BASIC TEXT PROCESSING

Gabriella Pasi
gabriella.pasi@diunimib.it

Basic Text Processing

"Basic text processing is often the first step in any text mining application and consists of several layers of simple processing such as tokenization, lemmatization, normalization, or more advanced "

From: Gabe Ignatow and Rada Mihalcea. "An Introduction to Text Mining". Apple Books.

Text and documents

Textual Documents: simplest case

- Free-format text
  - EBCDIC coding, ASCII (8 bit), UNICODE (16 bit), etc.

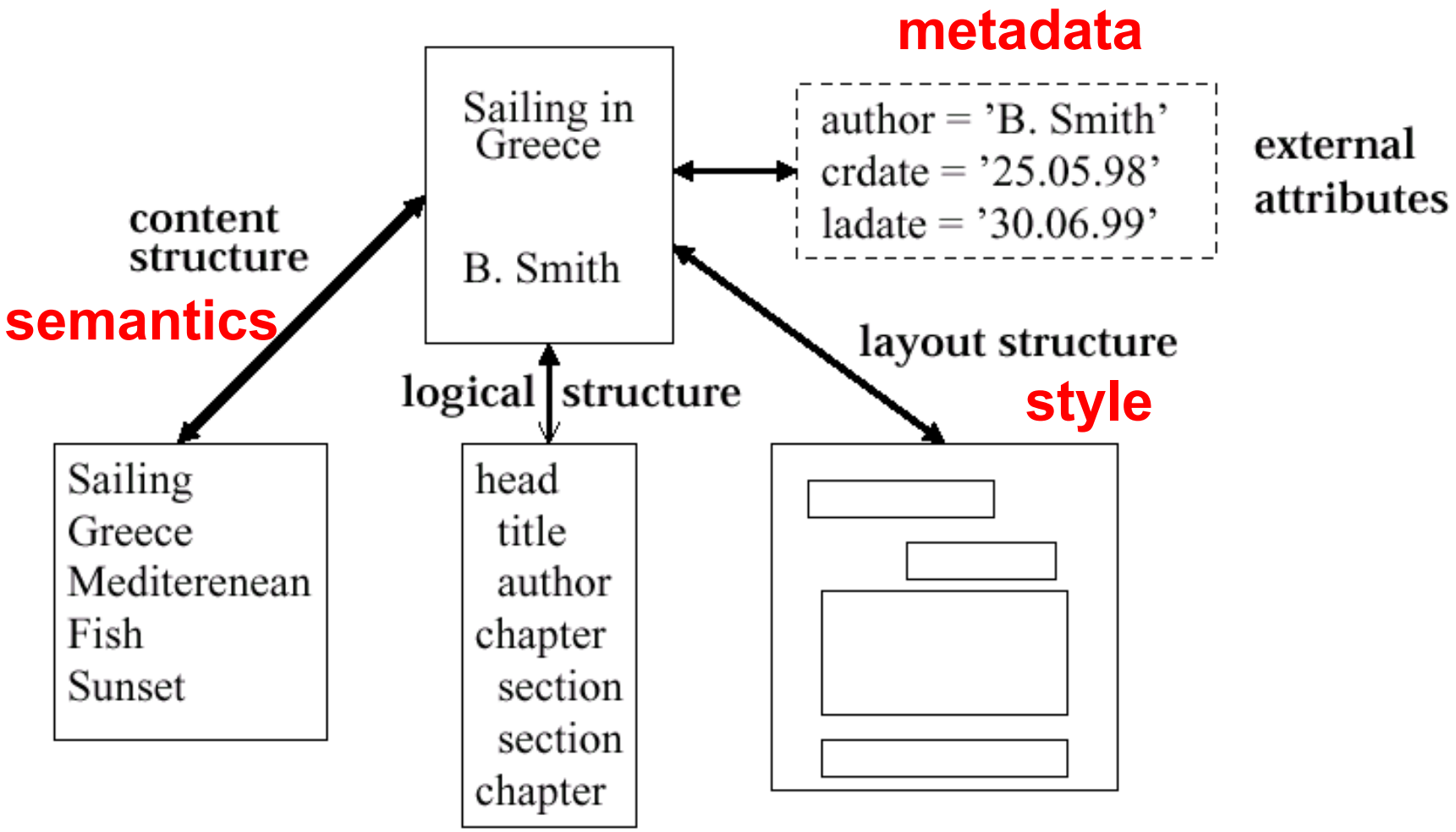
    → Language
  - Text consisting of strings of characters from an alphabet, etc.
    - E.g., genome sequences, formulas of chemical compounds, words in natural language

- Examples
  - Articles from newspapers, magazines, messages, letters, medical reports, Web pages, etc.

