Pre-processing for Text Mining and Traditional Machine Learning

Text and Web Mining (IS6751)

School of Communication and Information

Overview

- Introduce text-processing for Text Mining.
  - Text normalization
  - POS tagging and phrase recognition

    POS标记和短语识别
- Introduce a traditional machine learning algorithm.
  - Cover supervised learning concepts in Machine Learning
  - Naïve Bayesian algorithm for Text classification
  - Evaluation methods

Collecting documents

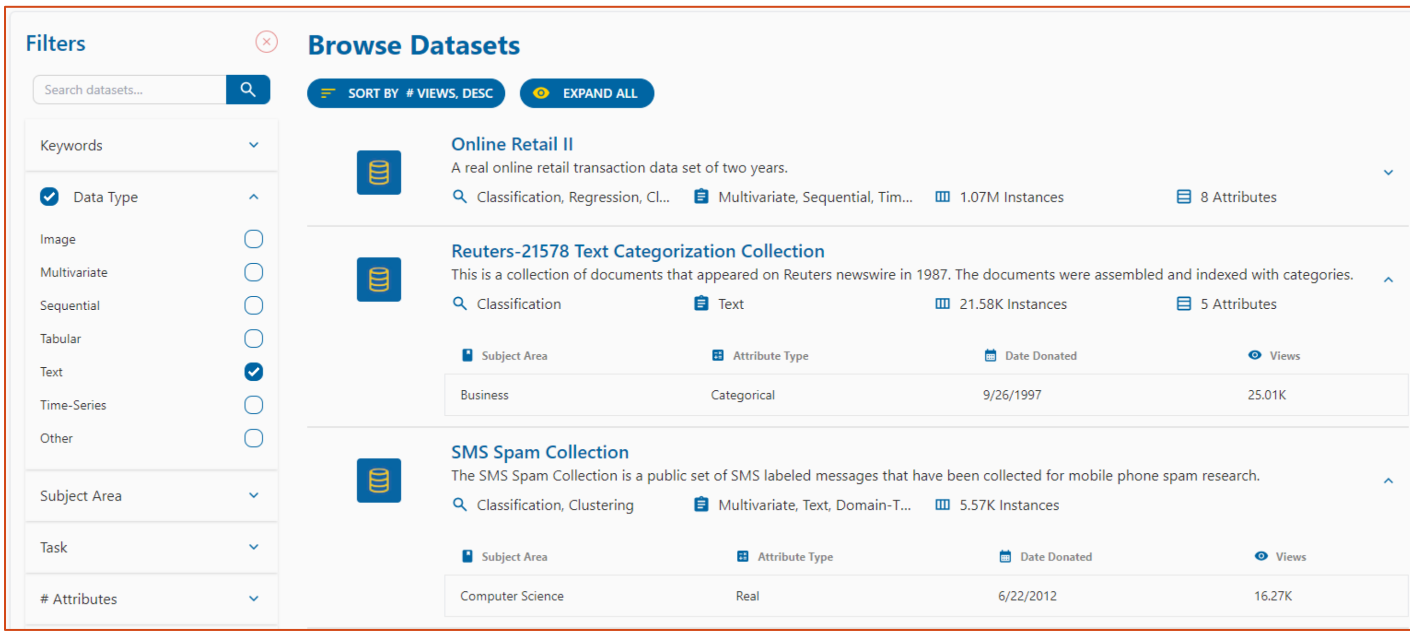
- The first step in text mining is to collect the ***data***.
- In some applications, need to have a data collection process.
  - For a web application, use a software tool such as **a Web crawler** 网络爬虫 that collects the documents.
- For research and development of text-mining techniques, more generic data may be necessary, usually called a **corpus**. 本体
  - E.g., the **Reuters 21578 corpus** has a collection of Reuters English news stories **tagged with economic subject categories**, such as Earn, Acquisition, Grain, Crude, etc.

    路透社21578语料库收集了路透社英语新闻报道，标有经济主题类别，如收益、收购、谷物、原油等。

Collecting documents

- **The UC Irvine Machine Learning Repository** currently maintains 674 data sets as a service to the machine learning community. http://archive.ics.uci.edu/ml/
  - E.g., amazon reviews, email spam, and sentiment-labelled sentences.
- **Kaggle** (data/text mining competition) also provides various data sets.

The UC Irvine Machine Learning Repository



Text Normalization

- Every NLP task including text mining needs to do **text normalization**:
  - Segmenting sentences in running text
  - Segmenting/tokenizing words in running text
  - Normalizing word formats
    - convert to standard or common forms.

"NLTK is a leading platform for building Python programs to work with human language data." →
['NLTK', 'is', 'a', 'leading', 'platform', 'for', 'building', 'Python', 'programs', 'to', 'work', 'with', 'human', 'language', 'data', '.']

Tokenization

- Breaks the stream of characters into **words** or **tokens**.
  对于熟悉语言结构的人来说是微不足道的。
  - Trivial for a person familiar with the language structure.
- A computer program, though, being linguistically challenged, would find the task more complicated.
  然而，计算机程序在语言上受到挑战，会发现任务更加复杂。
- The reason is that certain characters are sometimes **token delimiters** and sometimes not, depending on the application.
- The **white space** characters **space, tab,** and **newline** are always delimiters and are not counted as tokens.
- The characters **( ) < > ! ? "** are always delimiters and may also be tokens.

