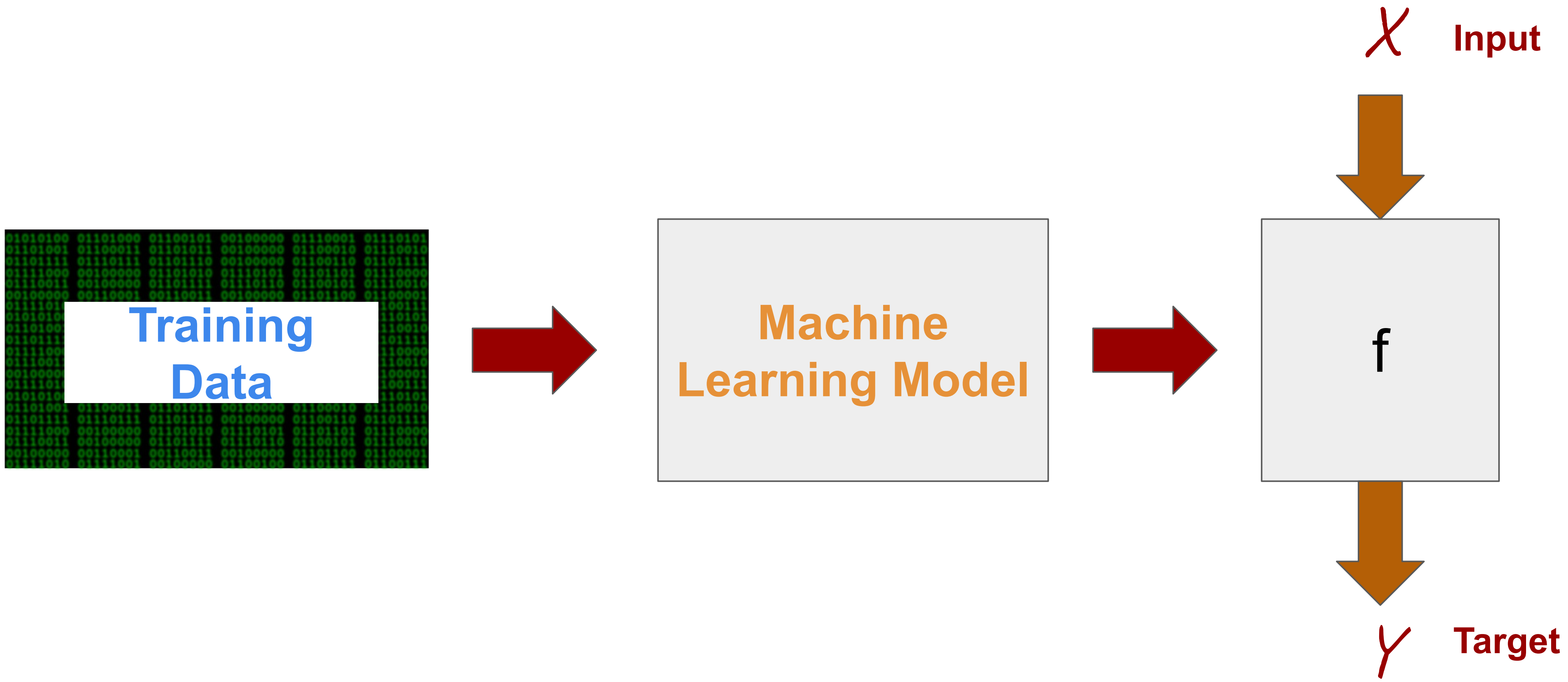
Text Preprocessing I

From textual information to numerical vector

A Simple Text Mining Case

Framework (supervised)



Our Task

- Example task: predict y, whether a string x is an email address
  - x: “rui.zhao@ntu.edu.sg” **y:1**
  - x: “ntuwkw” **y:0**
  - x: “@trump” **y:0**

- How do you address the problem?

Feature Extraction

- **Question**: what properties of x **might be** relevant for predicting y?

- **Feature extractor**: Given input x, output a set of (**feature name**, **feature value**) pairs.

