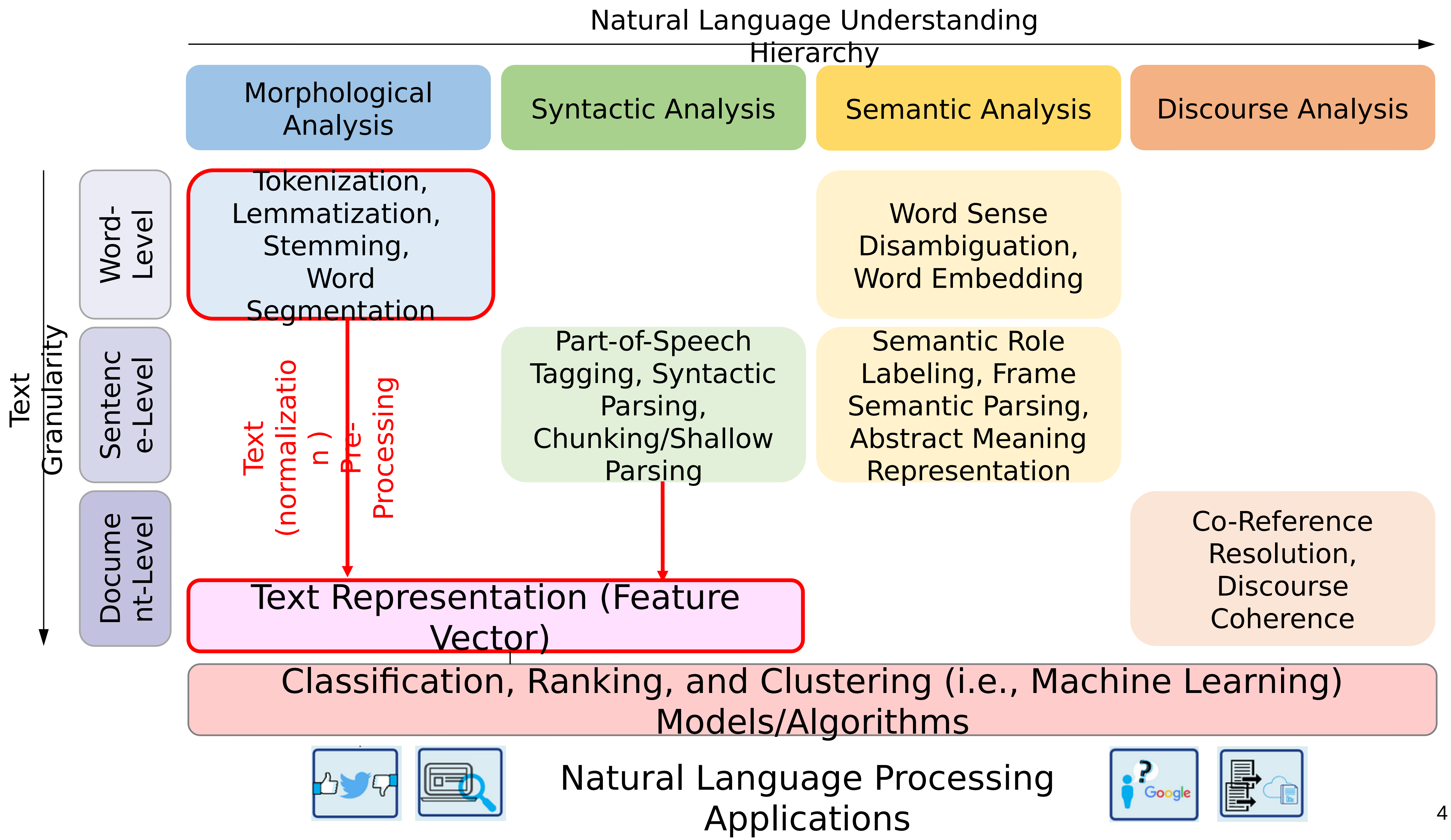
COMP5423 Natural Language Processing

Text Normalization and Representation

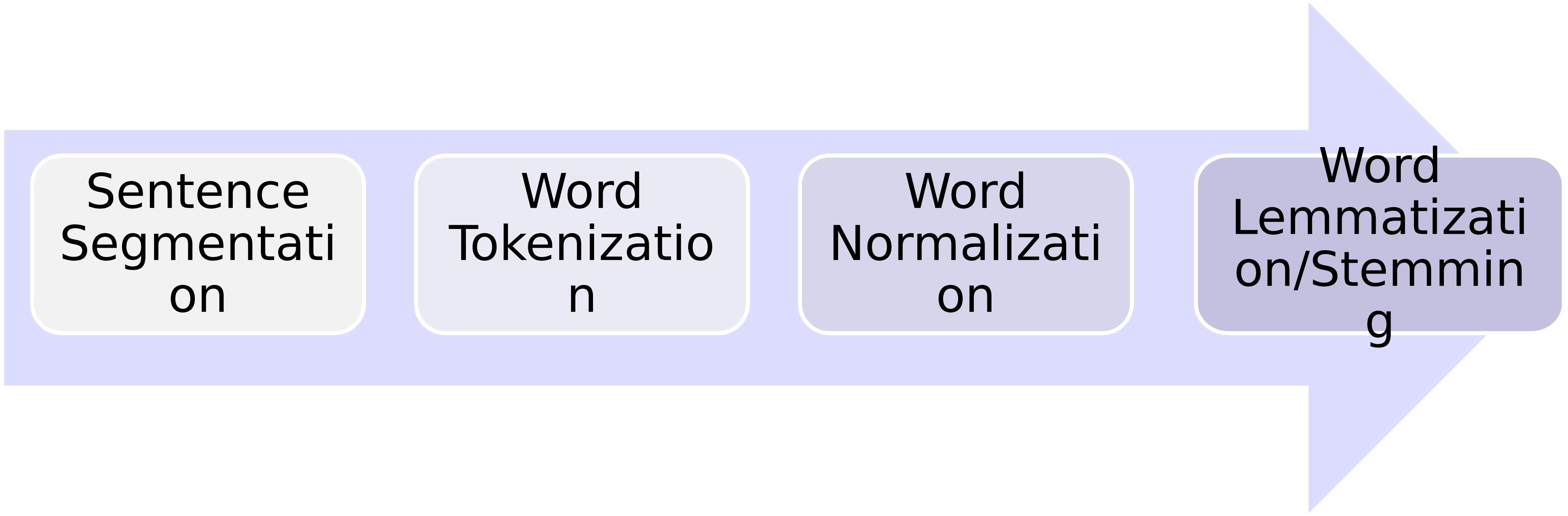
Outline

- **Learning Objectives**
  - Text Normalization (Pre-Processing)
    - Tokenization, Normalization and Segmentation
  - Morphological Analysis
    - Stemming and Lemmatization
  - Text (Document/Sentence) Representation
    - Bag-of-Words (BOW) Representation
    - Vector Representation and Vector Space Model
    - Term Weighting Schemes


Natural Language Understanding Hierarchy
Morphological Analysis
Syntactic Analysis
Semantic Analysis
Discourse Analysis
Text Granularity
Word-Level
Sentence-Level
Document-Level
Tokenization, Lemmatization, Stemming, Word Segmentation
Word Sense Disambiguation, Word Embedding
Text (normalization) Pre-Processing
Part-of-Speech Tagging, Syntactic Parsing, Chunking/Shallow Parsing
Semantic Role Labeling, Frame Semantic Parsing, Abstract Meaning Representation
Co-Reference Resolution, Discourse Coherence
Text Representation (Feature Vector)
Classification, Ranking, and Clustering (i.e., Machine Learning) Models/Algorithms
Natural Language Processing Applications
Google