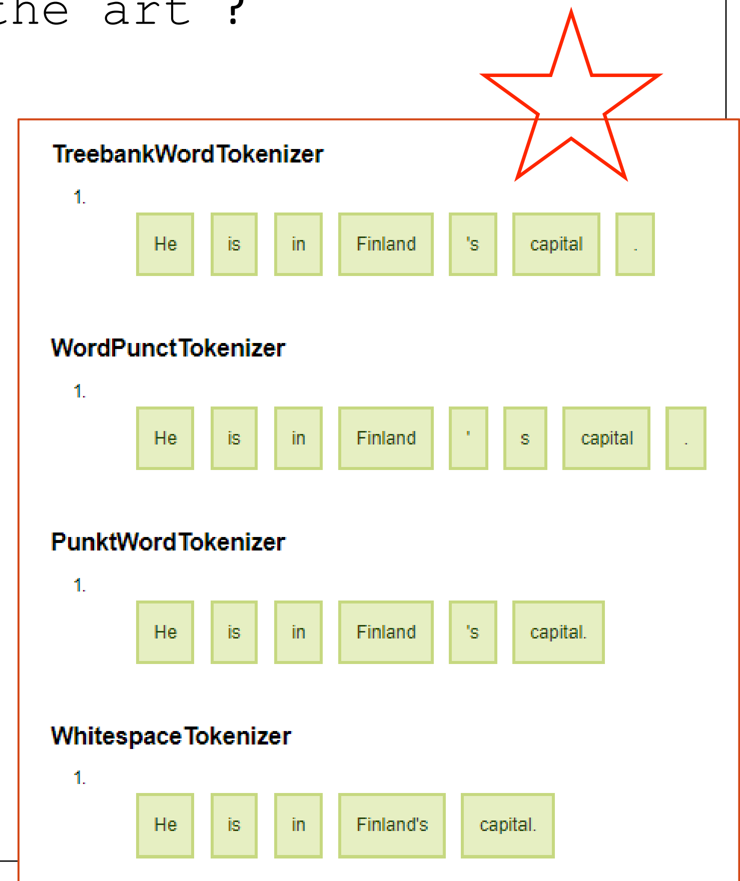
Tokenization

- The characters . , : - ’ may or may not be delimiters, depending on their environment.
- Example cases
  - Numbers: 100,000 or 333-1221
  - Abbreviations: Dr.
  - Part of the current token: isn’t or D’angelo
  - Possessive: Tess’
- To get the best possible features, one may need to customize the tokenizer for the available text.
  - E.g., part:123-4567
- The tokenization process is language-dependent.

Example Issues in Tokenization

- **Finland's** capital → Finland 's or 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 ?
- San Francisco → one token or two?

- **Online Word Tokenization with Python NLTK**
  - http://text-processing.com/demo/tokenize/
  - E.g., "He is in Finland's capital."
- Replacing *contractions* with their expansions before Tokenization may help.
  - isn't -> is not
  - can't -> can not



Stop Words

- Stop-words are words that from non-linguistic view do not carry information. 从非语言学的角度来看，停止词是不带信息的词。
  - They have mainly functional role.
- Natural language dependent – examples:
  - English: A, ABOUT, ABOVE, ACROSS, AFTER, AGAIN, AGAINST, ALL, ALMOST, ALONE, ALONG, ALREADY, ALSO, …
- Whether or not removing stop words is necessary is application-dependent.
  - Topic classification: may remove stop words
  - Author classification: may keep stop words

Case folding

- Reduce all letters to lower case
  - E.g., Car, CAR -> car
  - May lose the original meaning of words with case folding.
    - E.g.,:
      - ***General Motors* vs. *general motors***
      - ***Fed*** vs. ***fed***
        - Fed : Federal Reserve
      - ***SAIL*** vs. ***sail***
        - SAIL: Stanford Artificial Intelligence Language.
- For Sentiment Analysis and Information Extraction
  - Case is helpful (***US*** versus ***us*** is important).
  - E.g., *"US won a gold medal"; "They like US." vs. "They like us."*
- Whether or not this step is necessary is application-dependent.

Normalization

- Converts each of the tokens to **a standard form**, a process usually referred to as ***stemming*** or ***lemmatization***.
  将每个标记转换为标准形式，这个过程通常称为词干提取或词形还原。
- One effect of normalization is to reduce the number of distinct types (i.e., unique terms) in a text corpus and to **increase the frequency of occurrence of some individual types**.
  - E.g., *types and typed* → *type*
- For classification algorithms that take frequency into account, this can sometimes make a difference.
  - Reduce feature dimension (the number of features)
    缩小特征尺寸
  - May ease term scarcity issue (matrix is sparse, i.e., too many zero).
    可能会缓解短期稀缺问题
- Whether or not this step is necessary is application-dependent.