“ntu@gmail.com”

feature extractor

from top to down

Length > 10	**1**
Length < 50	**1**
contain “@”	**1**
endwith “com”	**1**
endwith “sg”	**0**
length between @ and .	**5**
fraction of alpha	**0.85**

Feature Vector notation

- **Mathematically, feature vector does not need feature names:**

Length > 10	1
Length < 50	1
contain “@”	1
endwith “com”	1
endwith “sg”	0
length between @ and .	5
fraction of alpha	0.85

feature
vector
space

Weight Vector notation

- **Weight vector: for each feature j, have a specified parameter representing contribution of feature to prediction**

Length > 10	-1.2
Length < 50	1.4
contain “@”	2.2
endwith “com”	0.6
endwith “sg”	0.5
length between @ and .	0.3
fraction of alpha	0.6

Linear Model

- **Linear combine the features by the weight:**
    - **weighted combination of features**

$$\mathbf{w} \cdot \phi(x) = \sum_{j=1}^{d} w_j \phi(x)_j$$

output: -1.2*(1) + 1.4*(1) + 2.2*(1) + 0.6*(1) + 0.5*(0) + 0.3*(5) + 0.6*(0.85)

Linear Model

- **Weight vector** $\mathbf{w} \in R^d$

- **Feature vector** $\phi(x) \in R^d$

- **For binary classification:**

$$f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x)) = \begin{cases} +1 & \text{if } \mathbf{w} \cdot \phi(x) > 0 \\ -1 & \text{if } \mathbf{w} \cdot \phi(x) < 0 \\ ? & \text{if } \mathbf{w} \cdot \phi(x) = 0 \end{cases}$$

How do we learn model parameters

- **From Data**
- **Define a loss function and then optimize**

Introduction to Text Preprocessing

From Text to Numerical Features

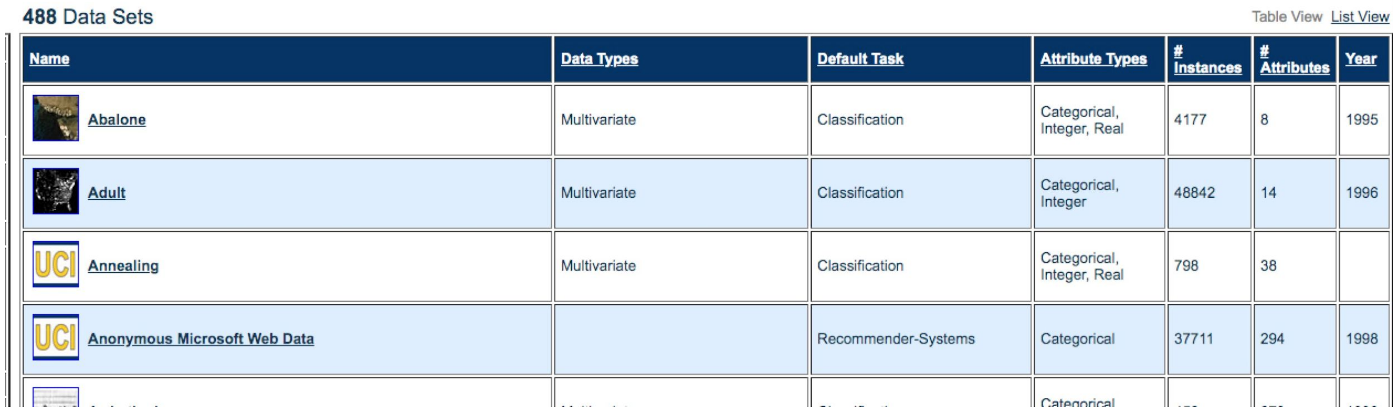
- To mine text, we first need to process it into a form that data mining procedures can use.
- First of all, we have to determine features (think it as the columns of the spreadsheet).
- Some useful features are easy to obtain.
  - the occurence of words
- Some semantic information are much more difficult.
  - The grammatical function of a word in a sentence such as subject, object, et.

Collecting Documents

- The first step in text mining is to collect the data (i.e., the relevant documents).
- In some applications, need to have a data collection process.
  - For a Web application, deploy a software tool such as a Web Crawler that collects the documents.
  - In another application, an email audit application may log all incoming and outgoing messages at a mail server for a period of time.
- For research and development of text-mining techniques, more generic data may be necessary, usually called a corpus
  - the collection of Reuters news stories, such as the Reuters 21578 corpus and RCV1 (Reuters Corpus Volume 1; about 810, 000 Reuters, English Language News stories; tagged with topics).
  - a corpus from the Gutenberg Project, a very large collection of literary and other texts put into machine-readable form as the material comes out of copyright.
  - The Linguistic Data Consortium (LDC) provides various data.