Text Normalization

- **Text Pre-Processing (Normalization)**

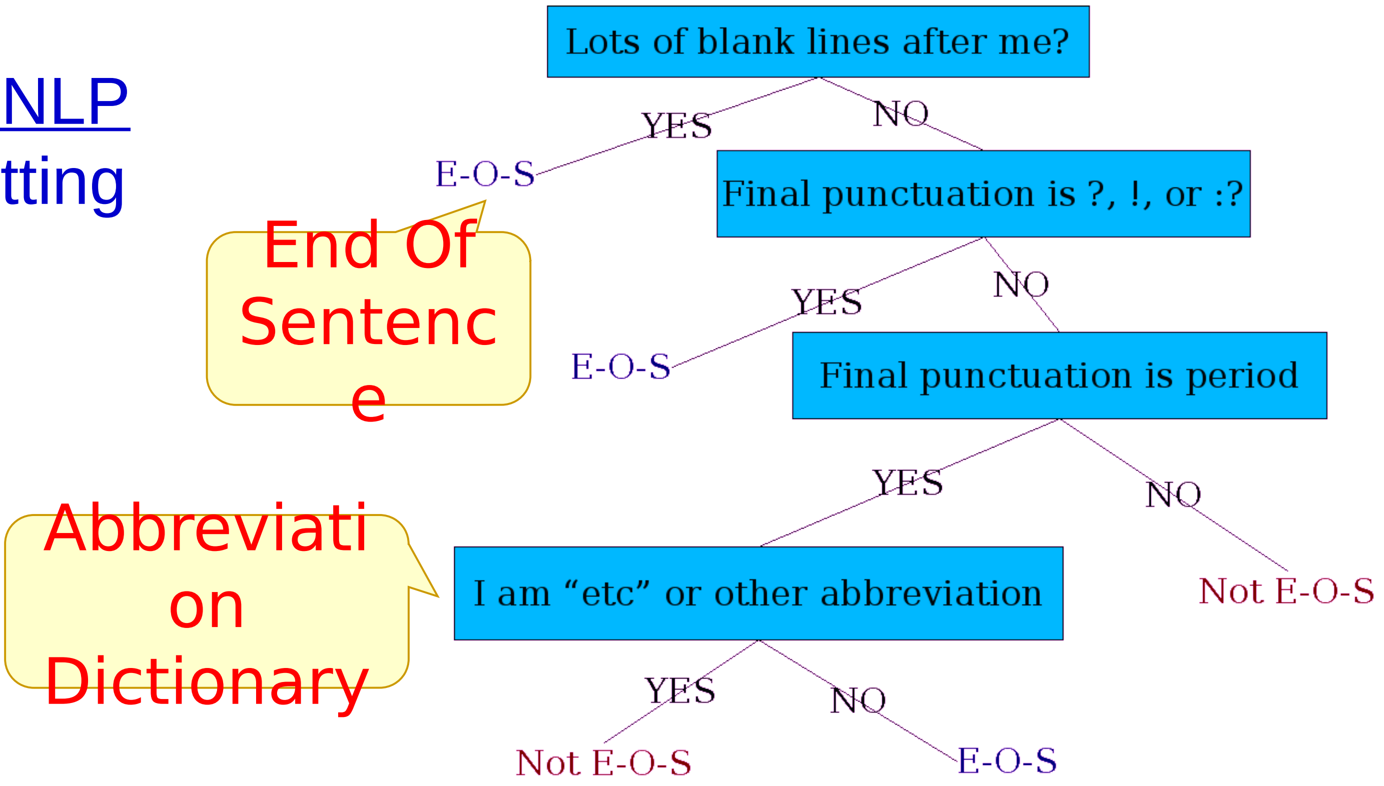


Text Normalization

- **Sentence Segmentation**
  - Sentence segmentation is an important first step in text pre-processing.
  - The most useful cues for segmenting a text into sentences are punctuations, like periods ".", question marks "?", and exclamation points "!".
  - Ambiguity with Periods
    - Sentence Boundary
    - Abbreviation (e.g., Mr. or Inc.)
    - Number (e.g., 6.66% or 8.8)
  - Disambiguation is a binary classification task.

Text Normalization

- **Sentence Segmentation**
  - Rule-based Classifiers
    - In the Stanford CoreNLP toolkit, sentence splitting is rule-based.

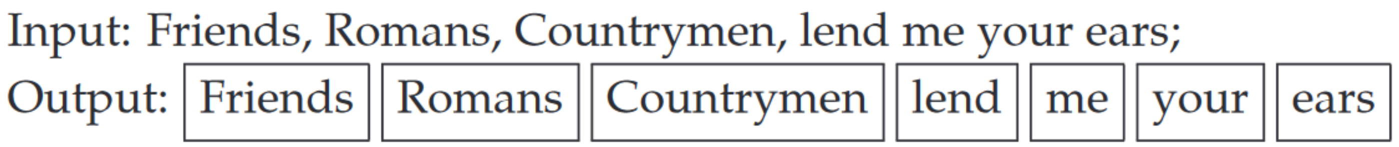

Lots of blank lines after me?
YES
E-O-S
NO
Final punctuation is ?, !, or :?
YES
E-O-S
NO
Final punctuation is period
YES
NO
Not E-O-S
I am "etc" or other abbreviation
YES
Not E-O-S
NO
E-O-S
End Of Sentence
Abbreviation Dictionary

Text Normalization

- **Sentence Segmentation**
  - Learning-based Approaches
    - Classifiers: Decision Tree, Naïve Bayes, SVM, Logistic Regression, etc.
    - Features: Length of the word with “.”, Case of the word with “.”, Probability of the word with “.” that occurs at EOS, etc.

Text Normalization

- **Word Tokenization**
  - Given a sequence of characters, tokenization is the task of chopping it up into pieces, called word tokens, perhaps at the same time throwing away certain characters, such as punctuations.


Input: Friends, Romans, Countrymen, lend me your ears;
Output: Friends | Romans | Countrymen | lend | me | your | ears

Word Tokens

Word Types: Distinct Words	Word Tokens: Word Occurrences
If the set of words in the vocabulary is *V*, the number of word types is the vocabulary size *V*.	There are much more tokens, *N*, in a large corpus than *V*.

  - Question: How many unique word types and word tokens are there in the sentence “A good wine is a wine that you like”?

Text Normalization

- **Word Tokenization**
  - Ambiguity in Tokenization

U.S.A.
T.V.
Ph.D.