Document characteristics

- Document
  - Text
    +
  - Structure
    +
  - Other media (images, sounds, …)
    +
  - Metadata

What is a document?



Metadata

- Metadata associated with a document is data related to the document:
  - Descriptive metadata (Dublin Core Metadata Set):
    - Relating to the creation of the document
    - E.g.,: title, authors, date, length (in pages, words, bytes, etc.), genre (book, article, memo, mail, etc.)
  - Semantic metadata
    - Relating to the subject dealt with in the document
    - E.g.,: Library of Congress subject codes, controlled keywords extracted from an ontology

Metadata

- Taken from: http://www.w3.org/Metadata/Activity

- Metadata is *information about information* - labeling, cataloging and descriptive information structured in such a way that allows pages to be properly searched and processed in particular *by computer*. In other words, what is now very much needed on the Web is *metadata*.

- W3C's (World Wide Web Consortium) Metadata Activity is concerned with ways to model and encode metadata.

Formats for text documents

In classical IR systems, documents had to be represented in an "internal" format in order to be indexed and managed

- 1st OPERATION: source file → generation or input file

- Documents written with common word-processors:
  - Word
  - TeX, RTF, HTML, XML (Rich Text Format) (ASCII format)

- Formats for viewing and printing:
  - PDF (Portable Document Format)
  - MIME (Multipurpose Internet Mail Exchange) for e-mail, it supports various character encodings

- Compressed formats:
  - ARJ, ZIP (Winzip, Gzip)

Basic Text Processing

How to represent a text

- How to represent a text?
  - Make it computable
- Represent by a string?
  - No semantic meaning
- Represent by a list of sentences?
  - Sentence is just like a short document (recursive definition)

Words as the basic “features” of texts

- Doc1: Text mining is to identify useful information.
- Doc2: Useful information is mined from text.
- Doc3: Apple is delicious.

Initial stages of text processing

- Tokenization
  - Cut character sequence into word tokens
    - Deal with ***"John's", a state-of-the-art solution***
- Normalization
  - Map text and query term to same form
    - You want ***U.S.A.*** and ***USA*** to match
- Stemming
  - We may wish different forms of a root to match
    - ***authorize***, ***authorization***
- Stop words removal
  - We may omit very common words (or not)
    - ***the, a, to, of***

Basic Text Processing

- Words and Corpora

How many words in a sentence?

- "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 in a sentence?

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 in a corpus?

N = number of tokens

V = vocabulary = set of types, **|*V*|** is size of vocabulary

Heaps Law = Herdan's Law = $|V| = kN^{\beta}$ where often .67 < β < .75

i.e., vocabulary size grows with > square root of the number of word tokens

	Tokens = N	Types = |V|
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
COCA	440 million	2 million
Google N-grams	1 trillion	13+ million

Corpora

Words don't appear out of nowhere!

A text is produced by

- a specific writer(s),
- at a specific time,
- in a specific variety,
- of a specific language,
- for a specific function.

Corpora vary along dimension like

- **Language**: 7097 languages in the world
- **Variety**, like African American Language varieties.
  - AAE Twitter posts might include forms like "*iont" (I don't)*
- **Code switching**, e.g., Spanish/English, Hindi/English:

S/E: Por primera vez veo a @username actually being hateful! It was beautiful:)

[For the first time I get to see @username actually being hateful! it was beautiful:)]

H/E: dost tha or ra- hega ... dont wory ... but dherya rakhe

[“he was and will remain a friend ... don’t worry ... but have faith”]

- **Genre:** newswire, fiction, scientific articles, Wikipedia
- **Author Demographics**: writer's age, gender, ethnicity, SES

BASIC TEXT PROCESSING

WORD TOKENIZATION

Text Normalization

- Every NLP task requires text normalization:
  1. Tokenizing (segmenting) words
  2. Normalizing word formats
  3. Segmenting sentences

Space-based tokenization

- A very simple way to tokenize
  - For languages that use space characters between words
    - Arabic, Cyrillic, Greek, Latin, etc., based writing systems
  - Segment off a token between instances of spaces
- Unix tools for space-based tokenization
  - The "tr" command
  - Inspired by Ken Church's UNIX for Poets
  - Given a text file, output the word tokens and their frequencies

Tokenization

- Input: "***Friends, Romans and Countrymen***"
- Output: Tokens
  - ***Friends***
  - ***Romans***
  - ***and***
  - ***Countrymen***
- A token is an instance of a sequence of characters
- Each such token is now a candidate to be a meaningful term (index), after further processing
  - Described below
- But what are valid tokens to emit?