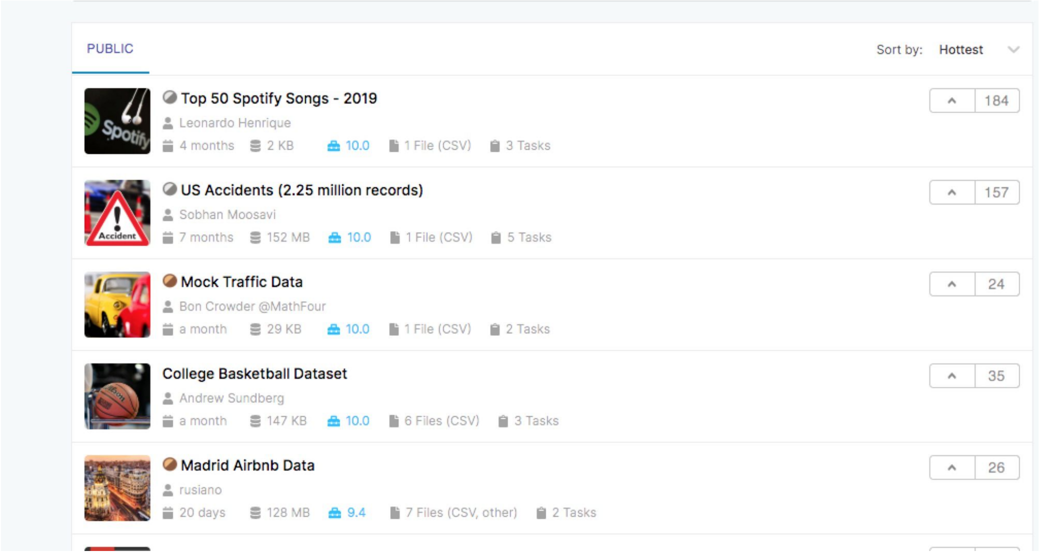
Collecting Documents

- For research and development of text-mining techniques, more generic data may be necessary, usually called a **corpus**.
  - **The UC Irvine Machine Learning Repository** currently maintain 468 datasets (e.g., amazon reviews, email spam and sentiment-labelled sentences) as a service to machine learning community.
  - **Kaggle** (data mining competition) also provides various data sets.



488 Data Sets   Table View List View

Name	Data Types	Default Task	Attribute Types	# Instances	# Attributes	Year
Abalone	Multivariate	Classification	Categorical, Integer, Real	4177	8	1995
Adult	Multivariate	Classification	Categorical, Integer	48842	14	1996
Annealing	Multivariate	Classification	Categorical, Integer, Real	798	38	
Anonymous Microsoft Web Data		Recommender-Systems	Categorical	37711	294	1998
			Categorical			



Text Normalization

Text Normalization

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

Tokenization

Tokenization

- Breaks the stream of characters into **words** or **tokens**.
  - Trivial for a person familiar with the language structure.
- A computer program, though, being linguistically challenged, would find the task more complicated.
- The reason is that certain characters are sometimes **token delimiters** and sometimes not, depending on the application.
- The characters space, tab, and newline are always delimiters are not counted as tokens, often collectively called **white space**.
- The characters **( ) <> ! ? “** are always delimiters and may also be tokens.

