CIS 530: Text Processing

MONDAYS AND WEDNESDAYS 1:30-3PM
3401 WALNUT, ROOM 401B
COMPUTATIONAL-LINGUISTICS-CLASS.ORG

PROFESSOR CALLISON-BURCH

Reminders

HW1 IS DUE TONIGHT BEFORE 11:59PM.

IF YOU DON'T YET HAVE A PERMIT AND YOU ARE HOPING TO GET INTO THE CLASS, YOU **MUST** TURN THE HOMEWORK IN ON TIME.

READ TEXTBOOK CHAPTER 2 AND [DEPRESSION AND SELF-HARM RISK ASSESSMENT IN ONLINE FORUMS]

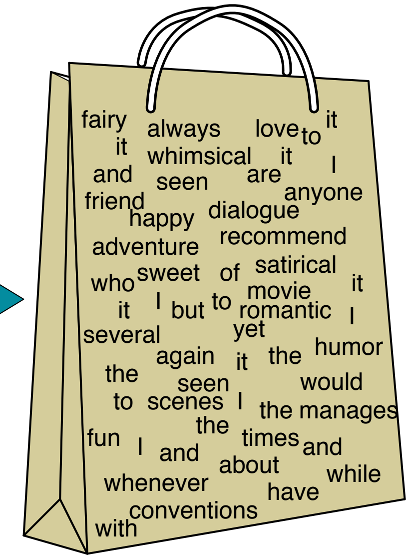
Text Classification with Naïve Bayes

THE TASK OF TEXT CLASSIFICATION

The Bag of Words Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…



Multinomial Naïve Bayes Independence Assumptions

$$P(x_1, x_2, \ldots, x_n \mid c)$$

Bag of Words assumption: Assume position doesn't matter

Conditional Independence: Assume the feature probabilities $P(x_i|c_j)$ are independent given the class c.

$$P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

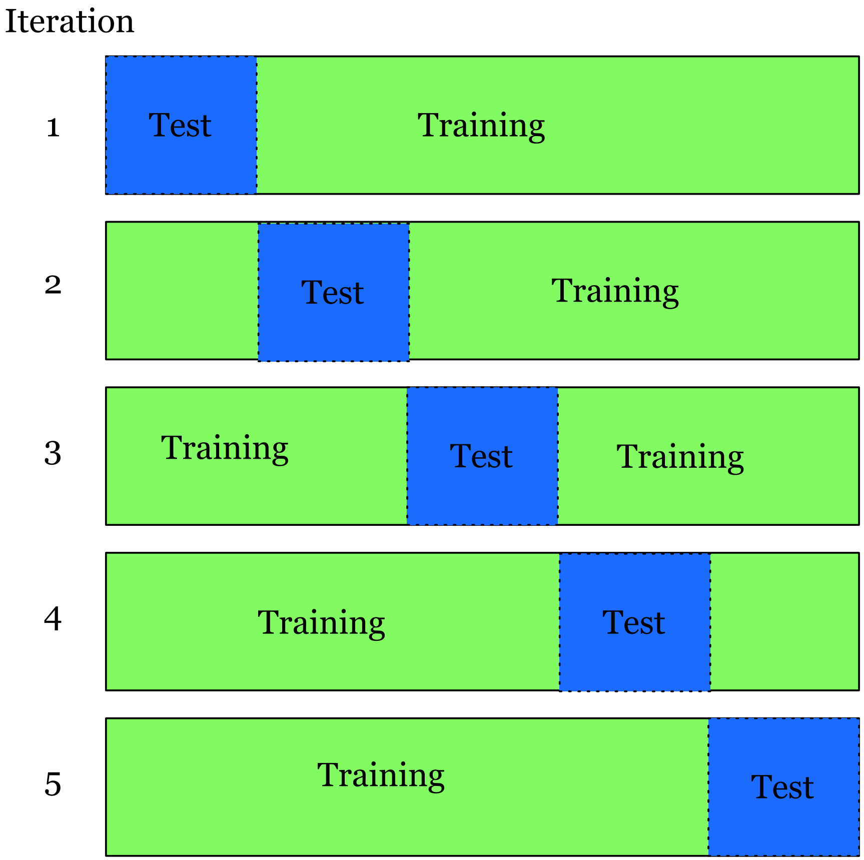
Multinomial Naïve Bayes Classifier

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c_j) \prod_{x \in X} P(x \mid c)$$

Text Classification and Naïve Bayes

TEXT CLASSIFICATION: EVALUATION



Cross-Validation

Break up data into 10 folds

- (Equal positive and negative inside each fold?)

