
  - Extract positive and negative word features from **sentiment lexicons**
    - lists of words that are pre-annotated with positive or negative sentiment

- MPQA subjectivity lexicon
  - 6885 words, 2718 positive and 4912 negative
  - + : *admirable, beautiful, confident, dazzling, ecstatic, favor, glee, great*
    − : *awful, bad, bias, catastrophe, cheat, deny, envious, foul, harsh, hate*

- If we do not have a lot of training data, add features:
  - **‘this word occurs in the positive lexicon’**
  - **‘this word occurs in the negative lexicon’**
  - And treat all instances of words in the lexicon as counts for that one feature, instead of counting each word separately
  - If we have lots of training data, and if the test data matches the training data, using just two features won’t work as well as using all the words

SPAM vs HAM: Optimization

- Rather than using all the words as individual features:
  - predefine likely sets of words or phrases as features
  - including features that are not purely linguistic
- E.g. the open-source SpamAssassin has features like:
  - the phrase “one hundred percent guaranteed”
  - the feature *mentions “millions of dollars”* (as a regex)
  - Not purely linguistic features: *HTML has a low ratio oftext to image area*
  - Non-linguistic features: “the path that the email took to arrive”
  - Other features:
    - 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

SPAM vs HAM: Optimization

Rather than using all the words as individual features:

1.Predefined Sets of Likely Features:

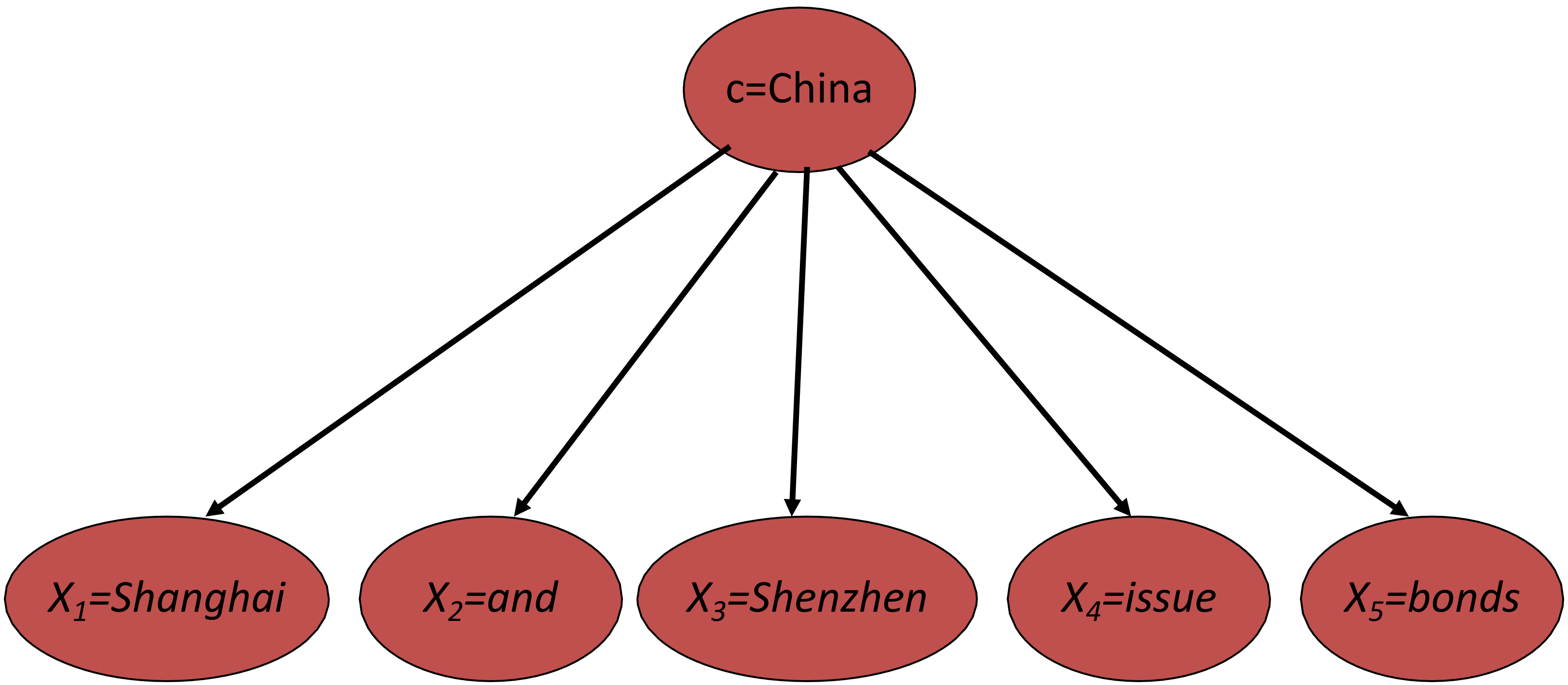
- Instead of using all words as individual features, predefined sets of words or phrases are identified as features.
- These sets are likely to appear frequently in either spam or ham emails. This approach reduces the dimensionality of the feature space and focuses on relevant linguistic patterns.