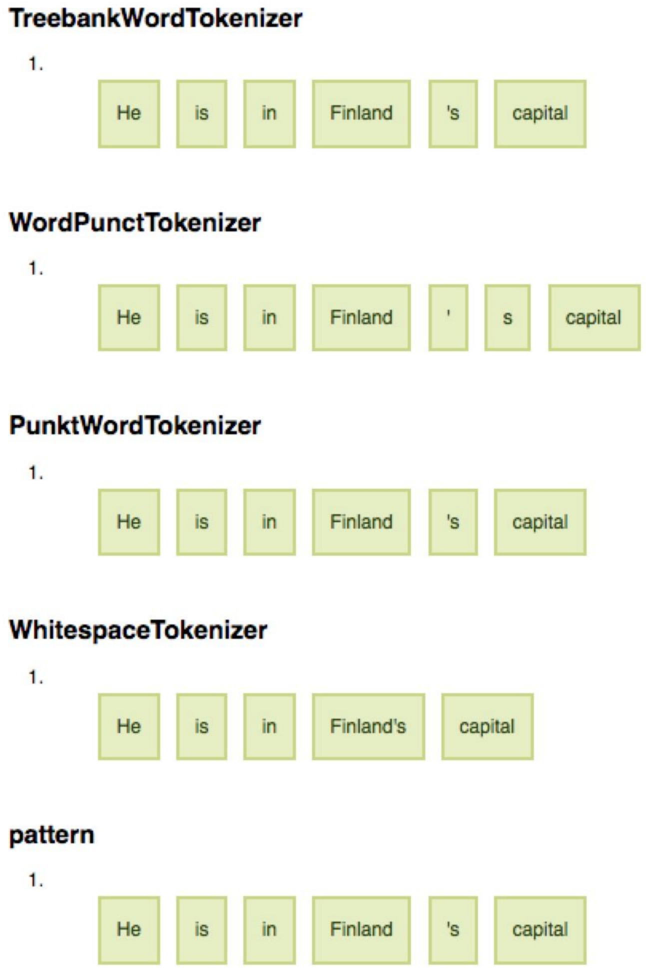
Tokenization

- 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

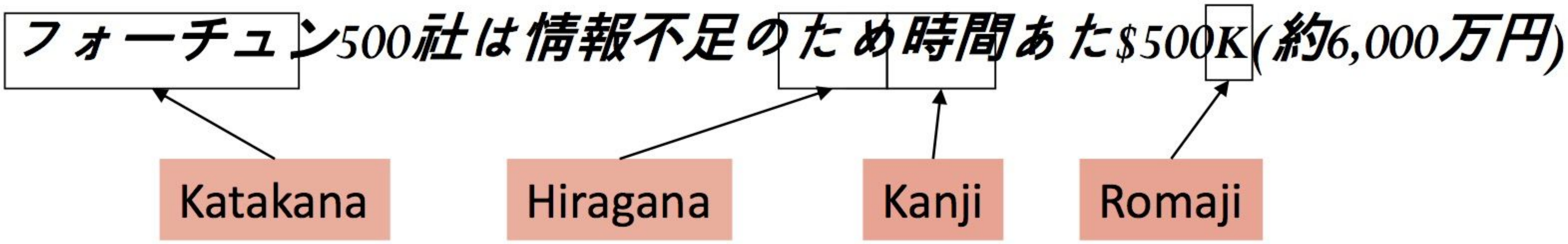
Raw Text	Tokenized Text
Finland's capital	Finland Finlands Finland's ?
What're, I'm, isn't	what are, I am, is not ?
Hewlett-Packard	Hewlett Packard ?
state-of-the-art	state of the art ?
San Francisco	one token or two?

- Online Word Tokenization with python NLTK
  - http://text-processing.com/demo/tokenize/
  - E.g., "He is in Finland's capital"



Tokenization: language issues

- Chinese and Japanese have no spaces between words:
  - 孙燕姿现在居住在新加坡东南部
  - 孙燕姿 现在 居住 在 新加坡 东南部
  - Stefanie Sun now lives in Singapore southeastern
- Further complicated in Japanese, with multiple alphabets intermingled.



Word Tokenization in Chinese

- Also called **Word Segmentation**
- Chinese words are composed of characters.
  - average length is 2.4 char. long.
- Standard baseline segmentation algorithm
  - **Maximum Matching** (also called Greedy)

Maximum Matching Word Segmentation Algorithm

- Given a wordlist of Chinese (i.e. dictionary), 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

  - 孙燕姿现在居住在新加坡东南部
  - 孙燕姿 现在 居住 在 新加坡 东南部
  - Stefanie Sun now lives in Singapore southeastern

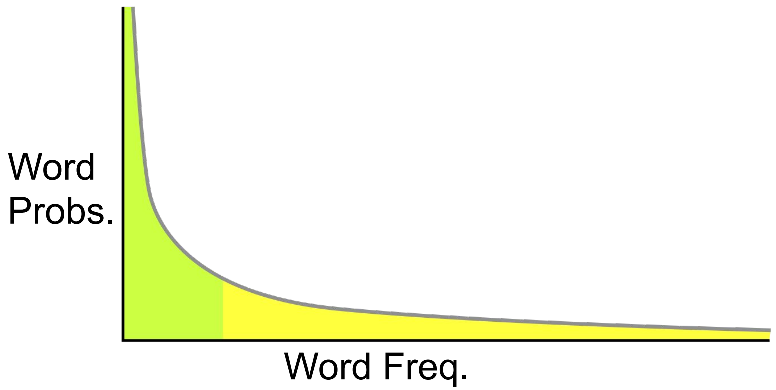
Max-match segmentation illustration

- Thecatinthehat — the cat in the hat
- Thetabledownthere — the table down there / theta bled own there

- Doesn't generally work in English!
- But works well in Chinese
  - 孙燕姿现在居住在新加坡东南部
  - 孙燕姿 现在 居住 在 新加坡 东南部
  - Stefanie Sun now lives in Singapore southeastern
- Moder probabilistic segmentation algorithms even better.
  - E.g., *"the table"* has a higher chance than *"theta bled"*.