For each fold

- Choose the fold as a temporary test set
- Train on 9 folds, compute performance on the test fold

Report average performance of the 10 runs

Development Test Sets and Cross-validation

Training set | Development Test Set | Test Set

Metric: P/R/F1 or Accuracy

Development test set
- avoid overfitting to the unseen test set
- Use dev set to select the “best” model

- Cross-validation over multiple splits
  - Handle sampling errors from different datasets
- Compute pooled dev set performance
- This way we can use all data for validation

Training Set | Dev Test

Training Set | Dev Test

Dev Test | Training Set

Test Set

Precision and Recall

gold standard labels

		gold positive	gold negative	
system output labels	system positive	**true positive**	**false positive**	$\textbf{precision} = \frac{tp}{tp+fp}$
	system negative	**false negative**	**true negative**	
		$\textbf{recall} = \frac{tp}{tp+fn}$		$\textbf{accuracy} = \frac{tp+tn}{tp+fp+tn+fn}$

Precision and Recall

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

Precision and Recall

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

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

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

Basic Text Processing

REGULAR EXPRESSIONS

Regular expressions

A formal language for specifying text strings

How can we search for any of these?

- woodchuck
- woodchucks
- Woodchuck
- Woodchucks

Regular Expressions: Disjunctions

Letters inside square brackets []

Pattern	Matches
`[wW]oodchuck`	Woodchuck, woodchuck
`[1234567890]`	Any digit

Ranges `[A-Z]`

Pattern	Matches	
`[A-Z]`	An upper case letter	`Drenched Blossoms`
`[a-z]`	A lower case letter	`my beans were impatient`
`[0-9]`	A single digit	`Chapter 1: Down the Rabbit Hole`

Regular Expressions: Negation in Disjunction

Negations `[^Ss]`

- Carat means negation only when first in []

Pattern	Matches	
`[^A-Z]`	Not an upper case letter	`Oyfn pripetchik`
`[^Ss]`	Neither 'S' nor 's'	`I have no exquisite reason"`
`[^e^]`	Neither e nor ^	`Look here`
`2^3`	The pattern 2 carat 3	`The value of 2^3 is 8.`

Regular Expressions: More Disjunction

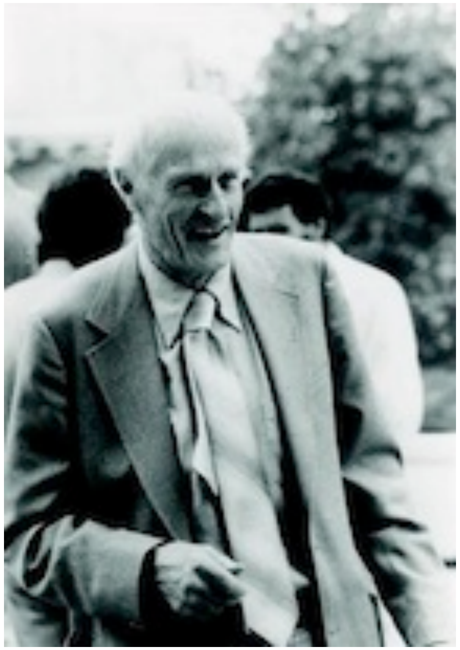
Woodchucks is another name for groundhog!

The pipe | for disjunction

Pattern	Matches
`groundhog|woodchuck`	woodchuck
`yours|mine`	`yours` `mine`
`a|b|c`	= [abc]
`[gG]roundhog|[Ww]oodchuck`	Woodchuck

Regular Expressions: ? * + .

Pattern	Matches	
colou?r	Optional previous char	color colour
oo*h!	0 or more of previous char	oh! ooh! oooh! ooooh!
o+h!	1 or more of previous char	oh! ooh! oooh! ooooh!
baa+		baa baaa baaaa baaaaa
beg.n		begin begun begun beg3n



Regular Expressions: Anchors ^ $

Pattern	Matches
`^[A-Z]`	`Palo Alto`
`^[^A-Za-z]`	`1` `"Hello"`
`\.$`	`The end.`
`.$`	`The end?` `The end!`

Find me all instances of the word “the” in a text.

