Text Classification with Naïve Bayes

The Task of Text Classification

Text Classification: definition

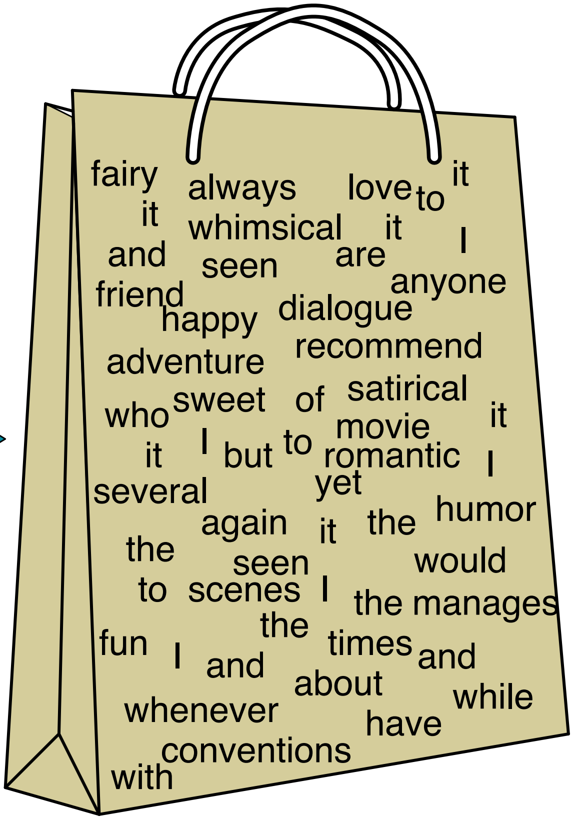
- *Input*:
  - a document d
  - a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
- *Output*: a predicted class $c \in C$

Naïve Bayes Intuition

- Simple (“naïve”) classification method based on Bayes rule
- Relies on very simple representation of document
  - Bag of words

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

Parameter Estimation and Smoothing

Learning the Multinomial Naïve Bayes Model

- First attempt: maximum likelihood estimates
  - simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{doccount(C = c_j)}{N_{doc}}$$

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

Parameter estimation

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

fraction of times word w_i appears among all words in documents of topic c_j

- Create mega-document for topic *j* by concatenating all docs in this topic
  - Use frequency of *w* in mega-document

Problem with Maximum Likelihood

- What if we have seen no training documents with the word ***fantastic*** and classified in the topic **positive** (***thumbs-up)***?

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

- Zero probabilities cannot be conditioned away, no matter the other evidence!

$$c_{MAP} = \operatorname{argmax}_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

Laplace (add-1) smoothing for Naïve Bayes

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

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

Multinomial Naïve Bayes: Learning

- From training corpus, extract *Vocabulary*

- Calculate $P(c_j)$ terms
  - For each c_j in C do

    $docs_j \leftarrow$ all docs with class $=c_j$

$$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

- Calculate $P(w_k \mid c_j)$ terms
  - $Text_j \leftarrow$ single doc containing all $docs_j$
  - For each word w_k in *Vocabulary*

    $n_k \leftarrow$ # of occurrences of w_k in $Text_j$

$$P(w_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$$

Text Classification and Naïve Bayes

Precision, Recall, and the F measure

The 2-by-2 contingency table

	correct	not correct
selected	tp	fp
not selected	fn	tn

Precision and recall

- **Precision**: % of selected items that are correct
  Recall: % of correct items that are selected

	correct	not correct
selected	tp	fp
not selected	fn	tn

A combined measure: F

- A combined measure that assesses the P/R tradeoff is F measure (weighted harmonic mean):