$45.55
555,500.50
27/01/2021

http://
www.stanford.edu
someone@cs.colorado.
edu

Clitic (附着詞):

We're ☰ We are
☰ We 're

Doesn't ☰ Does not
☰ Does n't

Hyphenated Words

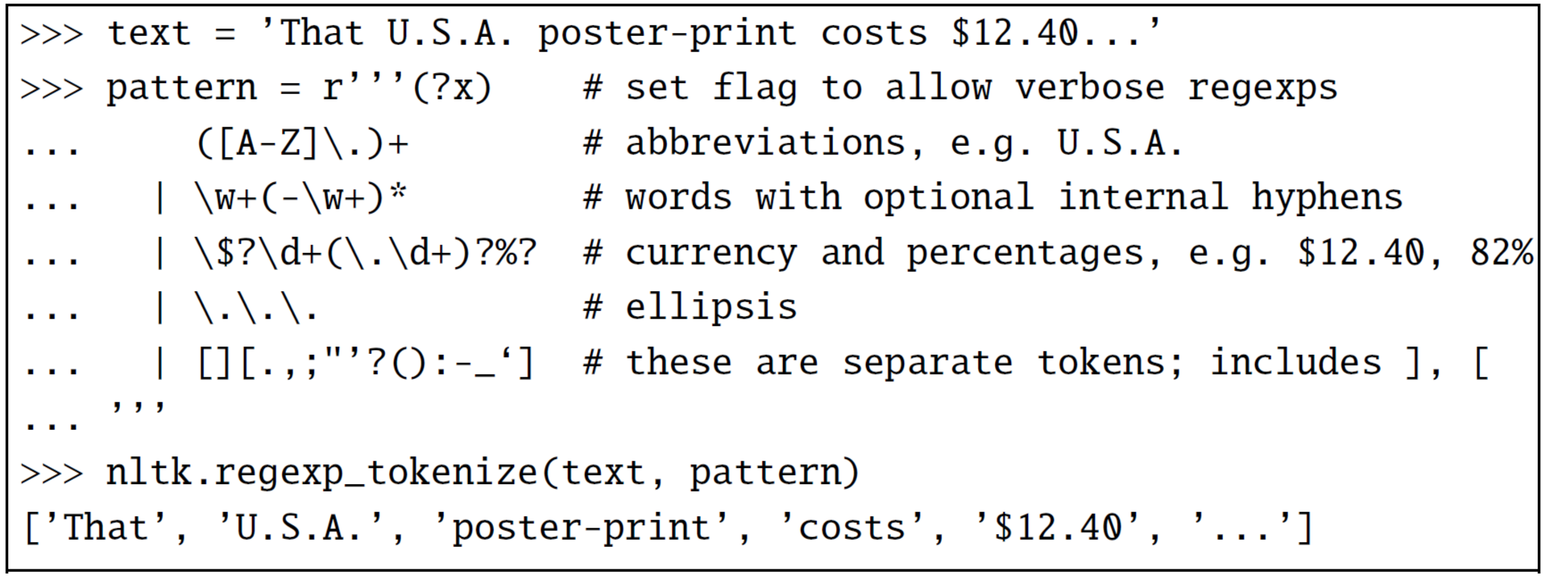
Two-fold

State-of-the-art

Penn Treebank Tokenization Standard

Text Normalization

- **Word Tokenization**
  - Tokenization based on Regular Expressions



```
>>> text = ’That U.S.A. poster-print costs $12.40...’
>>> pattern = r’’’(?x)    # set flag to allow verbose regexps
...     ([A-Z]\.)+        # abbreviations, e.g. U.S.A.
...   | \w+(-\w+)*        # words with optional internal hyphens
...   | \$?\d+(\.\d+)?%?  # currency and percentages, e.g. $12.40, 82%
...   | \.\.\.            # ellipsis
...   | [][.,;"’?():-_‘]  # these are separate tokens; includes ], [
... ’’’
>>> nltk.regexp_tokenize(text, pattern)
[’That’, ’U.S.A.’, ’poster-print’, ’costs’, ’$12.40’, ’...’]
```

An example of a basic regular expression that can be used to tokenize with the NLTK.regexp tokenize function (NLTK is a Python-based Natural Language Toolkit)

Text Normalization

- **Word Tokenization**
  - Regular Expressions:
    Chapter 2.1 in "Speech and Language Processing"
  - Word Tokenization in Chinese = Chinese Word Segmentation
  - Sub-word based Tokenization: Modern tokenizers often automatically induce sets of tokens that include tokens smaller than words, e.g., using Byte-Pair Encoding (BPE), from corpus.
    - Neural Machine Translation of Rare Words with Subword Units, ACL'2016.

Text Normalization

- **Word Normalization**
  - Word normalization is the task of putting words/tokens in a standard format, choosing a single normal form for words with multiple forms.

colour → color

U.S.A. → USA, or T.V. → TV

3 December 2014 → 03/12/2014

Cat → cat

Acronym（首字母縮寫詞）Normalization

Case-Folding

Time Normalization

Text Normalization

- **Tokenization and Normalization Tools**
  - Natural Language Toolkit (NLTK)
    - Book Review: Natural Language Processing with Python, Computational Linguistics, 2010.
  - Stanford Tokenizer
    - The Stanford CoreNLP Natural Language Processing Toolkit, ACL'2014.
  - Specialized Tokenizers for Sentiment

Text Normalization

- **Word Lemmatization and Stemming**
  - Morphological analysis is to study how words are formed, i.e., built up from smaller meaning-bearing units called morphemes.
    - Two broad classes of morphemes can be distinguished, i.e., stems and affixes.
    - The stem is the “main” morpheme of the word, supplying the main meaning, while the affixes add “additional” meaning of various kind.
  - Stemming and lemmatization belong to morphological analysis.

Text Normalization

- **Word Lemmatization and Stemming**

English mainly uses prefixes and suffixes to express inflectional and derivational morphology.

Inflectional morphology is relatively simple and includes person and number agreement (-s) and tense markings (-ed and -ing).

Derivational morphology is more complex and includes suffixes, like -tion, -ness, -able, as well as prefixes like co-, and re-).

Such an affix usually applies to words of one part-of-speech category and changes them into words of another such category.

adjective-to-noun: -*ness* (slow → slowness)

adjective-to-adverb: -*ly* (personal → personally)

noun-to-adjective: -*al* (recreation → recreational)

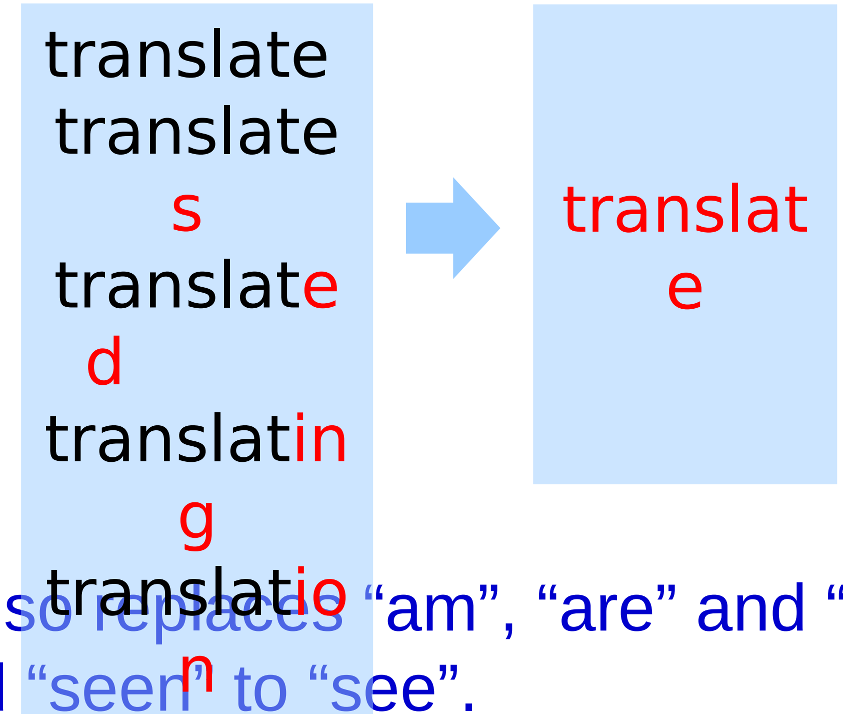
noun-to-verb: -*fy* (glory → glorify)

verb-to-adjective: -*able* (drink → drinkable)

verb-to-noun (agent): -*er* (write → writer)

Text Normalization

- **Word Lemmatization and Stemming**
  - Lemmatization is to reduce inflectional forms (surface forms) and sometimes derivationally related forms of a word to a common lemmatized (base, root) form or lemma.