`the`

Misses capitalized examples

`[tT]he`

Incorrectly returns o**the**r or **the**ology

`[^a-zA-Z][tT]he[^a-zA-Z]`

Is correct

Example

Errors

The process we just went through was based on fixing two kinds of errors

- Matching strings that we should not have matched (there, then, other)
  - False positives
- Not matching things that we should have matched (The)
  - False negatives

In NLP we are always dealing with these kinds of errors.

Reducing the error rate for an application often involves two antagonistic efforts:

- Increasing accuracy or precision (minimizing false positives)
- Increasing coverage or recall (minimizing false negatives).

Errors

Summary

Regular expressions play a surprisingly large role

Sophisticated sequences of regular expressions are often the first model for any text processing text

For many hard tasks, we use machine learning classifiers

But regular expressions are used as features in the classifiers

Can be very useful in capturing generalizations

Basic Text Processing

WORD TOKENIZATION

Text Normalization

Every NLP task needs to do text normalization:

1. Segmenting/tokenizing words in running text
2. Normalizing word formats
3. Segmenting sentences in running text

I do uh main- mainly business data processing

- Fragments, filled pauses

Seuss's cat in the hat is different from other cats!

- **Lemma**: same stem, part of speech, rough word sense
  - cat and cats = same lemma
- **Wordform**: the full inflected surface form
  - cat and cats = different wordforms

How many words?

they lay back on the San Francisco grass and looked at the stars and their

Type: an element of the vocabulary.

Token: an instance of that type in running text.

How many?

- 15 tokens (or 14)
- 13 types (or 12) (or 11?)

How many words?

N = number of tokens

V = vocabulary = set of types

|*V*| is the size of the vocabulary

	Tokens = N	Types = |V|
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
Google N-grams	1 trillion	13 million

Church and Gale (1990): $|V| > O(N^{½})$

How many words?

Simple Tokenization in UNIX

(Inspired by Ken Church's UNIX for Poets.)

Given a text file, output the word tokens and their frequencies

```
tr -sc 'A-Za-z' '\n' < shakes.txt
     | sort
     | uniq –c
     | sort -nr
```

Change all non-alpha to newlines

Sort in alphabetical order

Merge and count each type

Sort numerically descending

The first step: tokenizing

```
tr -sc 'A-Za-z' '\n' < shakes.txt | head
```

```
THE
SONNETS
by
William
Shakespeare
From
fairest
creatures
We
...
```

The second step: sorting

```
tr -sc 'A-Za-z' '\n' < shakes.txt | sort | head
```

```
A
A
A
A
A
A
A
A
A
...
```

More counting

Merging upper and lower case

```
tr 'A-Z' 'a-z' < shakes.txt | tr –sc 'A-Za-z' '\n' | sort |
uniq –c
```

Sorting the counts

```
tr 'A-Z' 'a-z' < shakes.txt | tr –sc 'A-Za-z' '\n' | sort | uniq –c |
sort –n –r
```

```
23243 the
22225 i
18618 and
16339 to
15687 of
12780 a
12163 you
10839 my
10005 in
8954  d
```

What happened here?

Issues in Tokenization

Finland's capital	→	Finland Finlands Finland's ?
what're, I'm, isn't	→	What are, I am, is not
Hewlett-Packard	→	Hewlett Packard ?
state-of-the-art	→	state of the art ?
Lowercase	→	lower-case lowercase lower case ?
San Francisco	→	one token or two?
m.p.h., PhD.	→	??

French

- ***L'ensemble*** $\rightarrow$ one token or two?
  - ***L*** ? ***L’*** ? ***Le*** ?
  - Want ***l’ensemble*** to match with ***un ensemble***

German noun compounds are not segmented

- ***Lebensversicherungsgesellschaftsangestellter***
- ‘life insurance company employee’
- German information retrieval needs **compound splitter**

Tokenization: language issues

Tokenization: language issues

Chinese and Japanese no spaces between words:

- 莎拉波娃现在居住在美国东南部的佛罗里达。
- 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
- Sharapova now lives. in US southeastern Florida

Further complicated in Japanese, with multiple alphabets intermingled

- Dates/amounts in multiple formats

フォーチュン500社は情報不足のため時間あた$500K(約6,000万円)

Katakana Hiragana Kanji Romaji

End-user can express query entirely in hiragana!