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

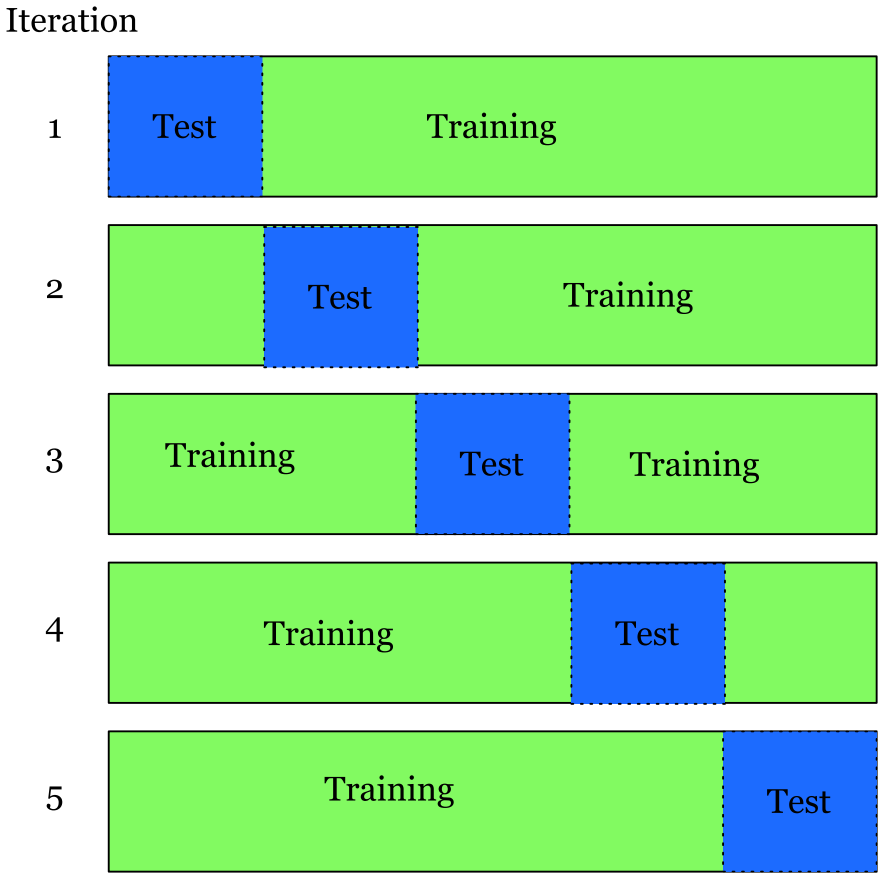
- The harmonic mean is a very conservative average
- People usually use balanced F1 measure
  - i.e., with $\beta = 1$ (that is, $\alpha = ½$): $F = 2PR/(P+R)$

Text Classification and Naïve Bayes

Text Classification: Evaluation

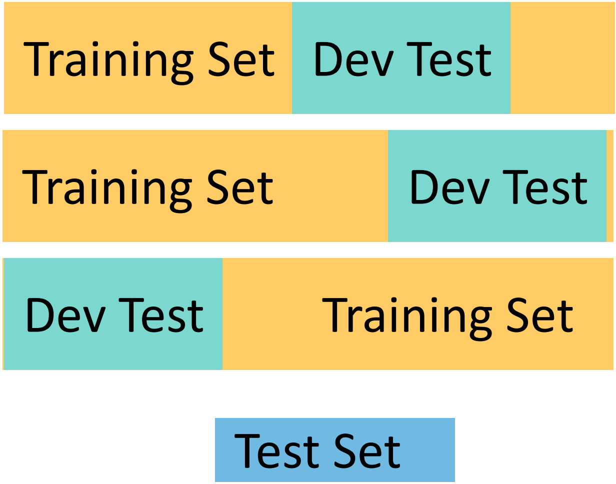
Cross-Validation

- 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

- Metric: P/R/F1 or Accuracy
- Unseen test set
  - avoid overfitting ('tuning to the test set')
  - more conservative estimate of performance
- Cross-validation over multiple splits
  - Handle sampling errors from different datasets
  - Pool results over each split
  - Compute pooled dev set performance



Text Classification and Naïve Bayes

Text Classification: Practical Issues

The Real World

- Gee, I'm building a text classifier for real, now!
- What should I do?