A lemma is the canonical form or dictionary form of a word.

- It also replaces "am", "are" and "is" with "be", and changes "saw" and "seen" to "see".

Text Normalization

- **Word Lemmatization and Stemming**
  - Stemming is a naive version of morphological analysis which simply strip off affixes.

translat~~e~~
translat~~es~~
translat~~ed~~
translat~~ing~~
translat~~ion~~

translat

  - It is preferable for applications like information retrieval in which exact morphological structure is not needed.

Text Normalization

- **Word Lemmatization and Stemming**
  - On-Line English Stemming Demos
    - Javascript Porter Stemmer Online
    - Stemming and Lemmatization with Python NLTK

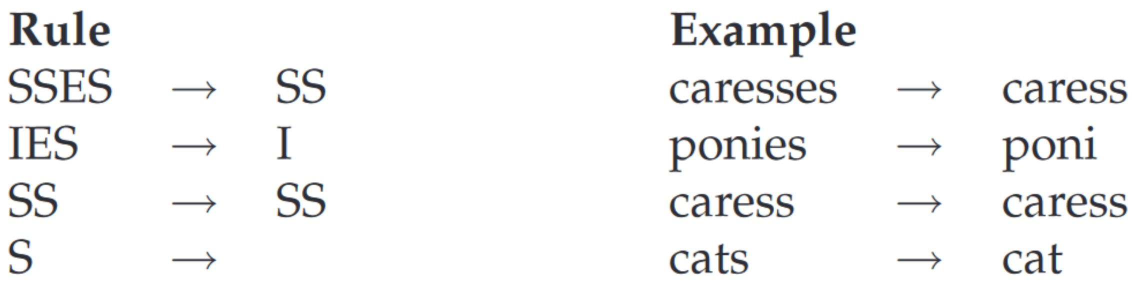
This was not the map we found in Billy Bones's chest, but an accurate copy, complete in all things-names and heights and soundings-with the single exception of the red crosses and the written notes.

Thi wa not the map we found in Billi Bone ' s chest but an accur copi, complet in all thing - name and height and sound - with the singl except of the red cross and the written note.

Text Normalization

- **Word Lemmatization and Stemming**
  - Examples of Porter Stemming Rules

The algorithm is based on series of rules run in series, as a cascade, in which the output of each pass is fed as input to the next pas



Rule			Example		
SSES	→	SS	caresses	→	caress
IES	→	I	ponies	→	poni
SS	→	SS	caress	→	caress
S	→		cats	→	cat

- Porter Stemming Algorithm
- An Algorithm for Suffix Stripping (M.F. Porter, 1980)

Text Normalization

- **Morphological Analysis for Chinese**
  - Chinese is not a language with morphological variations.

English:

The machine translation system is a practical tool that can help us translate one language into another.

Chinese:

機器翻譯系統是一種能夠幫助我們將一種語言翻譯成另外一種語言的實用工具。

English:

They are boys. She is a girl.

Chinese:

他們是男孩。她是女孩。

Text Normalization

- **Morphological Analysis for Chinese**
  - Chinese Word Tokenization: Word Segmentation
    - For a language like Chinese, where text is written without any spaces between words, we can perform word segmentation using dictionary based, statistical based or machine learning based approaches.

莎拉波娃现在居住在美国东南部的佛罗里达。今年 4 月 9 日，莎拉波娃在美国第一大城市纽约度过了 18 岁生日。生日派对上，莎拉波娃露出了甜美的微笑。

莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达。今年 4 月 9 日，莎拉波娃 在 美国 第一 大 城市 纽约 度过 了 18 岁 生日。生日 派对 上，莎拉波娃 露出 了 甜 美 的 微笑。

Text Normalization

- **Chinese Word Segmentation**
  - Dictionary-based Word Segmentation
    - Basically, dictionary-based segmentation is to match a Chinese character string against the words in a dictionary.
    - The successfully matched string is segmented as a word.
    - Matching Direction: Forward Matching vs. Backward Matching
    - Matching Length Priority: Maximum Matching vs. Minimum Matching

Text Normalization

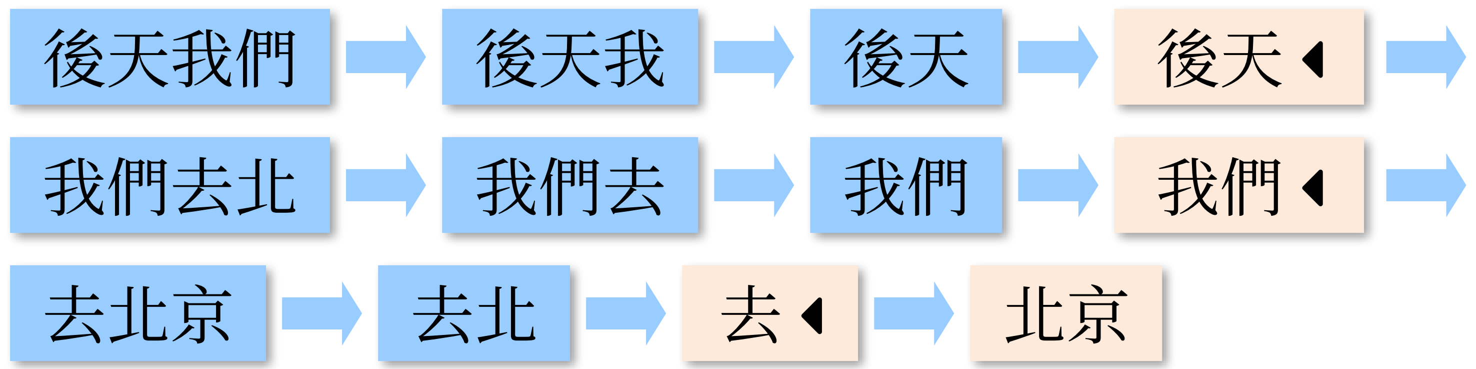
- **Chinese Word Segmentation**
  - Dictionary-based Word Segmentation with Forward Maximum Matching
    - Let *MaxLen* represent the maximum word length.
    - (1) From left to right, select a sub-string from input with the length of *MaxLen* and match it with the words in a dictionary.
    - (2) If the matching is successful, then insert a space (or any symbol indicating boundaries) at the end of this sub-string, move forward with *MaxLen* and repeat the matching step.
    - (3) Otherwise, reduce one character from the end of the string, and match again until a word is found.

Text Normalization

- **Chinese Word Segmentation**
  - Dictionary-based Word Segmentation with Forward Maximum Matching
    - (4) If a two-character string still cannot be located in the dictionary, then regard the current single character as a word (i.e., an isolated word, not necessarily a real word), move to the next and repeat the matching step.

Text Normalization

- **Chinese Word Segmentation**
  - Dictionary-based Word Segmentation with Forward Maximum Matching
    - Example: Segment the character string of 後天我們去北京 with the forward maximum matching approach (Let *MaxLen*=4) and a given dictionary { 後天 , 我們 , 去 , 北京 }.