Some Terms: Morphology

形态学

- **Morphemes**: 语素
  - The small meaningful units that make up words.
  - E.g., un-like-ly contains three.
  - **Stems**: The main part of a word that stays the same when endings are added to it.
    - E.g., **writ** is the stem of writes, writing, and written.
  - **Affixes**: Bits and pieces that adhere to stems (i.e., the prefix and suffix)
    - Often with grammatical functions
    - E.g., like**s**

Stemming

- Reduces terms to their stems
  - E.g., used in information retrieval and text mining applications.
- *Stemming* is **crude chopping of affixes** 粗糙的词缀切割
  - language dependent
  - E.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

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

Original Input

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

Stemmed output

Porter's algorithm
The most common English stemmer

old already

- The best-known algorithm is the **Porter stemmer** (www.tartarus.org/~martin/PorterStemmer; http://text-processing.com/demo/stem/).

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
              loving    → lov
              sing      → sing
              king      → king
(*v*)ed  → ø  plastered → plaster
…
```

```
Note: v =[aeiou]
```

Step 2 (for long stems)

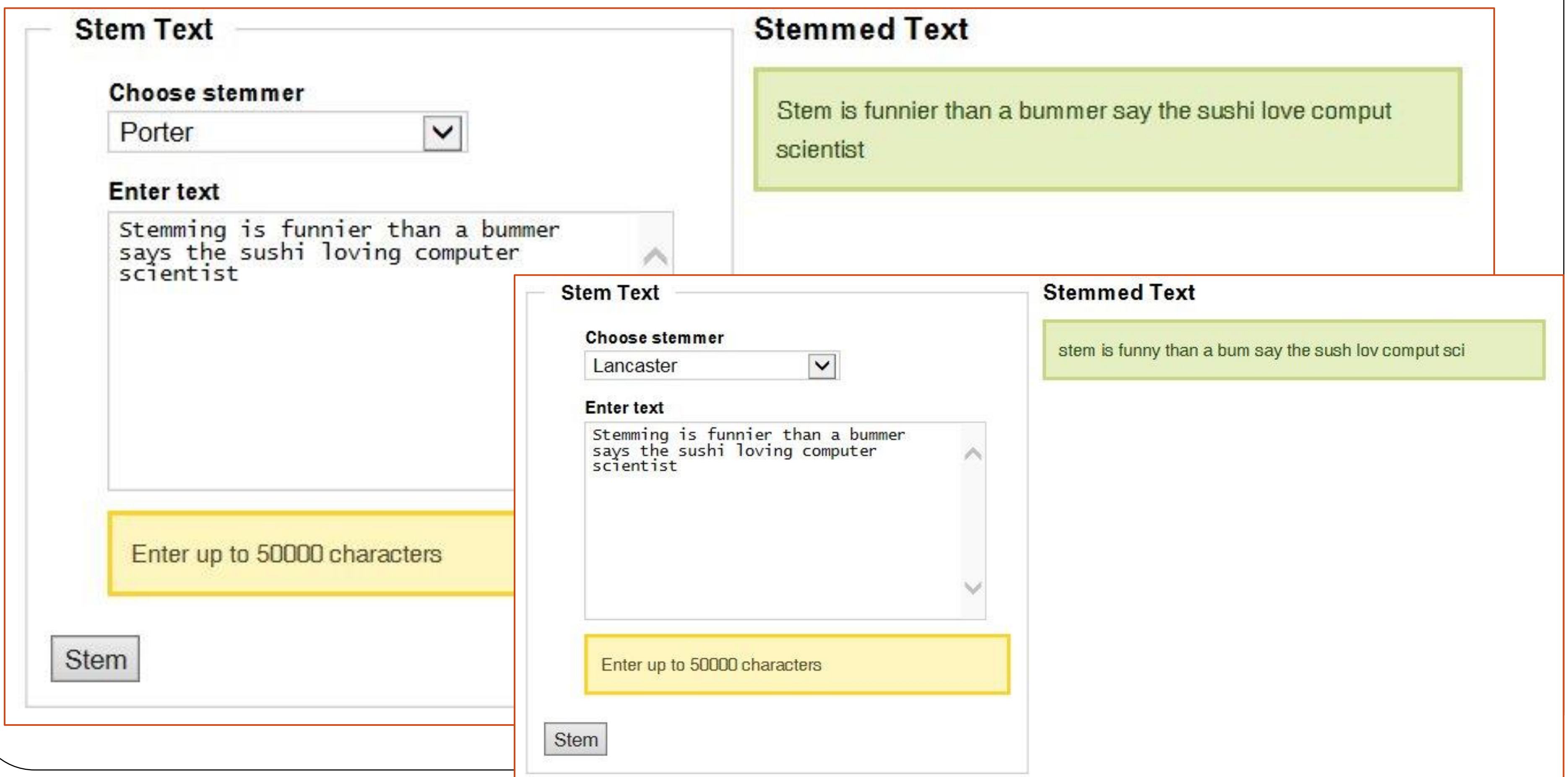
```
ational→ ate  relational→ relate
izer→ ize     digitizer → digitize
ator→ ate     operator  → operate
…
```

Step 3 (for longer stems)

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

Porter's algorithm
The most common English stemmer

- **Stemming with Python NLTK** (http://text-processing.com/demo/stem/).



Lemmatization – Stemming to a Root

- Converts to a **root form** with no inflectional or derivational prefixes and suffixes. 转换为没有转折或派生前缀和后缀的根形式。
  - **Inflectional suffixes** are endings such as "-ed", "-ing", "s", etc.
    - Create different grammatical forms of the same word 屈折后缀是诸如"-ed"、"-ing"、"s"等词尾。为同一个单词创建不同的语法形式
  - **Derivational suffixes** are endings such as "-ism", "-ful", "-fy", etc. 派生后缀是"-ism"、"-ful"、"-fy"等结尾。
    - Change the meaning of the word
  - E.g., "denormalization" is reduced to the stem "norm".