Word Tokenization in Chinese

Also called **Word Segmentation**

Chinese words are composed of characters

- Characters are generally 1 syllable and 1 morpheme.
- Average word is 2.4 characters long.

Standard baseline segmentation algorithm:

- Maximum Matching (also called Greedy)

Maximum Matching Word Segmentation Algorithm

Given a wordlist of Chinese, and a string:

1) Start a pointer at the beginning of the string
2) Find the longest word in dictionary that matches the string starting at pointer
3) Move the pointer over the word in string
4) Go to 2

Max-match segmentation illustration

Thecatinthehat

Thetabledownthere

the cat in the hat

the table down there

theta bled own there

Doesn’t generally work in English!

But works suprisingly well in Chinese

- 莎拉波娃现在居住在美国东南部的佛罗里达。
- 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达

Modern probabilistic segmentation algorithms even better

Basic Text Processing

WORD NORMALIZATION AND STEMMING

Normalization

Need to “normalize” terms

- Information Retrieval: indexed text & query terms must have same form.
  - We want to match ***U.S.A.*** and ***USA***

We implicitly define equivalence classes of terms

- e.g., deleting periods in a term

Alternative: asymmetric expansion:

- Enter: ***window*** Search: ***window, windows***
- Enter: ***windows*** Search: ***Windows, windows, window***
- Enter: ***Windows*** Search: ***Windows***

Potentially more powerful, but less efficient

Case folding

Applications like IR: reduce all letters to lower case

- Since users tend to use lower case
- Possible exception: upper case in mid-sentence?
  - e.g., ***General Motors***
  - ***Fed*** vs. ***fed***
  - ***SAIL*** vs. ***sail***

For sentiment analysis, MT, Information extraction

- Case is helpful (***US*** versus ***us*** is important)

Lemmatization

Reduce inflections or variant forms to base form

- *am, are, is* $\rightarrow$ *be*
- *car, cars, car's, cars'* $\rightarrow$ *car*

the boy's cars are different colors $\rightarrow$ *the boy car be different color*

Lemmatization: have to find correct dictionary headword form

Machine translation

- Spanish quiero ('I want'), quieres ('you want') same lemma as querer 'want'

Morphology

Morphemes:

- The small meaningful units that make up words
- **Stems**: The core meaning-bearing units
- **Affixes**: Bits and pieces that adhere to stems
- Often with grammatical functions

Stemming

Reduce terms to their stems in information retrieval

Stemming is crude chopping of affixes

- language dependent
- e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

for example compressed and compression are both accepted as equivalent to compress.

→

for exampl compress and compress ar both accept as equival to compress

Porter's algorithm
The most common English stemmer

Step 1a

```
sses → ss    caresses → caress
ies  → i     ponies   → poni
ss   → ss    caress   → caress
s    → ø     cats     → cat
```

Step 1b

```
(*v*)ing → ø  walking   → walk
              sing      → sing
(*v*)ed  → ø  plastered → plaster
…
```

Step 2 (for long stems)

```
ational→ ate  relational→ relate
izer→ ize     digitizer → digitize
ator→ ate     operator  → operate
…
```

Step 3 (for longer stems)

```
al   → ø  revival    → reviv
able → ø  adjustable → adjust
ate  → ø  activate   → activ
…
```

```
(*v*)ing → ø  walking   → walk
              sing      → sing
```

Viewing morphology in a corpus
Why only strip –ing if there is a vowel?

```
(*v*)ing → ø  walking  → walk
              sing     → sing
```

```
tr -sc 'A-Za-z' '\n' < shakes.txt | grep 'ing$' | sort | uniq -c | sort –nr
```

1312 King	548 being
548 being	541 nothing
541 nothing	152 something
388 king	145 coming
375 bring	130 morning
358 thing	122 having
307 ring	120 living
152 something	117 loving
145 coming	116 Being
130 morning	102 going

```
tr -sc 'A-Za-z' '\n' < shakes.txt | grep '[aeiou].*ing$' | sort | uniq -c | sort –nr
```

Viewing morphology in a corpus
Why only strip –ing if there is a vowel?