Issues in Tokenization

- Can't just blindly remove punctuation:
  - m.p.h., Ph.D., AT&T, cap'n
  - prices ($45.55)
  - dates (01/02/06)
  - URLs (http://www.stanford.edu)
  - hashtags (#nlproc)
  - email addresses (someone@cs.colorado.edu)
- Clitic: a word that doesn't stand on its own
  - "are" in we're, French "je" in j'ai, "le" in l'honneur
- When should multiword expressions (MWE) be words?
  - New York, rock 'n' roll

Tokenization

- Issues in tokenization:
  - ***Finland's capital*** $\rightarrow$

    Finland AND ***s***? ***Finlands***? ***Finland's***?
  - ***Hewlett-Packard*** $\rightarrow$ ***Hewlett*** and ***Packard*** as two tokens?
    - ***state-of-the-art***: break up hyphenated sequence.
    - ***co-education***
    - ***lowercase***, ***lower-case***, ***lower case*** ?
  - ***San Francisco***: one token or two?
    - How do you decide it is one token?

Numbers

- ***3/20/91*** ***Mar. 12, 1991***
  20/3/91
- ***55 B.C.***
- ***B-52***
- ***My PGP key is 324a3df234cb23e***
- ***(800) 234-2333***
  - Often have embedded spaces
  - Older IR systems may not index numbers
    - But often very useful: think about things like looking up codes on the web

Tokenization: language issues

- French
  - ***L'ensemble*** → one token or two?
    - ***L*** ? ***L'*** ? ***Le*** ?
    - Want ***l'ensemble*** to match with ***un ensemble***
      - Until at least 2003, it didn't on Google
- German noun compounds are not segmented
  - ***Lebensversicherungsgesellschaftsangestellter***
  - 'life insurance company employee'
  - German retrieval systems benefit greatly from a **compound splitter** module
    - Can give a 15% performance boost for German

Tokenization in NLTK (Natural Language Toolkit https://www.nltk.org/)

Bird, Loper and Klein (2009), *Natural Language Processing with Python*. O'Reilly

```
>>> 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', '...']
```

Tokenization in languages without spaces

Many languages (like Chinese, Japanese, Thai) don't use spaces to separate words!

How do we decide where the token boundaries should be?

Word tokenization in Chinese

Chinese words are composed of characters called "**hanzi**" (or sometimes just "**zi**")

Each one represents a meaning unit called a morpheme.

Each word has on average 2.4 of them.

But deciding what counts as a word is complex and not agreed upon.

How to do word tokenization in Chinese?

- 姚明进入总决赛 “Yao Ming reaches the finals”

- 3 words?
- 姚明 进入 总决赛
- YaoMing reaches finals

- 5 words?
- 姚 明 进入 总 决赛
- Yao Ming reaches overall finals

- 7 characters? (don't use words at all):
- 姚 明 进 入 总 决 赛
- Yao Ming enter enter overall decision game

Word tokenization / segmentation

So in Chinese it's common to just treat each character (zi) as a token.

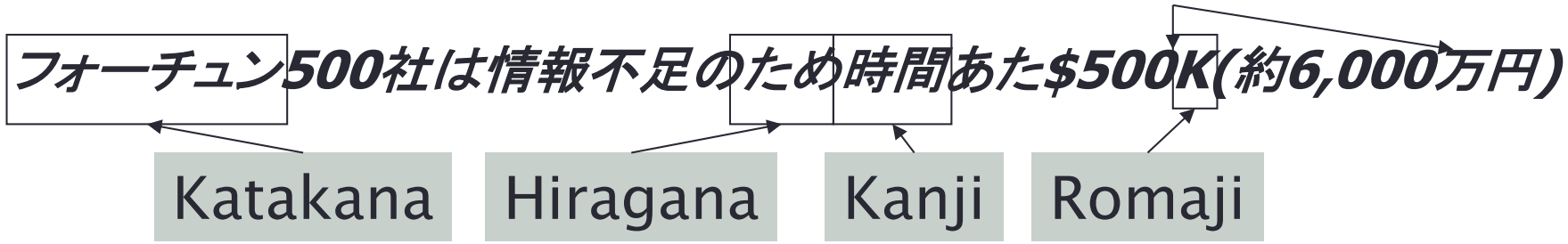
- So the **segmentation** step is very simple

In other languages (like Thai and Japanese), more complex word segmentation is required.

- The standard algorithms are neural sequence models trained by supervised machine learning.