  - E.g., "reapplied", "applications" -> "apply"
- Words with the same core meaning are coalesced.
- The end result of such **aggressive stemming** is to reduce the number of types in a text collection very drastically, thereby making distributional statistics more reliable.

这种激进的词干的最终结果是大幅减少文本集合中的类型数量，从而使分布统计更加可靠

Stanford CoreNLP Lemmatization:
denormalization -> denormalization
reapplications -> reapplication
reapplied -> reapply

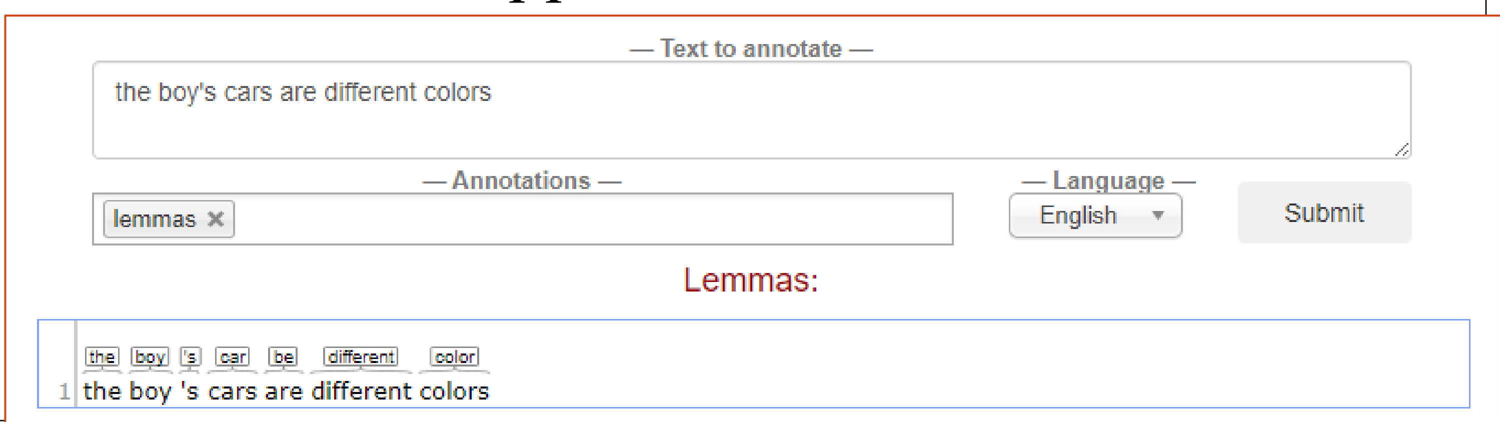
Lemmatization

- Additional examples:
  - Reduce variant forms to base form.
    - *am, are, is* $\rightarrow$ *be*
    - *the boy's cars are different colors* $\rightarrow$ *the boy 's car be different color*
- Lemmatization: have to find **correct dictionary headword form** (i.e., root or lemma form)

正确的字典标题形式

- E.g., Stanford CoreNLP supports lemmatization.

https://corenlp.run/
(online version)



Sentence Boundary Determination

- For more sophisticated linguistic parsing, the algorithms often require **a complete sentence as input**. 对于更复杂的语言解析，算法通常需要一个完整的句子作为输入。
  - E.g., Sentence-level sentiment analysis
- We shall also see other information extraction algorithms that operate on a sentence at a time.
- Tokenization function also uses a complete sentence as input.
- Sentence boundary determination is essentially the problem of deciding ***which instances of a period (.) followed by whitespace are sentence delimiters and which are not*** since we assume that the characters **?** and **!** are unambiguous sentence boundaries. 句子边界确定本质上是决定哪些句号 (.) 后跟空格的实例是句子分隔符，哪些不是，因为我们假设字符 ? 和 ! 是明确的句子边界。

Input: "this's a sent tokenize test. this is sent two. is this sent three? sent 4 is cool! Now it's your turn."

Sentence Segmentation

- !, ? are relatively unambiguous.
- Period "." is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02% or 4.3

```
from nltk.tokenize import sent_tokenize

text = "this's a sent tokenize test. this is sent two. is this sent three? sent 4 is cool! Now it's your turn."
sent_tokenize_list = sent_tokenize(text)

print("The length of sentences", " = ", len(sent_tokenize_list))
print(sent_tokenize_list)
```