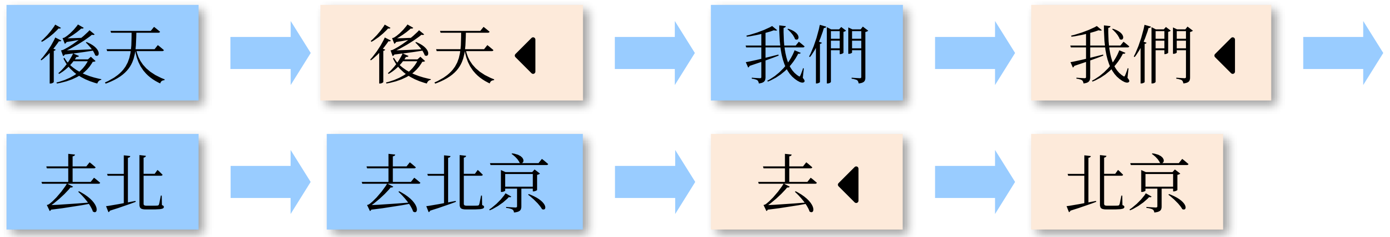


Text Normalization

- **Chinese Word Segmentation**
  - Dictionary-based Word Segmentation with Forward Minimum Matching
    - (1) From left to right, select a sub-string with two characters and match it with the words in a dictionary.
    - (2) If it is matched successfully, move two characters ahead and match again.
    - (3) Otherwise, add one character each time and match again until the word is found.
    - (4) When the length=*MaxLen* and the matching still fails, regard the current character as a word.

Text Normalization

- **Chinese Word Segmentation**
  - Dictionary-based Word Segmentation with Forward Minimum Matching
    - Example: Segment the character string of 後天我們去北京 with the forward minimum matching approach (Let *MaxLen*=4) and a given dictionary { 後天 , 我們 , 去 , 北京 }.



Text Normalization

- **Chinese Word Segmentation**
  - Question 1: Why is the parameter *MaxLen* is ofen set to 4?
  - Question 2: Compare the maximum matching and minimum matching strategies, which one do you believe is more computationally effective, and why?
  - Question 3: What should be noted when using backward matching? Take " 後天我們去北京" as an example.
  - Performance of Dictionary-based Word Segmentation
    - In general, backward matching is more accurate than forward matching. Both can achieve above 95% of accuracy.

Text Normalization

- **Chinese Word Segmentation**
  - On-Line Word Segmentation Demos and Tools
    - CKIP (for Traditional Chinese Only)
    - LTP (for Simplified Chinese)
    - NLPIR (for Simplified Chinese)
    - Examples: 內存在漲價。中國人。他馬上就來。他從馬上摔下來。你們後天再來吧，到家後天就黑了。這種病的病因到目前為止醫學界都不清楚，他的病因我而起。習近平，男，漢族，1953 年 6 月生，陝西富平人。

Text Normalization

- **Chinese Word Segmentation**
  - Ambiguity in Word Segmentation
    - Overlapping Ambiguity (90%): The string ABC may be segmented into AB ◂ C or A ◂ BC, when AB ⊞ W and BC ⊞ W, W is a collection of words in a dictionary.

計算機會下象棋　　　　她的確切地址在這兒

計算機 ◂ 會 ◂ 下 ◂ 象棋	她 ◂ 的 ◂ 確切 ◂ 地址 ◂ 在 ◂ 這兒
計算 ◂ 機會 ◂ 下 ◂ 象棋	她 ◂ 的確 ◂ 切 ◂ 地址 ◂ 在 ◂ 這兒

Text Normalization

- **Chinese Word Segmentation**
  - Ambiguity in Word Segmentation
    - Grouping Ambiguity (10%): For a string of AB, AB is a word, and meanwhile, A and B itself is also an independent word, i.e., AB ⊞ W ◂ A ⊞ W ◂ B ⊞ W.

學生會聽老師的話嗎

學生會 ◂ 聽 ◂ 老師 ◂ 的 ◂ 話 ◂ 嗎

學生 ◂ 會 ◂ 聽 ◂ 老師 ◂ 的 ◂ 話 ◂ 嗎

Text Normalization

- **Sub-Word Segmentation (Sub-Word based Tokenization)**
  - To deal with this unknown word problem, modern tokenizers often automatically induce sets of tokens that include tokens smaller than words, called sub-words, from corpus.

Comparative Adjectives

lower:	low	er
newer:	new	er
faster:	fast	er

+

Superlative Adjectives

lowest:	low	est
+		
newest:	new	est
+		
fastest:	fast	est
+		

Text Normalization

- **Sub-Word Segmentation (Sub-Word based Tokenization)**
  - Byte-Pair Encoding (BPE) Algorithm

It begins with a vocabulary that is just the set of all individual characters.

Corpus: {‘lower</w>’: 2, ‘newest</w>’: 6, ‘widest</w>’: 3}

(Token) Vocabulary: {‘l’, ‘o’, ‘w’, ‘e’, ‘r’, ‘</w>’, ‘n’, ‘s’, ‘t’, ‘i’, ‘d’}

It then examines the corpus, chooses the two symbols that are most frequently adjacent (e.g., ‘e’, ‘s’), adds a new merged symbol ‘es’ to the vocabulary, and replaces every adjacent ’e’ ’s’ in the corpus with the new ‘es’.

Corpus: {‘lower</w>’: 2, ‘new es t</w>’: 6, ‘wid es t</w>’: 3}

Vocabulary: {‘l’, ‘o’, ‘w’, ‘e’, ‘r’, ‘</w>’, ‘n’, ‘t’, ‘i’, ‘d’, ‘es’ }

Text Normalization

- **Sub-Word Segmentation (Sub-Word based Tokenization)**
  - Byte-Pair Encoding (BPE) Algorithm

Corpus: {‘lower</w>’: 2, ‘new est</w>’: 6, ‘wid est</w>’: 3}

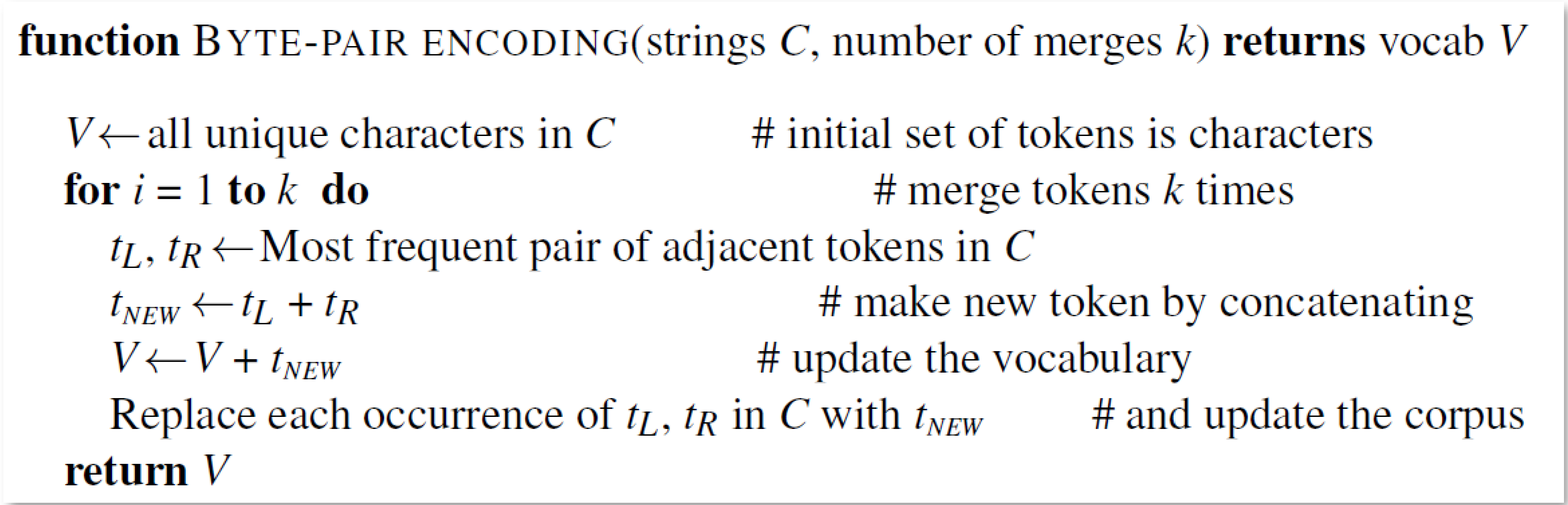
Vocabulary: {‘l’, ‘o’, ‘w’, ‘e’, ‘r’, ‘</w>’, ‘n’, ‘i’, ‘d’, ‘est’ }

......