Words Properties

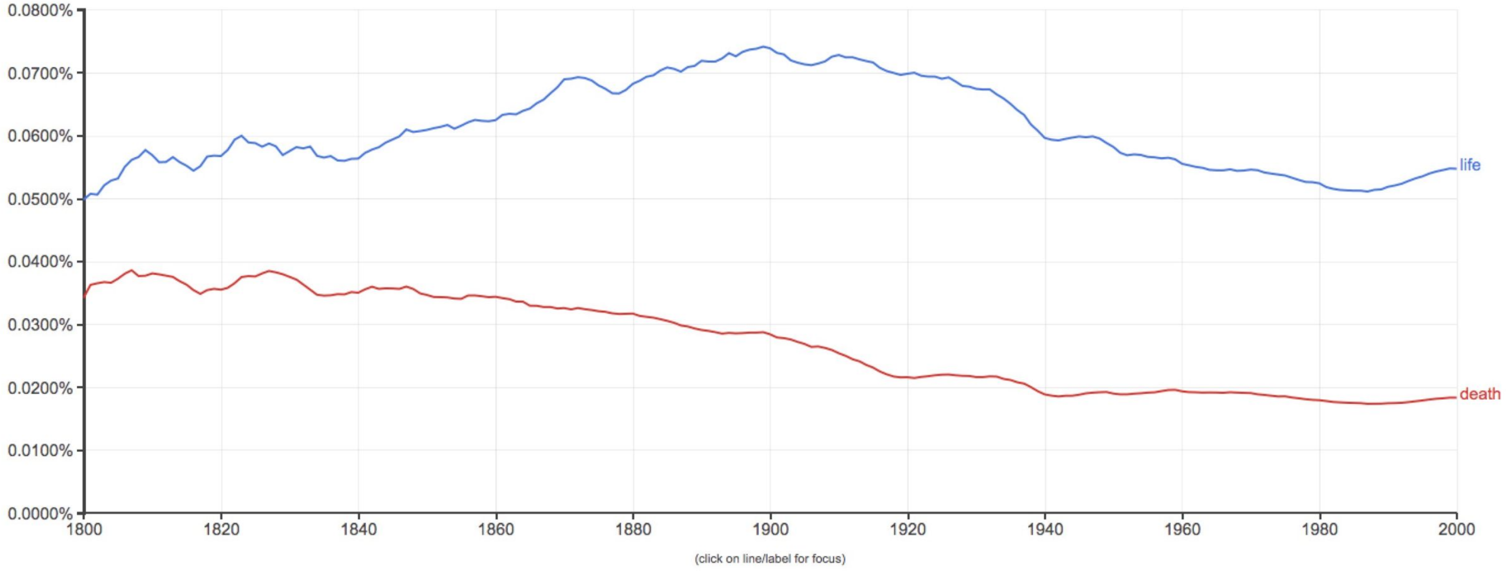
- Relations among word surface forms and their senses:
  - **Homonymy**: same form, but different meaning
    - E.g., bank: river bank and financial institution
  - **Polysemy**: same form, related meaning
    - E.g., man: the human species, male of the human species, and adult males of the human species.
  - **Synonymy**: different form, same meaning
    - E.g., singer and vocalist
- Word frequencies in texts have **power law distribution**:
  - …small number of very frequent words
  - ...big number of low frequent words
  - Also called Zipf's Law



How many words?

- N= number of tokens
- V= vocabulary=set of types

|v| is the size of the vocabulary



https://books.google.come/ngrams

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

Stop Words

- Stop-words are words that from non-linguistic view do not carry information
  - They have mainly functional role.
  - Usually we remove them to help the methods to perform better.
- Natural language dependent - examples:
  - English: A, ABOUT, ABOVE, ACROSS, AFTER, FROM, AGAIN,.....
  - Chinese: 的，一，不，在，有，。。。。

https://www.ranks.nl/stopwords

Stop Words

- Example Stop words
    - Information System Asia Web - provides research, IS-related commercial materials, interaction, **and even** research sponsorship **by** interested corporations **with a** focus **on** Asia Pacific region.
    - Survey **of** Information Retrieval - guide **to** IR, **with an** emphasis **on** web-based projects. Includes **a** glossary, **and** pointers **to** interesting papers.

Normalization

Normalization

- Need to **normalize** terms
  - Information Retrieval (IR): indexed text & query terms must have the same form.
    - we want to match ***U.S.A*** and ***USA***
- We define equivalence class of terms
- Alternative: query expansion
  - Enter: ***window*** Search: *window, windows*
  - Enter: ***windows*** Search: *Windows, windows, window*
- Potentially more powerful, but less efficient

Normalization

- Converts each of the tokens to **a standard form**, a process usually referred to as ***stemming*** or ***lemmatization***.
- Whether or not this step is necessary is application-dependent.
- 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.

Case Folding

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case, such as Car, CAR -> car
  - Possible exception: upper case in mid-sentence?
    - E.g.:
      - ***General Motors vs. general motors***
      - ***Fed vs. fed***
        - Fed: Federal Reserve
      - ***SAIL vs. sail***
        - SAIL: Stanford Artificial Intelligence Language, etc
- 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*."