2.Including Non-Purely Linguistic Features:

- Features are not limited to linguistic elements alone.
- They may include non-linguistic aspects of emails, such as their structure, formatting, or metadata.
- For instance, features like HTML text-to-image ratio or the email's routing path can be considered.
- Structural irregularities in HTML, such as unbalanced "head" tags.

Naïve Bayes: Relationship to Language Modeling

Generative Model for Multinomial Naïve Bayes



Naïve Bayes and Language Modeling

- Naïve bayes classifiers can use any sort of feature
  - URL, email address, dictionaries, network features
- But if, as in the previous slides
  - We use **only** word features
  - we use **all** of the words in the text (not a subset)
- Then
  - Naïve bayes has an important similarity to language modeling.

Each class = a unigram language model

- Assigning each word: P(word | c)
- Assigning each sentence: $P(s|c) = \prod_{i \in positions} P(w_i|c)$

Class	*pos*
0.1	I
0.1	love
0.01	this
0.05	fun
0.1	film
…	

I	love	this	fun	film
0.1	0.1	.05	0.01	0.1

P(s | pos) = 0.0000005

Naïve Bayes as a Language Model

- Which class assigns the higher probability to s?

Model pos

0.1	I
0.1	love
0.01	this
0.05	fun
0.1	film

Model neg

0.2	I
0.001	love
0.01	this
0.005	fun
0.1	film

I	love	this	fun	film
0.1	0.1	0.01	0.05	0.1
0.2	0.001	0.01	0.005	0.1

$P(s|pos) > P(s|neg)$

Multinomial Naïve Bayes: Another Worked Example

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

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

	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:

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

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

Conditional Probabilities:

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

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

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

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

P(j|d5)

$\propto$ 1/4 * $(2/9)^3$ * 2/9 * 2/9

≈ 0.0001

Summary: Naive Bayes is Not So Naive

- Very good in domains with many equally important features: It excels in scenarios where multiple features are equally relevant, as it doesn't prioritize one feature over another.
- Optimal if the independence assumptions hold:
  - If assumed independence is correct, then it is the Bayes Optimal Classifier for problem
- A good dependable baseline for text classification
  - **But we will see other classifiers that give better accuracy**

Gold standards

- In spam detection (for example) for each item (email document)
  - we therefore need to know whether our system called it spam or not
  - We also need to know whether the email is actually spam or not, i.e. the human-defined labels for each document
  - We will refer to these human labels as the **gold labels**.

Gold Labels, Annotators and Agreement

Instance	Annotator 1	Annotator 2
1	Positive	Positive
2	Positive	Negative
3	Negative	Positive
4	Negative	Negative
5	Positive	Negative
6	Negative	Positive

1. Raw Agreement

- **Formula:** (Number of instances both annotators agreed on) / (Total number of instances)

  - Raw agreement = 2 / 6 = 0.33 (33.33%)

Chance Agreement

- Chance agreement is the level of agreement that could be expected by chance alone.

```
Chance agreement = Σ(pi * pj)
```

Where:

- `pi` is the proportion of instances in class i annotated by Annotator 1.
- `pj` is the proportion of instances in class j annotated by Annotator 2.
- Σ is the summation symbol, meaning we add up the products for all classes.