47

Some languages requires complex morpheme segmentation

- Turkish
- Uygarlastiramadiklarimizdanmissinizcasina
- `(behaving) as if you are among those whom we could not civilize'
- Uygar `civilized' + las `become'

  + tir `cause' + ama `not able'

  + dik `past' + lar 'plural'

  + imiz 'p1pl' + dan 'abl'

  + mis 'past' + siniz '2pl' + casina 'as if'

Dealing with complex morphology is sometimes necessary

Basic Text Processing

SENTENCE SEGMENTATION AND DECISION TREES

Sentence Segmentation

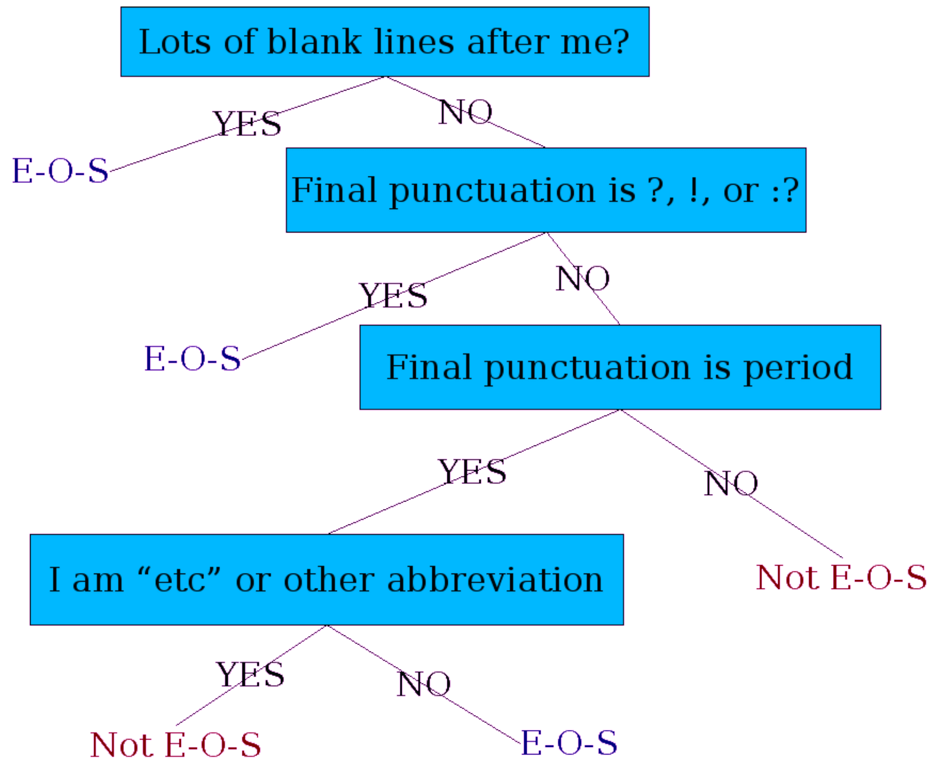
!, ? are relatively unambiguous

Period "." is quite ambiguous

- Sentence boundary
- Abbreviations like Inc. or Dr.
- Numbers like .02% or 4.3

Build a binary classifier

- Looks at a "."
- Decides EndOfSentence/NotEndOfSentence
- Classifiers: hand-written rules, regular expressions, or machine-learning



Determining if a word is end-of-sentence: a Decision Tree

Case of word with ".": Upper, Lower, Cap, Number

Case of word after ".": Upper, Lower, Cap, Number

Numeric features

- Length of word with "."
- Probability(word with "." occurs at end-of-s)
- Probability(word after "." occurs at beginning-of-s)

More sophisticated decision tree features

A decision tree is just an if-then-else statement

The interesting research is choosing the features

Setting up the structure is often too hard to do by hand

- Hand-building only possible for very simple features, domains
  - For numeric features, it's too hard to pick each threshold
- Instead, structure usually learned by machine learning from a training corpus

Implementing Decision Trees

We can think of the questions in a decision tree

As features that could be exploited by any kind of classifier

- Logistic regression
- SVM
- Neural Nets
- etc.

Decision Trees and other classifiers