Lemmatization - Stemming to a Root

- Converts to a **root form** with no inflectional or derivational prefixes and suffixes.
  - **Inflectional suffixes** are endings such as “-ed”, “-ing”, “s”, etc.
    - Create different forms of the same word (different grammatical forms)
  - **Derivational suffixes** are endings such as “-ism”, “-ful”, “-fy”, etc.
    - 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.

Lemmatization

- Additional examples
  - Reduce variant forms to base form
    - am, are, is -> *be*
    - *car, cars, car's, cars'* -> *car*
  - *the boy's cars are different colors* -> *the boy car be different color*
- Lemmatization: have to find correct dictionary headword form (i.e. root or lemma form).
- E.g., Stanford CoreNLP (http://stanfordnlp.github.io/CoreNLP/) supports lemmatization.
  - http://nlp.stanford.edu:8080/corenlp/

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

Some Terms: Morphology

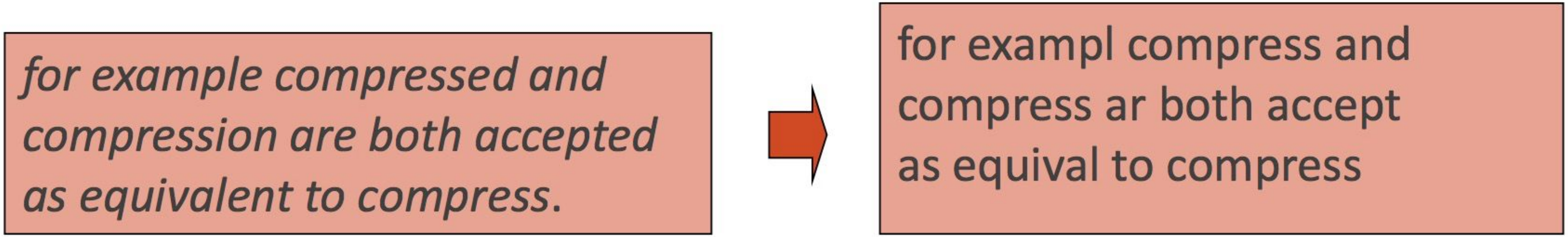
- **Morphemes**:
  - The small meaningful units that make up words.
  - E.g., un-like-ly contains three.
  - Stems: the main part of a word that stays the same when endings are added to it.
    - E.g., writ is the stem of writes, writing, and written.
  - Affixes: Bits and pieces that adhere to stems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

Stemming

- When the normalization is confined to **regularizing grammatical variants** such as singular/plural and present/past, the process is called "**inflectional stemming**."
    - This is called "morphological analysis"
- For a language such as English, with may irregular word forms and non-intuitive spelling, it is more difficult.
    - E.g., sought -> seek
- In English, an algorithm for inflectional stemming must be part rule-based and part dictionary-based.
- Any stemming algorithm for English that operates only on tokens, without more grammatical information such as part-of-speech, will make some mistakes because of ambiguity.
    - For example, is "bored" the adjective as in "he is bored" or is it the past tense of the verb "bore"?
    - He bored her with his stories about military life.

Stemming

- Reduces terms to their stems.
  - E.g., used in information retrieval and text mining applications.
- Stemming is **crude chopping of affixes**.
  - Language dependent
  - e.g., *automates, automatic, automation* all reduced to automat.