No training data? Manually written rules

If (wheat or grain) and not (whole or bread) then

Categorize as grain

- Need careful crafting
  - Human tuning on development data
  - Time-consuming: 2 days per class

Very little data?

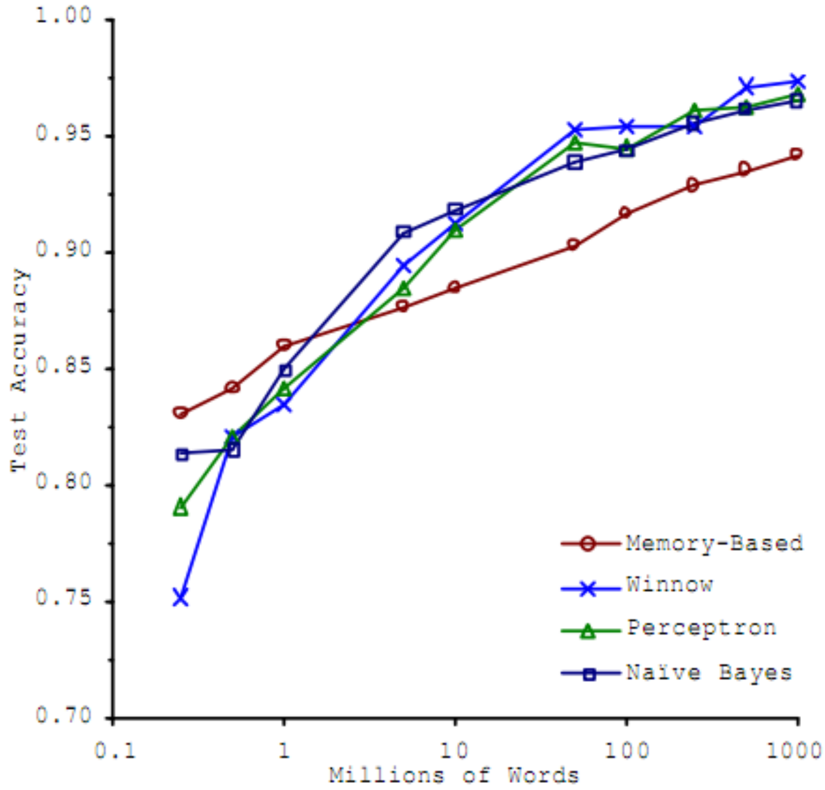
- Use Naïve Bayes
  - Naïve Bayes is a "high-bias" algorithm (Ng and Jordan 2002 NIPS)
- Get more labeled data
  - Find clever ways to get humans to label data for you
- Try semi-supervised training methods:
  - Bootstrapping, EM over unlabeled documents, …

A reasonable amount of data?

- Perfect for all the clever classifiers
  - SVM
  - Regularized Logistic Regression
- You can even use user-interpretable decision trees
  - Users like to hack
  - Management likes quick fixes

Accuracy as a function of data size

- With enough data
  - Classifier may not matter



Brill and Banko on spelling correction

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`
`a^b`	The pattern a carat b	`Look up a^b now`

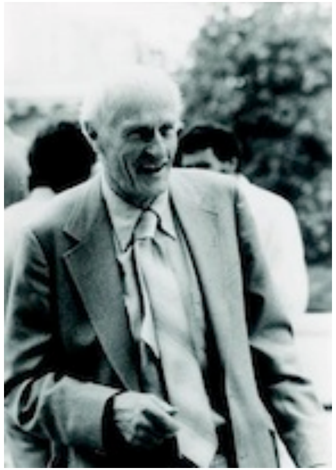
Regular Expressions: More Disjunction

- Woodchucks is another name for groundhog!
- The pipe | for disjunction

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

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



Stephen C Kleene

Kleene *, Kleene +

Regular Expressions: Anchors ^ $

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

Example

- Find me all instances of the word "the" in a text.