- 100 instances
- Annotator 1: 30 positive, 70 negative
- Annotator 2: 50 positive, 50 negative

Step 1: Calculate proportions:

- For Annotator 1:
  - `p1` (positive) = 30 / 100 = 0.3
  - `p2` (negative) = 70 / 100 = 0.7
- For Annotator 2:
  - `p1` (positive) = 50 / 100 = 0.5
  - `p2` (negative) = 50 / 100 = 0.5

Step 2: Apply the formula:

- Chance agreement = (0.3 * 0.5) + (0.7 * 0.5)
- Chance agreement = 0.15 + 0.35 = 0.5

The chance agreement of 0.5 means that there's a 50% chance that the two annotators would agree on an instance randomly, even if they were guessing. This serves as a baseline to compare with the observed agreement (raw agreement).

• Chance Agreement

Example	Annotator 1	Annotator 2
1	Joy	Joy
2	Sadness	Joy
3	Anger	Anger
4	Fear	Fear
5	Joy	Sadness

- **Joy:**
  - Annotator 1: 2/5 = 0.4
  - Annotator 2: 2/5 = 0.4
- **Sadness:**
  - Annotator 1: 1/5 = 0.2
  - Annotator 2: 1/5 = 0.2
- **Anger:**
  - Annotator 1: 1/5 = 0.2
  - Annotator 2: 1/5 = 0.2
- **Fear:**
  - Annotator 1: 1/5 = 0.2
  - Annotator 2: 1/5 = 0.2

- Chance agreement = (0.4 * 0.4) + (0.2 * 0.2) + (0.2 * 0.2) + (0.2 * 0.2) = 0.32

The chance agreement is 0.32. This means that if the annotators were guessing randomly, we would expect them to agree on about 32% of the instances.

Cohen’s Kappa

Cohen’s Kappa Statistic is used to measure the level of agreement between two raters or judges who each classify items into mutually exclusive categories.

$$k = (p_o - p_e) / (1 - p_e)$$

where:

- p_o: Relative observed agreement among raters
- p_e: Hypothetical probability of chance agreement

https://www.statology.org/cohens-kappa-statistic/

Cohen’s Kappa

Cohen’s Kappa always ranges between 0 and 1, with 0 indicating no agreement between the two raters and 1 indicating perfect agreement between the two raters.

Suppose two museum curators are asked to rate 70 paintings on whether they’re good enough to be hung in a new exhibit.
The following 2×2 table shows the results of the ratings:

		Rater 2	
		Yes	No
Rater 1	Yes	25	10
	No	15	20

Cohen’s Kappa

Step 1: Calculate relative agreement (p_o) between raters

the proportion of total ratings that the raters both said “Yes” or both said “No” on. We can calculate this as:

- p_o = (Both said Yes + Both said No) / (Total Ratings)
- p_o = (25 + 20) / (70) = **0.6429**

Step 2: Calculate the hypothetical probability of chance agreement (p_e) between raters

Next, calculate the probability that the raters could have agreed purely by chance.

This is calculated as the total number of times that Rater 1 said “Yes” divided by the total number of responses, multiplied by the total number of times that Rater 2 said “Yes” divided by the total number of responses, added to the total number of times that Rater 1 said “No” multiplied by the total number of times that Rater 2 said “No.” For our example, this is calculated as:

- P(“Yes”) = ((25+10)/70) * ((25+15)/70) = 0.285714
- P(“No”) = ((15+20)/70) * ((10+20)/70) = 0.214285

= 0.285714 + 0.214285 = **0.5**

Cohen’s Kappa

Step 3: Calculate Cohen’s Kappa

Lastly, we’ll use p_o and p_e to calculate Cohen’s Kappa:

- $k = (p_o – p_e) / (1 – p_e)$
- $k = (0.6429 – 0.5) / (1 – 0.5)$
- $k = 0.2857$

Cohen’s Kappa turns out to be **0.2857**.

Cohen's Kappa	Interpretation
0	No agreement
0.10 - 0.20	Slight agreement
0.21 - 0.40	Fair agreement
0.41 - 0.60	Moderate agreement
0.61 - 0.80	Substantial agreement
0.81 - 0.99	Near perfect agreement
1	Perfect agreement

What about 3 classes?