Stemming with Python NLTK

Stem Text

Choose stemmer

Porter

Enter text

Stemming is funnier than a bummer says the sushi loving computer scientist

Stemmed Text

Stem is funnier than a bummer say the sushi love comput scientist

Stem Text

Choose stemmer

Lancaster

Enter text

Stemming is funnier than a bummer says the sushi loving computer scientist

Stemmed Text

stem is funny than a bum say the sush lov comput sci

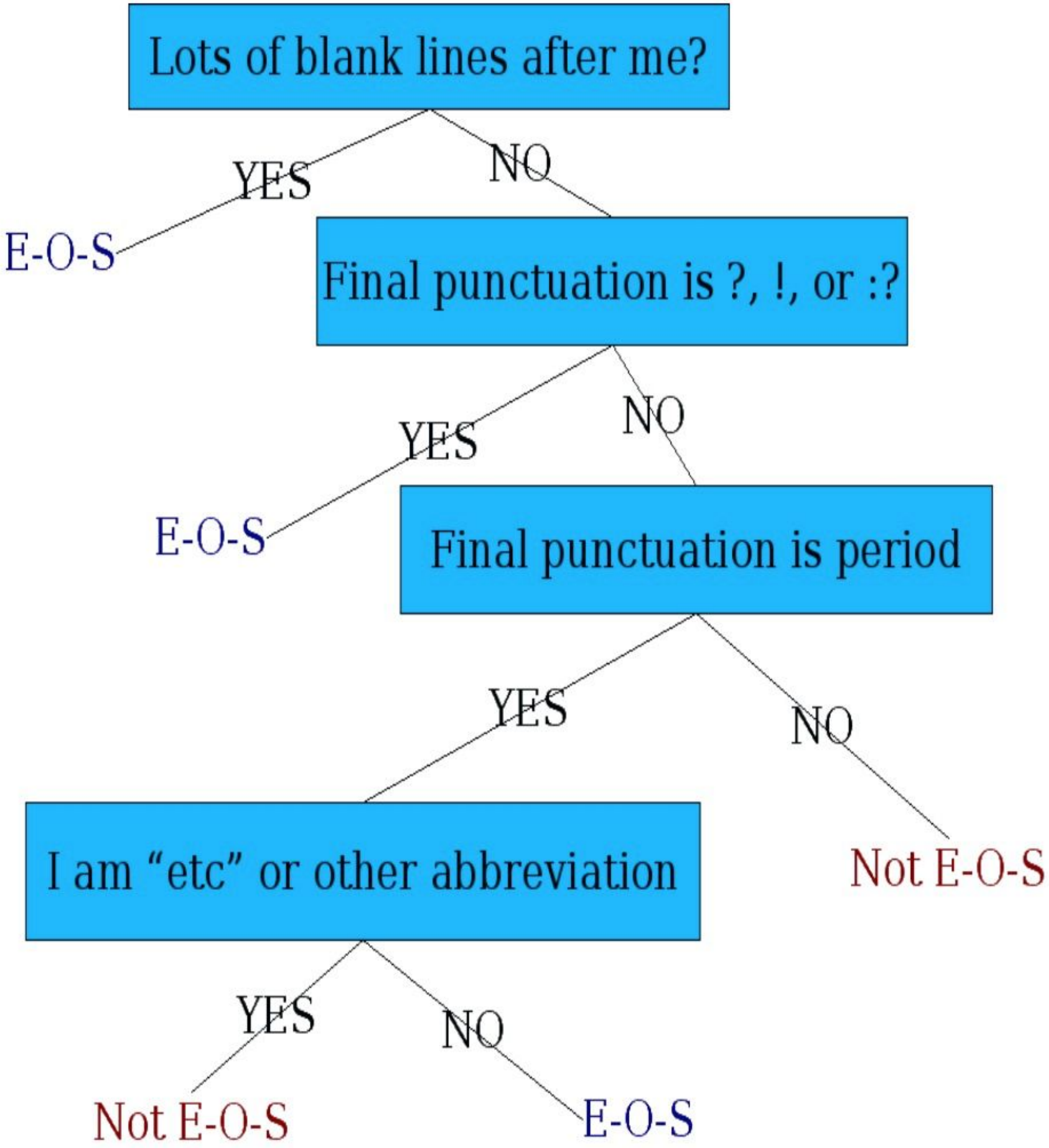
Sentence Boundary Detection

Sentence Boundary Determination

- For more sophisticated linguistic parsing, the algorithms often require **a complete sentence as input**.
  - E.g., sentence-level sentiment analysis
- We shall also see other information extraction algorithms that operate on a sentence at a time.
- 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.

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



Lots of blank lines after me?
YES
NO
E-O-S
Final punctuation is ?, !, or :?
YES
NO
E-O-S
Final punctuation is period
YES
NO
I am “etc” or other abbreviation
Not E-O-S
YES
NO
Not E-O-S
E-O-S

Implementing Decision Trees

- A decision tree is just an if-then-else statement.
    - We can think of the questions in a decision tree.
- The interesting research is choosing the features.
- Setting up the structure is often too hard to do by hand.
    - Hand-building only possible for every simple features, domains.
        - For numeric features, it is too hard to pick each threshold.
    - Instead, structure usually learned by machine learning from a training corpus.
- The features could be exploited by any kind of classifier
    - SVM, Neural Networks, Logistic Regression, etc.

Sentence Boundary Determination

Sentence Detection Algorithm.

- **Hand-written rule**
- Examples of EOS
  - …”. …’.
  - …). …}.
  - …]. …x.Y
  - . $. (
  - . { . [
  - . “ . ‘
- Examples of not EOS
  - Ph.D.
  - www.google.com
  - i.e.