  - Sub-words can be arbitrary substrings, or they can be meaning-bearing units like the morphemes.

Text Normalization

- **Sub-Word Segmentation (Sub-Word based Tokenization)**
  - Byte-Pair Encoding (BPE) Algorithm



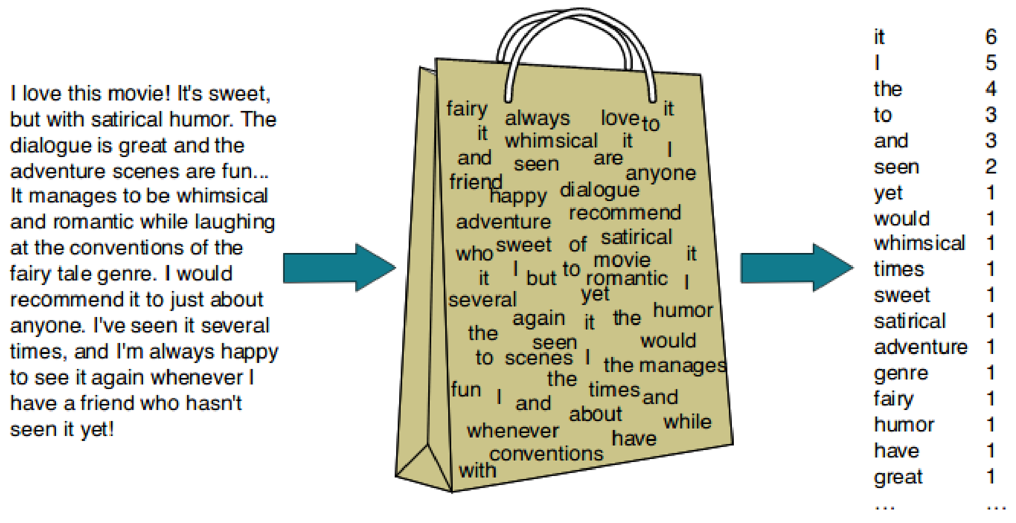
```
function BYTE-PAIR ENCODING(strings $C$, number of merges $k$) returns vocab $V$

    $V \leftarrow$ all unique characters in $C$            # initial set of tokens is characters
    for $i = 1$ to $k$ do                                  # merge tokens $k$ times
        $t_L, t_R \leftarrow$ Most frequent pair of adjacent tokens in $C$
        $t_{NEW} \leftarrow t_L + t_R$                     # make new token by concatenating
        $V \leftarrow V + t_{NEW}$                         # update the vocabulary
        Replace each occurrence of $t_L, t_R$ in $C$ with $t_{NEW}$      # and update the corpus
    return $V$
```

- Neural Machine Translation of Rare Words with Subword Units, ACL’2016.

Text Representation

- **Bag-of-Words (BOW) Representation**
  - A running text (either a document or a sentence) can be simply represented as a set (or a "bag") of words appearing in it.



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
…	…

A Bag of Unique Words

Text Representation

- **Bag-of-Words (BOW) Representation**
  - Representative Words
    - Representative words are those such as “adventure”, “humor” and “sweet”, etc., whose semantics helps in remembering main themes of documents.
  - Non-Representative Words
    - Non-representative words are those such as “of”, “the” and “that” etc.
    - Words, which are too frequent among the documents are not good discriminators.

Text Representation

- **Bag-of-Words (BOW) Representation**
  - Stop Words
    - Such non-representative words are frequently referred to as stop words, and normally filtered out.
  - Classical information retrieval (IR) models consider that each document is represented by a set of representative words, called index terms.

a, about, above, after, again, against, all, am, an, and, any, are, as, at, be, because, been, before, being, below, between, both, but, by, can, did, do, does, doing, don, down, during, each, few, for, from, further, had, has, have, having, he, her, here, hers, herself, him, himself, his, how, i, if, in, into, is, it, its, itself, just, me, more, most, my, myself, no, nor, not, now, of, off, on, once, only, or, other, our, ours, ourselves, out, over, own, same, she, should, so, some, such, than, that, the, their, theirs, them, themselves, then, there, these, they, this, those, through, to, too, under, until, up, very, was, we, were, what, when, where, which, while, who, whom, why, will, with, you, your, yours, yourself, yourselves

NLTK's List of English Stop Words

Text Representation

Original

To be, or not to be: that is the
Whether 'tis nobler in the mind to
suffer
The slings and arrows of outrageous
fortune,
Or to take arms against a sea of
troubles,
And by opposing end them? To die:
to sleep;
end ...

Tokenized and Normalized

to be or not to be that is the
whether tis nobler in the mind to
suffer
the slings and arrows of outrageous
fortune
or to take arms against a sea of
troubles
and by opposing end them to die to
sleep
no more and by a sleep to say we
end

Stemmed

to be or not to be that is the
whether ti nobler in the mind to
suffer
the sling and arrow of outrag
fortun
or to take arm against a sea of
troubl
and by oppos end them to die to
sleep
end ...

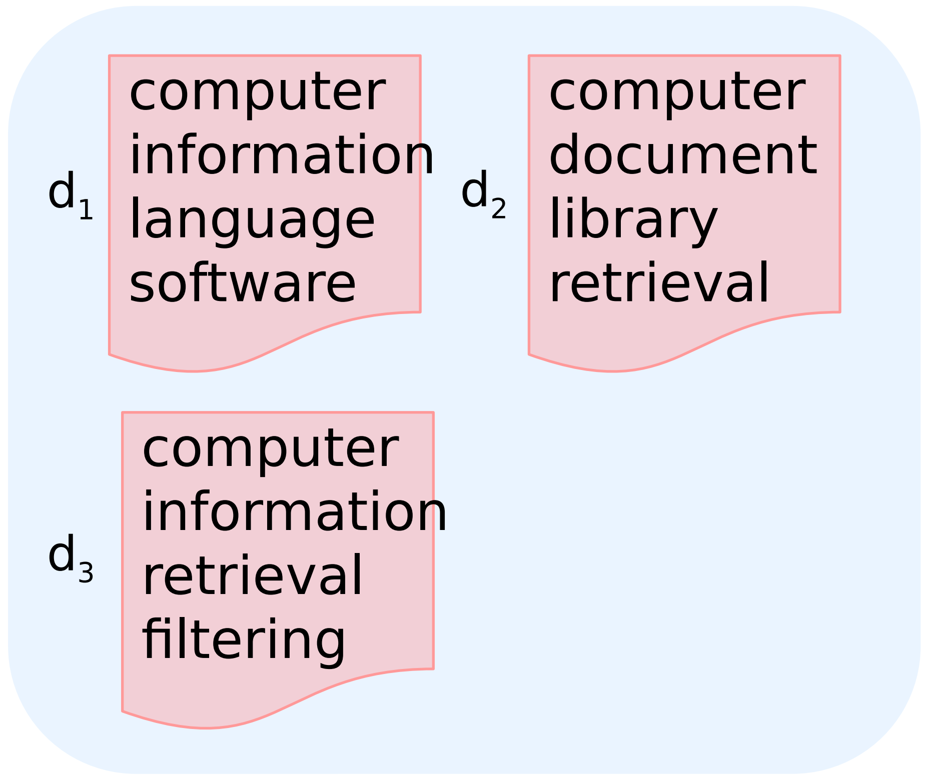
Stop Word Removed

question
ti nobler mind suffer
sling arrow outrag fortun
take arm sea troubl
oppos end die sleep
sleep sai end …