```
The length of sentences  =  5
["this's a sent tokenize test.", 'this is sent two.', 'is this sent three?', 'sent 4 is cool!', "Now it's your turn."]
```

A	…	Zyzzyva	# of Noun	# of Verb	Author?
1	…	0	5	5	0
0	…	1	1	10	1
1	…	0	3	2	2

POS

Part-of-Speech Tagging

- If no further *linguistic analysis* is necessary, one might proceed directly to feature generation. 如果不需要进一步的语言分析，人们可能会直接进行特征生成。
- However, if the goal is more specific, say recognizing names of **people**, **places**, and **organizations**, it is desirable to perform additional linguistic analyses of the text and extract more sophisticated features. 然而，如果目标更具体，例如识别人员、地点和组织的名称，最好对文本进行额外的语言分析，并提取更复杂的特征。
  - E.g., San Francisco: San/NNP Francisco/NNP (NNP: Proper noun, singular)
- Most English grammars would have as a minimum **noun, verb, adjective, adverb, preposition,** and **conjunction**.
  大多数英语语法都有最小 名词、动词、形容词、副词、介词和连词。
  - POS can be used for feature reduction, e.g., use only verb, adjective, and adverb for sentiment classification.
  - Distribution of POS (e.g., the numbers of noun and verb) can be used for author, gender, and document genre (formal vs. informal) classification.

Part-of-Speech Tagging

- A set of 36 POS categories used in the **Penn Tree Bank**, constructed from the Wall Street Journal corpus (see Table 2.4).
  - A **tree bank** is a parsed text corpus that annotates sentence structure, such as POS and phrases. 树库是一个解析的文本语料库，它注释句子结构，如POS和短语。
- Almost all POS taggers have been trained on the Wall Street Journal corpus available from LDC (Linguistic Data Consortium, www.ldc.upenn.edu)
  - E.g., *I love you* -> *I* (**personal pronoun**) *love* (**verb**, not noun)

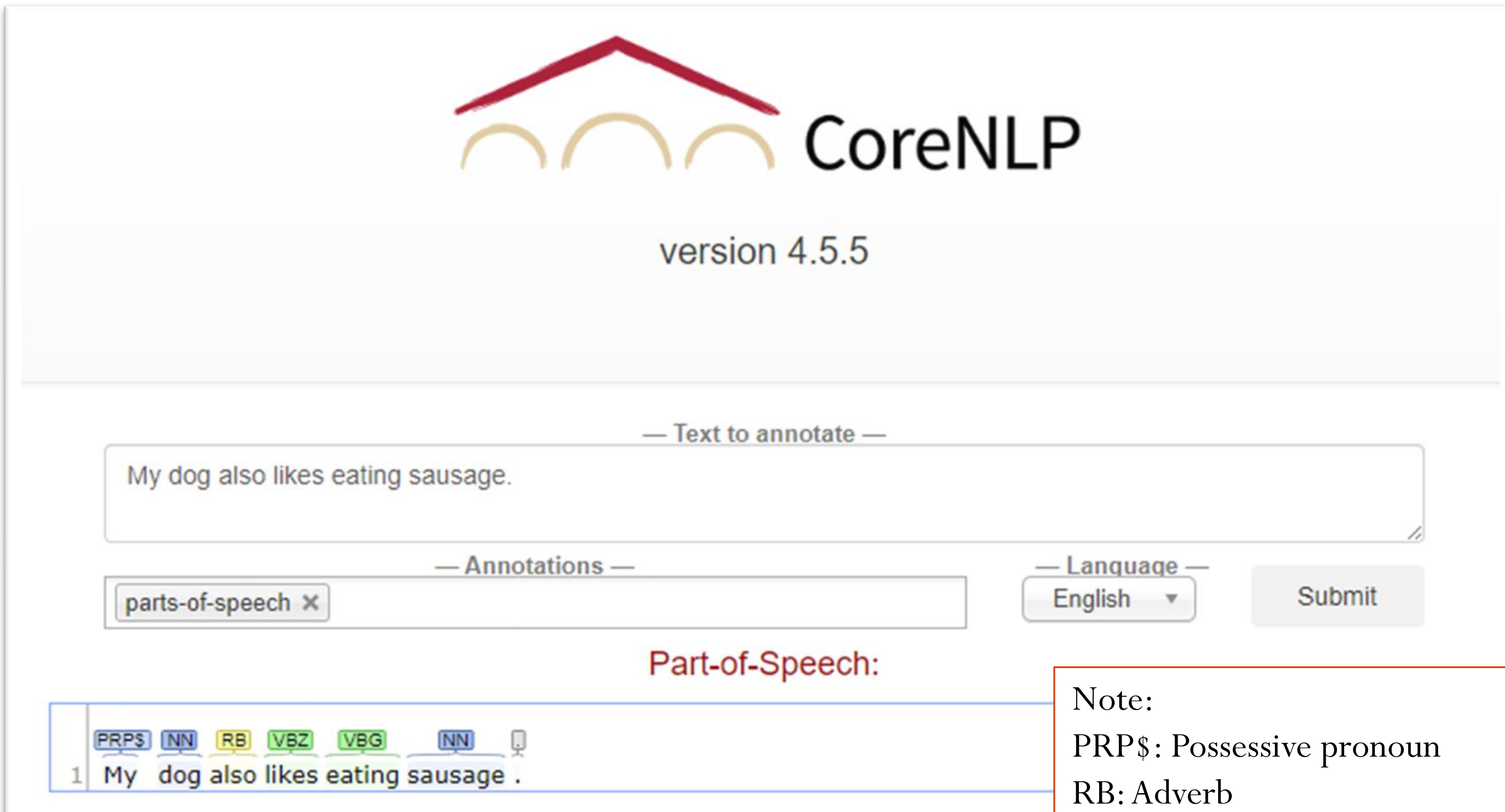
Table 2.4 Some of the categories in the penn tree bank POS set

Tag	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
POS	Possessive ending
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
WDT	Wh-determiner

All the POS categories:
http://cs.nyu.edu/grishman/jet/guide/PennPOS.html

Part-of-Speech Tagging

- **The Stanford Parser: online parser**
  https://corenlp.run/



Note:
PRP$: Possessive pronoun
RB: Adverb
VBZ: Verb, 3rd person singular present

Word Sense Disambiguation

A	…	bore: person	bore: hole	…	Zyzzyva	Author?
1	…	1	0	…	0	0
0	…	0	1	…	1	1
1	…	1	0	…	0	2

- English words are very often **ambiguous** as to their meaning or reference.
- For the example "bore" in a sentence, cannot tell without context, even after POS tagging
  - "He is a bore." : "bore" is referring to a person
  - "The bore is not large enough." : referring to a hole
  - In the above sentences, "bore" is tagged as NN (Noun, singular or mass).
  - If possible, we may have the following features: bore:person and bore:hole
- There are no algorithms that can completely disambiguate a text.
- Unless a particular text-mining project requires word sense disambiguation, it is best to proceed without such a step.

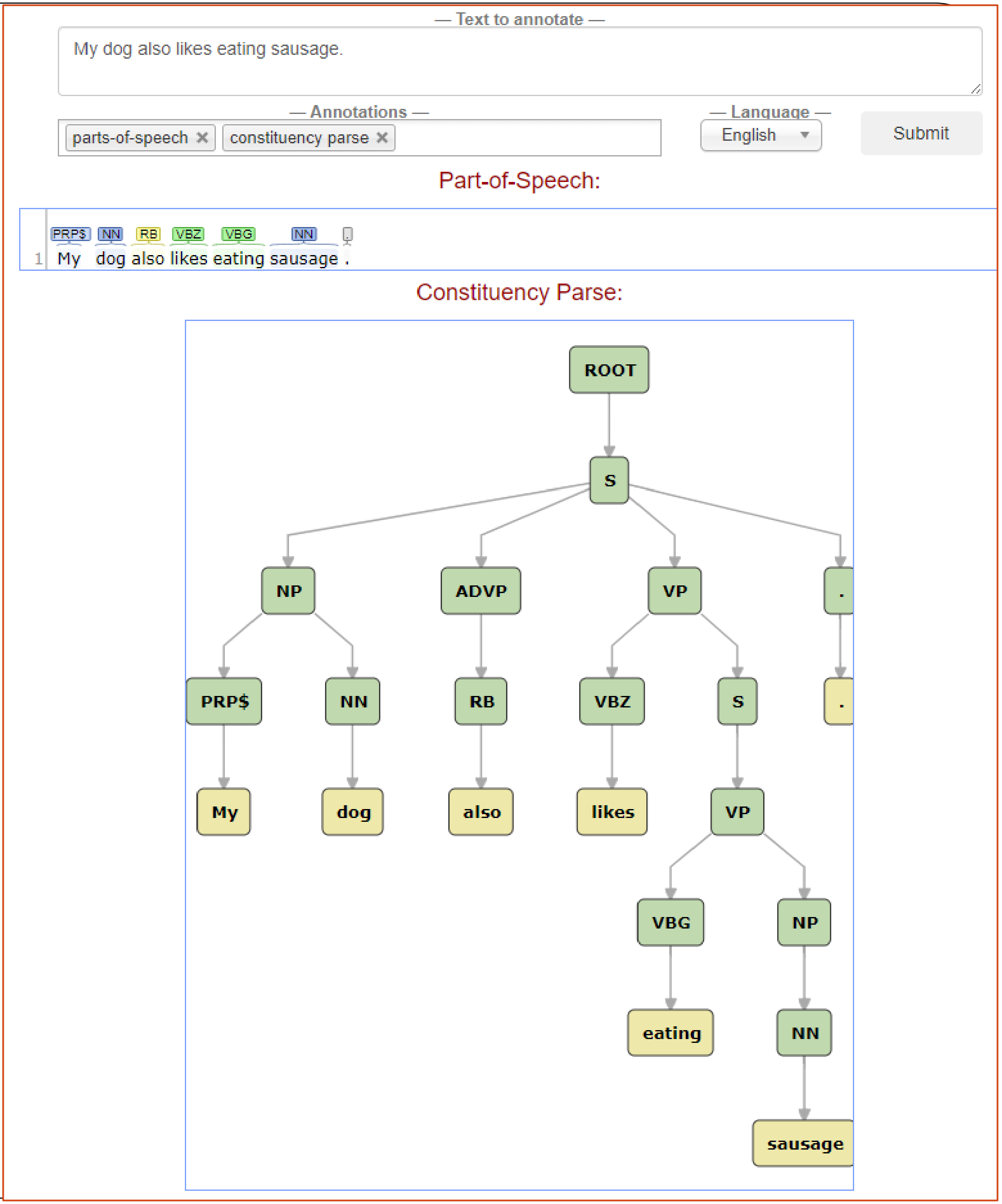