Input: a text with periods
Output: same text with End-of-Sentence (EOS) periods identified

Overall Strategy:
1. Replace all identifiable non-EOS periods with another character
2. Apply rules to all the periods in text and mark EOS periods
3. Retransform the characters in step 1 to non-EOS periods
4. Now the text has all EOS periods clearly identified

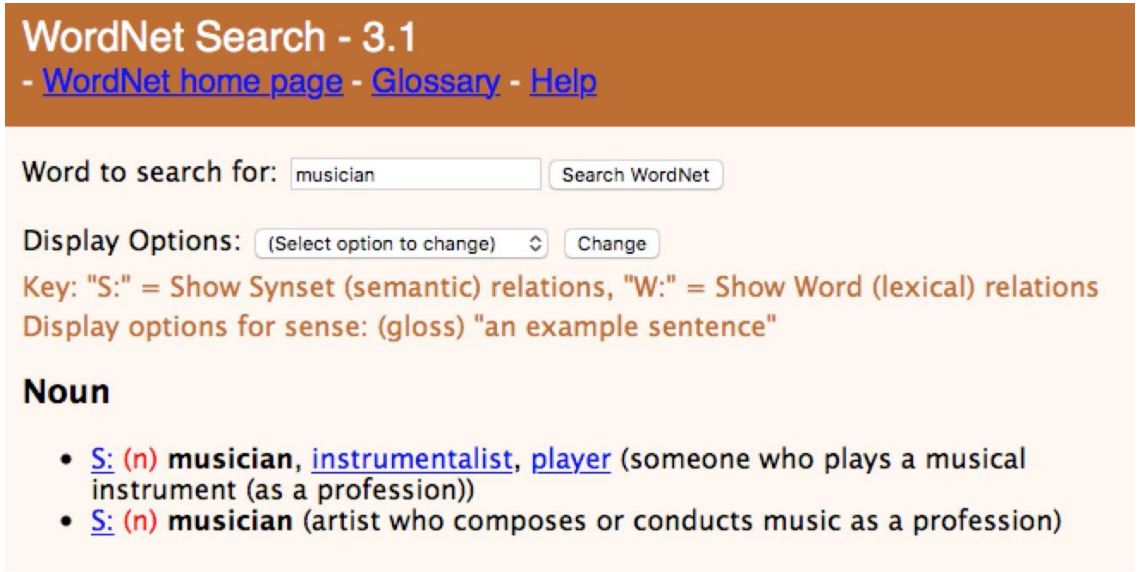
Rules:
All ? ! are EOS
If " or ’ appears before period, it is EOS
If the following character is not white space, it is not EOS
If) }] before period, it is EOS
If the token to which the period is attached is capitalized
  and is < 5 characters and the next token begins uppercase,
  it is not EOS
If the token to which the period is attached has other periods,
  it is not EOS
If the token to which the period is attached begins with a lowercase
  letter and the next token following whitespace is uppercase,
  it is EOS
If the token to which the period is attached has < 2 characters,
  it is not EOS
If the next token following whitespace begins with $ ({ [" ′ it is EOS
Otherwise, the period is not EOS

End-of-sentence detection algorithm

WordNet: Linguistic Resources

WordNet - a database of lexical relations

- **WordNet** is the most well developed and widely used lexical database for English
    - It consists from 4 databases (nouns, verbs, adjectives, and adverbs)
    - On-line version: http://wordnetweb.princeton.edu/perl/webwn
- Each database consists of sense entries consisting from a set of synonyms (synsets), e.g.,:
    - musician, instrumentalist, player
    - person, individual, someone
    - life form, organism, being



WordNet - a database of lexical relations

Category	Unique Forms	Number of Senses
Noun	94474	116317
Verb	10319	22066
Adjective	20170	29881
Adverb	4546	5677

WordNet relations

- Each WordNet entry is connected with other entries in a graph through relations.

- S: (n) **breakfast** (the first meal of the day (usually in the morning))
  - *direct hyponym* / *full hyponym*
  - ***direct hypernym*** / *inherited hypernym* / *sister term*
    - S: (n) meal, repast (the food served and eaten at one time)
  - *derivationally related form*

- S: (n) **course** (part of a meal served at one time) *"she prepared a three course meal"*
  - *direct hyponym* / *full hyponym*
  - *direct hypernym* / *inherited hypernym* / *sister term*
  - ***part holonym***
    - S: (n) meal, repast (the food served and eaten at one time)

WordNet relations

- Relations in the database of **nouns**.

Relation	Definition	Example
Hypernym	From concepts to superordinate	breakfast -> meal
Hyponym	From concepts to subtypes	meal -> lunch
Has-Member (member meronym)	From groups to their members	faculty -> professor
Member-Of (member holonym)	From members to their groups	co-pilot -> crew
Has-Part (part meronym)	From wholes to parts	table -> leg
Part-Of (part holonym)	From parts to wholes	course -> meal
Antonym	Opposites	leader -> follower