Index Terms

Text Representation

- **Vector Space Model**


d_1
computer information language software
d_2
computer document library retrieval
d_3
computer information retrieval filtering

Dataset

Index terms (Vocabulary)

Documents

	d_1	d_2	d_3
t_1=computer	1	1	1
t_2=document	0	1	0
t_3=filtering	0	0	1
t_4=information	1	0	1
t_5=language	1	0	0
t_6=library	0	1	0
t_7=retrieval	0	1	1
t_8=software	1	0	0

Term-Document Matrix

Each row represents a word in the vocabulary and each column represents a document from some collection.

Text Representation

- **Vector Space Model**

	d_1	d_2	d_3
t_1=computer	1	1	1
t_2=document	0	1	0
t_3=filtering	0	0	1
t_4=information	1	0	1
t_5=language	1	0	0
t_6=library	0	1	0
t_7=retrieval	0	1	1
t_8=software	1	0	0

Vocabulary

Document Vectors

Binary Weights

$$\begin{pmatrix}\text{computer}\\ \text{document}\\ \text{filtering}\\ \text{information}\\ \text{language}\\ \text{library}\\ \text{retrieval}\\ \text{software}\end{pmatrix} \quad d_1=\begin{pmatrix}1\\0\\0\\1\\1\\0\\0\\1\end{pmatrix} \quad d_2=\begin{pmatrix}1\\1\\0\\0\\0\\1\\1\\0\end{pmatrix} \quad d_3=\begin{pmatrix}1\\0\\1\\1\\0\\0\\1\\0\end{pmatrix}$$

- A vector space is a mathematical structure formed by a collection of vectors.
- Question: How many distinct vectors can be represented by an

Text Representation

- **Vector Space Model**
  - A document is represented as a vector of index terms.
    - Each dimension corresponds to an index term in the vocabulary.
    - Thus, if there are *n* index terms in the dictionary, a document is a *n*-dimensional vector.
    - If a term occurs in the document, its value in the vector is non-zero (known as the weight of the term, which can be binary, count, or real-valued).

Text Representation

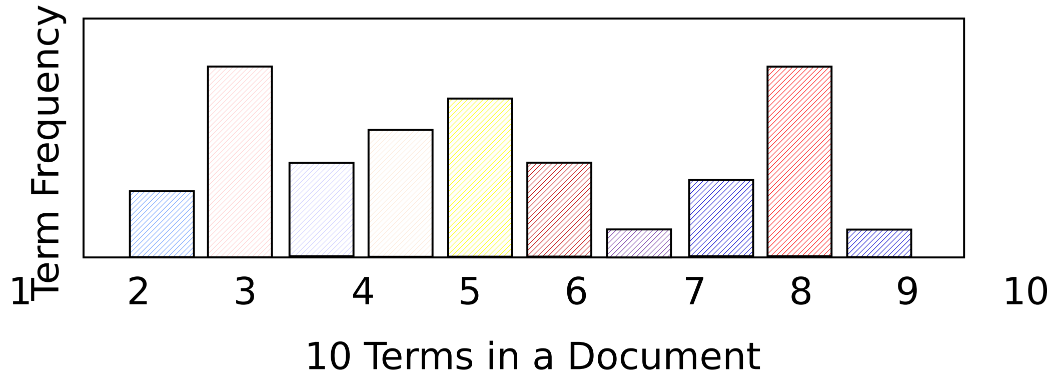
- **Term Weighting Schemes**
  - Given a set of index terms for a document, not all terms are equally useful for describing document contents.
  - The importance of these terms for describing document semantic contents is represented by the weights associated to them.
  - If *t* is an index term, *d* is a document, we use $w_{t,d}$ to represent the weight associated with the pair (*t*, *d*).
  - Binary Weight Scheme

$$w_{t,d} = \begin{cases} 1 \; if \; term \; t \; is \; present \in document \; d \\ 0 \; otherwise \end{cases}$$

Text Representation

- **Term Weighting Schemes**
  - Term Frequency (TF)

Assumption: Terms that occur frequently within a document may reflect its meaning more strongly than terms that occur less frequently and thus should have high weights.



- The term frequency $tf_{t,d}$ of term *t* in document *d* is defined as the number of times *t* occurs in *d* (i.e., the number of the occurrences of term *t* in document *d*).

Text Representation

- **Term Weighting Schemes**
  - Term Frequency (TF)

Frequency of a Term in a Document

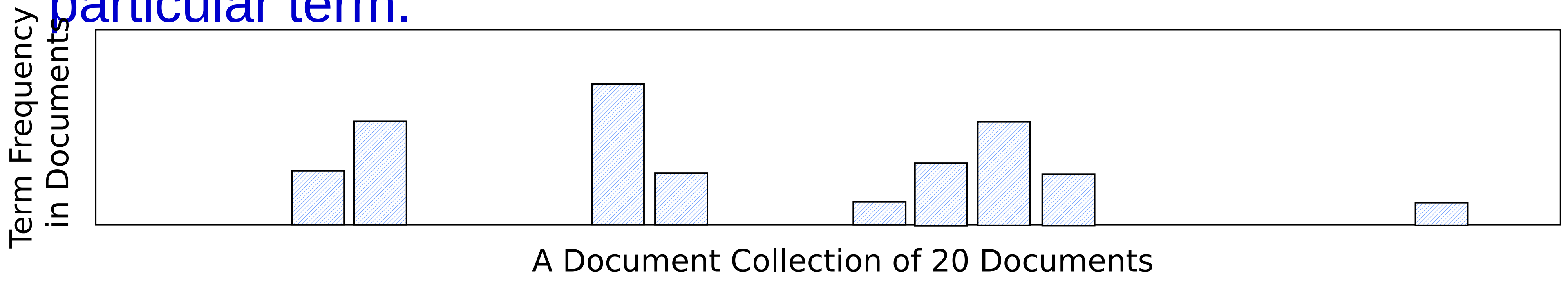
$$\begin{pmatrix} \text{computer} \\ \text{document} \\ \text{filtering} \\ \text{information} \\ \text{language} \\ \text{library} \\ \text{retrieval} \\ \text{software} \end{pmatrix} \qquad d_1 = \begin{pmatrix} 4 \\ 0 \\ 0 \\ 2 \\ 4 \\ 0 \\ 0 \\ 5 \end{pmatrix} \qquad d_2 = \begin{pmatrix} 8 \\ 4 \\ 0 \\ 0 \\ 0 \\ 3 \\ 7 \\ 0 \end{pmatrix} \qquad d_3 = \begin{pmatrix} 6 \\ 0 \\ 1 \\ 9 \\ 0 \\ 0 \\ 5 \\ 0 \end{pmatrix}$$

Text Representation

- **Term Weighting Schemes**
  - Inverse Document Frequency (IDF)

    Assumption: Terms that are limited to a few documents are useful for discriminating those documents from the rest of the collection; while terms that occur frequently across the entire collection are less useful.