除非特定的文本探矿项目需要词义消除歧义，否则最好不要采取这样的步骤

Phrase Recognition

- Once the tokens of a sentence have been assigned **POS tags**, the next step is to group individual tokens into units, generally called **phrases**.
  一旦句子的令牌被分配了POS标签，下一步是将单个令牌分组为单位，通常称为短语。
- Phrase recognition systems are supposed to scan a text and mark the beginnings and ends of phrases, of which the most important are **noun phrases**, **verb phrases**, and **prepositional phrases**.
  短语识别系统应该扫描文本并标记短语的开头和结尾，其中最重要的是名词短语、动词短语和介词短语。
- **Name Entity Recognition** is **the recognition of particular types of proper noun phrases**, specifically persons, organizations, and locations, sometimes along with money, dates, times, and percentages.
  名称实体识别是对特定类型的专有名词短语的识别，特别是人、组织和地点，有时还与金钱、日期、时间和百分比一起。

Phrase Recognition

- **The Stanford Parser: online parser** https://corenlp.run/
- Shows **a full parse of a sentence**.
- Each word in a sentence is connected to a single structure, called a **parse tree**.



Classification Methods: Supervised Machine Learning

- *Input:*
  - a document d
  - a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
  - a training set of m hand-labeled documents $(d_1, c_1), (d_2, c_1), (d_3, c_2), \ldots, (d_m, c_j)$
- *Output:*
  - a learned classifier $\gamma : d \rightarrow c$

Scoring by Probabilities: Naive Bayesian classifier

- Simple (“naïve”) classification method based on Bayes rule
- Relies on very simple representation of document
  - Bag of words
    - disregards grammar and word order
- A **good dependable baseline** for text classification

The bag of words representation

$\gamma\Big($

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.

$\Big) = c$

- Use verb, adjective, and adverb only.

The bag of words representation

$\gamma($

great	2
love	2
recommend	1
laugh	1
happy	1
...	...

$) = c$

Bayes' Rule Applied to Documents and Classes

- For a document *d* and a class *c* *(e.g., pos vs. neg)*

P(pos | d) ?

P(neg | d) ?

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

Our *posterior* belief P(C | *D*) is calculated by multiplying our *prior* belief P(C) by the *likelihood* P(D | C) that D will occur if C is true.

Naïve Bayes Classifier

Find the argument **c** that maximizes P(c | d).
For instance, if P(pos | d) is higher than P(neg | d), return "pos" class, where $C = \{\text{pos}, \text{neg}\}$.

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}}\, P(c \mid d)$$

MAP is "maximum a posteriori" = most likely class

$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(d \mid c)P(c)}{P(d)}$$

Bayes Rule

$$= \underset{c \in C}{\operatorname{argmax}}\, P(d \mid c)P(c)$$

Dropping the denominator

Naïve Bayes Classifier

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(d \mid c) P(c)$$

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

Document d represented as features $x_1 .. x_n$

Could only be estimated if a very, very large number of training examples was available.