`the`

Misses capitalized examples

`[tT]he`

Incorrectly returns `other` or `theology`

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

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 (Type I)
  - Not matching things that we should have matched (The)
    - False negatives (Type II)

Errors cont.

- 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).

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

Regular Expressions

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

How many words?

- 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

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

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

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
```

Change all non-alpha to newlines

Sort in alphabetical order

Merge and count each type

```
1945 A
  72 AARON
  19 ABBESS
   5 ABBOT
 ... ...
```

```
25 Aaron
 6 Abate
 1 Abates
 5 Abbess
 6 Abbey
 3 Abbot
.... …
```

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. → ??

Tokenization: language issues

- French
  - ***L'ensemble*** → 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

- 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 — the cat in the hat
- Thetabledownthere — the table down there

  theta bled own there

- Doesn’t generally work in English!

- But works astonishingly 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 tokenization

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
…
```

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

```
(*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
```

Dealing with complex morphology is sometimes necessary

- 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’

Basic Text Processing

Word Normalization and Stemming

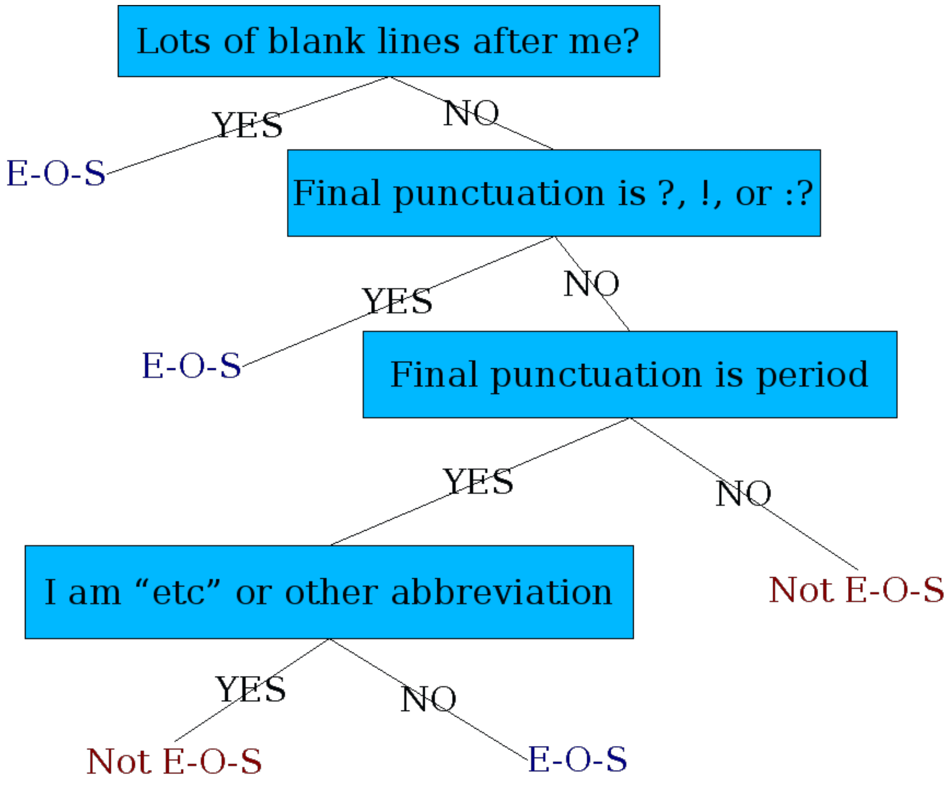
Basic Text Processing

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



More sophisticated decision tree features

- 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)

Implementing Decision Trees

- 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

Decision Trees and other classifiers

- 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.