    - The document frequency is the number of documents containing a particular term.



Text Representation

- **Term Weighting Schemes**
  - Inverse Document Frequency (IDF)

$$\mathrm{idf}_t = \log_{10} \frac{N}{\mathrm{df}_t}$$

where is the total number of documents in the document collection, is the number of the documents containing the term .

- IDF formulates the distribution of terms across the collection as a whole and it is a measure of the general importance of a term *t* in the document collection

$$w_{t,d} = tf_{t,d} \times idf_t$$

  - TF-IDF Weight Scheme

Text Representation

- **Term Weighting Schemes**
  - Inverse Document Frequency (IDF)

tf	d_1	d_2	d_3
t_1=computer	4	8	6
t_2=document	0	4	0
t_3=filtering	0	0	1
t_4=information	2	0	9
t_5=language	4	0	0
t_6=library	0	3	0
t_7=retrieval	0	7	5
t_8=software	5	0	0

df

$$\text{idf}_t = \log_{10} \frac{N}{\text{df}_t}$$

Text Representation

tf	d_1	d_2	d_3	d_4
t_1	1	0	0	1
t_2	0	1	0	1
t_3	0	1	1	1
t_4	0	1	1	0
t_5	1	1	0	1

df
2
2
3
2
3

idf
log(4/2)
log(4/2)
log(4/3)
log(4/2)
log(4/3)

tf-idf	d_1	d_2	d_3	d_4
t_1	1 log(4/2)=0.30	0	0	1 log(4/2)=0.30
t_2	0	1 log(4/2)=0.30	0	1 log(4/2)=0.30
t_3	0	1 log(4/3)= 0.12	1 log(4/3)= 0.12	1 log(4/3)= 0.12
t_4	0	1 log(4/2)=0.30	1 log(4/2)=0.30	0

Text Representation

- **Term Weighting Schemes**

Features

Feature Values

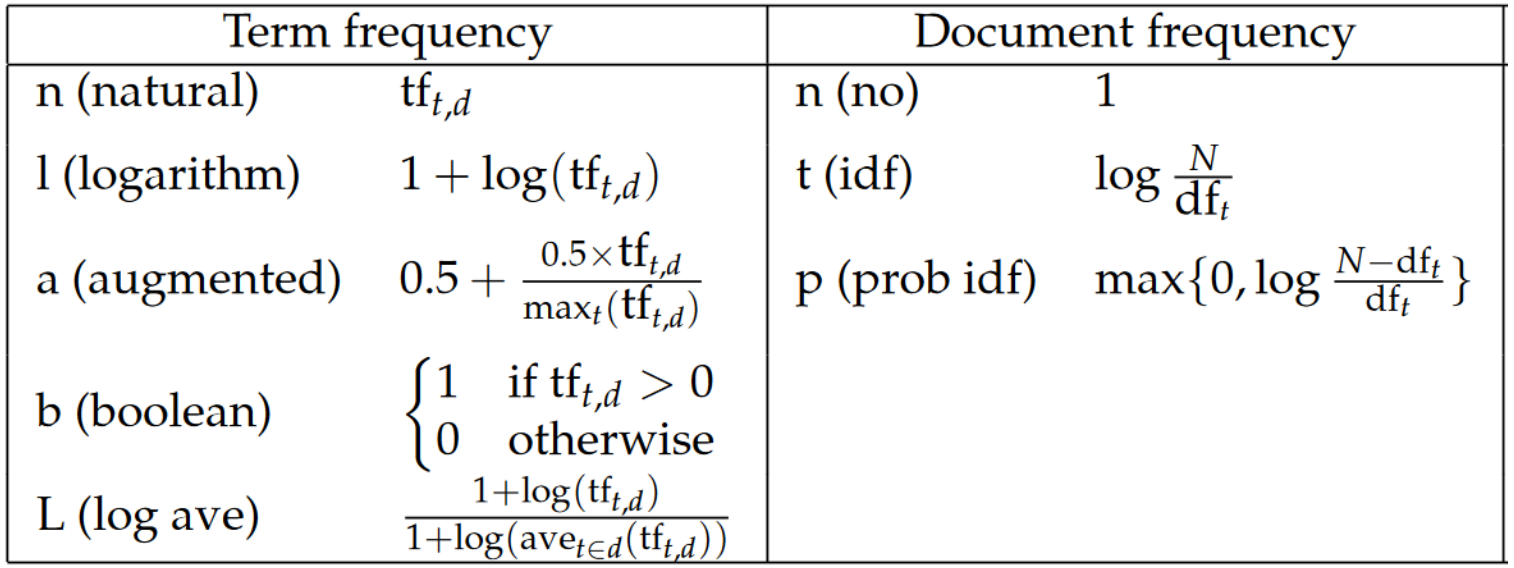
	d_1	d_2	d_3	d_4
t_1	0.30	0	0	0.30
t_2	0	0.30	0	0.30
t_3	0	0.12	0.12	0.12
t_4	0	0.30	0.30	0
t_5	0.12	0.12	0	0.12

$$d_1 = \begin{pmatrix} 0.3 \\ 0 \\ 0 \\ 0 \\ 0.12 \end{pmatrix} \quad d_2 = \begin{pmatrix} 0 \\ 0.30 \\ 0.12 \\ 0.30 \\ 0.12 \end{pmatrix} \quad d_3 = \begin{pmatrix} 0 \\ 0 \\ 0.12 \\ 0.30 \\ 0 \end{pmatrix} \quad d_4 = \begin{pmatrix} 0.30 \\ 0.30 \\ 0.12 \\ 0 \\ 0.12 \end{pmatrix}$$

Document Vectors

Text Representation

- **Term Weighting Schemes**
  - TF-IDF Variants



Term frequency		Document frequency	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N - \mathrm{df}_t}{\mathrm{df}_t}\}$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$		
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$		

Text Representation

- **Term Weighting Schemes**
  - (Okapi) BM25 Weighting Scheme

$$\sum_{t\in q} \overbrace{\log\left(\frac{N}{df_t}\right)}^{\text{IDF}} \overbrace{\frac{tf_{t,d}}{k\left(1-b+b\left(\frac{|d|}{|d_{\text{avg}}|}\right)\right)+tf_{t,d}}}^{\text{weighted tf}}$$

 adjusts the balance between TF and IDF. When is 0, BM25 reverts to no use of term frequency, just a binary selection of terms in the query (plus idf). controls the importance of document length normalization. When is 0, there is no length scaling. It is suggested that the reasonable values are k = [1.2, 2] and b = 0.75.

- [Which BM25 Do You Mean? A Large-Scale Reproducibility Study of]

Announcement

- **Lab 1**
  - Venue: PQ604A/B/C
  - Time: 6:30pm ~ 9:20pm
  - Date: Tuesday, February 11, 2025
  - Tutor: Heming Xia

References

- **Book Chapters**
  - Speech and Language Processing
    - Chapter 2: Regular Expressions, Text Normalization, and Edit Distance
    - Chapter 14.1: Information Retrieval
  - Introduction to Information Retrieval
    - Chapter 6: Scoring, Term Weighting and the Vector Space Model

thank you