E.g., (1, 3, 0, 4, 0, …………, 3)
(1, 2, 4, 0, 3, …………, 2)

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 $P(x_i|c)$ are independent given the class c.

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

E.g., P(good | C); P(bad | C)

Naïve Bayes Classifier

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

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

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

Applying Naive Bayes Classifiers to Text Classification

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

positions $\leftarrow$ all word positions in test document

When an input is "I love this movie",

E.g., For pos class, P(pos) * (P(I|pos) * **P(love|pos)** * P(this|pos) * P(movie|pos)) = 0.7
For neg class, P(neg) * (P(I|neg) * **P(love|neg)** * P(this|neg) * P(movie|neg)) = 0.4

Learning the (Multinomial) Naïve Bayes Model

- From training corpus, extract ***Vocabulary*** ***(V)***.

词汇

- ***P()*** values are estimated simply using the **frequencies** in the data

频率

estimated

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

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

$$c_{NB} = \operatorname*{argmax}_{c_j \in C} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

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 Parameter estimation

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

如果我们没有看到包含单词“fantastic”且分类为积极（竖起大拇指）主题的训练文档，该怎么办?

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

- Zero probabilities will affect results.

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

When an input is “This movie is fantastic”,

For pos class, P(pos) * (P(This | pos) * P(movie | pos) * P(is | pos) * **P(fantastic | pos)**) = 0

Laplace (add-1) smoothing for Naïve Bayes

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

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

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

Laplace (add-1) smoothing: unknown words

- When testing, unknown words (e.g., impetus) can appear.
- Add one extra word to the vocabulary, the **“unknown word” $\mathbf{W_u}$**

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

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

A worked example

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

Priors:

$P(c)$ = 3/4

$P(j)$ = ¼

	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	?

|V| = 6

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

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

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

Underflow Prevention: log space

下溢预防：日志空间

- Multiplying lots of probabilities can result in floating-point underflow.

乘以大量概率可能会导致浮点不足。

- Since $\log(xy) = \log(x) + \log(y)$
  - Better to sum logs of probabilities instead of multiplying probabilities.
- Model is now just max of sum of weights.

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} \log P(c_j) + \sum_{i \in positions} \log P(x_i \mid c_j)$$

- Class with the highest log probability score is still most probable.

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Evaluation of Performance with test set

Accuracy = (4627 + 741) / (4627 + 741 + 6 + 200) = 96.30%

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

$$\text{Precision} = tp / (tp + fp)$$

$$\text{Precision} = \frac{\textit{number of correct positive predictions}}{\textit{number of positive predictions}}$$

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

$$\text{Recall} = tp / (tp + fn)$$

$$\text{Recall} = \frac{\textit{number of correct positive predictions}}{\textit{number of positive class documents}}$$

	correct	not correct
selected	tp	fp
not selected	fn	tn

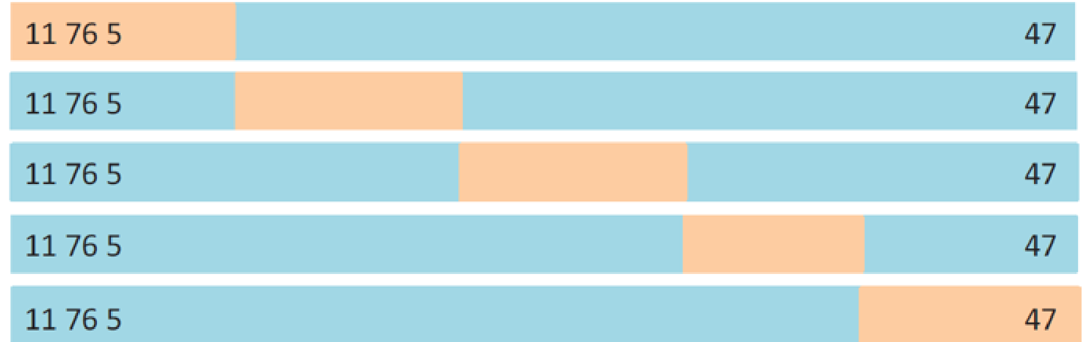
The 2-by-2 Contingency Table

	Truth: spam	Truth: not-spam	Class precision
Predict: spam	741	200	**78.75%**
Predict: not-spam (ham)	6	4627	99.87%
Class recall	**99.20%**	95.86%	

A combined measure: F measure

- A combined measure that assesses a trade-off between Precision and Recall is **F measure** (weighted harmonic mean).
- People usually use balanced **F1-measure**.
  - F1-measure $= \frac{2 * precison * recall}{precision + recall}$

Cross-validation



- Handles **sampling errors** from datasets
  - Especially when dividing **training set** into training and validation sets.
- Dataset (Training set + Validation set) is partitioned into *k* subsets of equal size. (e.g., *k*=5)
- A single subset is retained as the validating data set, and the remaining *k* − *1* subsets are used as training data set.
- The cross-validation process is then repeated *k* times, with each of the *k* subsets used **exactly once** as the validating data.

然后重复交叉验证过程 k 次，其中 k 个子集中的每一个子集都只用作一次验证数据。

Evaluation Practices

- Common approaches of traditional machine learning algorithms
  - **Train Set**: use ***cross-validation*** for hyperparameter tuning, and select good hyperparameters (such as removing stop words and case folding)
  - **Test set**: build a model using Train Set and apply it to Test set.
- Common approaches of deep learning algorithms
  - **Train Set and Validation Set**: build a model with Train Set, and use validation set for hyperparameter tuning, and select good hyperparameters
  - **Test set**: build a model using Train Set + Validation Set (or Train Set only) and apply it to Test set.

Summary of today's lecture

- Learned text-processing for Text Mining.
  - Text normalization: tokenization, stemming, and lemmatization
  - POS tagging and phrase recognition
- Learned a traditional machine learning algorithm
  - Supervised learning classification
  - Naïve Bayesian Classifier
  - Evaluation method: precision, recall, and F1-score

Referenced Materials

- Speech and Language Processing, Dan Jurafsky and James H. Martin, https://web.stanford.edu/~jurafsky/slp3/
- Fundamentals of Predictive Text Mining, Sholom M. Weiss, Nitin Indurkhya, and Tong Zhang, Springer.
  - Chapters 2 & 3