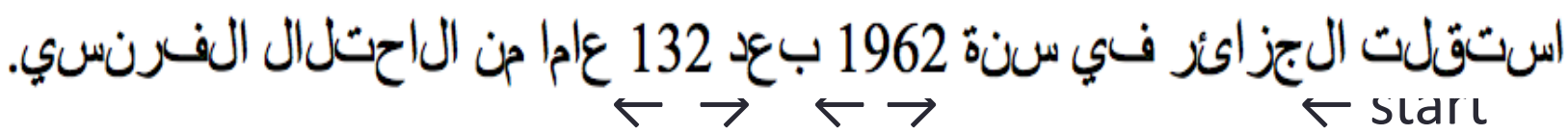
Tokenization: language issues

- Further complicated in Japanese, with multiple alphabets intermingled
  - Dates/amounts in multiple formats



End-user can express query entirely in hiragana!

Tokenization: language issues

- Arabic (or Hebrew) is basically written right to left, but with certain items like numbers written left to right
- Words are separated, but letter forms within a word form complex ligatures
- 

- ‘Algeria achieved its independence in 1962 after 132 years of French occupation.’
- With Unicode, the surface presentation is complex, but the stored form is straightforward

Tokenization

- Break a stream of text into meaningful units
  - Tokens: words, phrases, symbols
    - **Input:** It’s not straight-forward to perform so-called “tokenization.”
    - **Output(1):** 'It’s', 'not', 'straight-forward', 'to', 'perform', 'so-called', '“tokenization.”'
    - **Output(2):** 'It', '’', 's', 'not', 'straight', '-', 'forward, 'to', 'perform', 'so', '-', 'called', ‘“', 'tokenization', '.', '”’‘
  - Definition depends on language, corpus, or even context

Another option for text tokenization

Instead of

- white-space segmentation
- single-character segmentation

Use the data to tell us how to tokenize.

Subword tokenization (because tokens can be parts of words as well as whole words)

Subword tokenization

- Three common algorithms:
  - **Byte-Pair Encoding (BPE)** (Sennrich et al., 2016)
    - https://huggingface.co/learn/nlp-course/chapter6/5
  - **Unigram language modeling tokenization** (Kudo, 2018)
    - https://huggingface.co/learn/nlp-course/chapter6/7
  - **WordPiece** (Schuster and Nakajima, 2012)
    - https://huggingface.co/learn/nlp-course/chapter6/6
- All have 2 parts:
  - A token **learner** that takes a raw training corpus and induces a vocabulary (a set of tokens).
  - A token **segmenter** that takes a raw test sentence and tokenizes it according to that vocabulary

Tokenization recap

- Solutions
  - Regular expressions (https://www.nltk.org/api/nltk.tokenize.regexp.html)
    - [\w]+: so-called -> ‘so’, ‘called’
    - [\S]+: It’s -> ‘It’s’ instead of ‘It’, ‘’s’
  - Statistical methods
    - Explore rich features to decide where the boundary of a word is
      - Apache OpenNLP (http://opennlp.apache.org/)
      - Stanford NLP tokenizer (http://nlp.stanford.edu/software/lex-parser.shtml)
    - Online Demo
      - Stanford (https://corenlp.run/)

WORD NORMALIZATION

Word Normalization

- Putting words/tokens in a standard format
  - U.S.A. or USA
  - uhhuh or uh-huh
  - Fed or fed
  - am, is, be, are

Normalization to terms

- We may need to “normalize” words :
  - We want to match ***U.S.A.*** and ***USA***
- Result is terms: a term is a (normalized) word type
- Most commonly, an implicit definition of equivalence classes of terms is obtained by, e.g.,
  - deleting periods to form a term
    - ***U.S.A., USA*** (***USA***
  - deleting hyphens to form a term
    - ***anti-discriminatory, antidiscriminatory*** (***antidiscriminatory***

Normalization: other languages

- Accents: e.g., French ***résumé*** vs. ***resume*.**
- Umlauts: e.g., German: ***Tuebingen*** vs. ***Tübingen***
  - Should be equivalent
- This is important is search engines:
  - How are users like to write their queries for these words?

- Even in languages that standardly have accents, users often may not type them
  - Often best to normalize to a de-accented term
    - ***Tuebingen, Tübingen, Tubingen*** ⎩ ***Tubingen***

Normalization: other languages

- Normalization of things like date forms
  - ***7月30日 vs. 7/30***
  - ***Japanese use of kana vs. Chinese characters***

- Tokenization and normalization may depend on the language and so is intertwined with language detection

- Crucial: in search engines there is the need to "normalize" indexed text as well as query terms identically

Morgen will ich in MIT …

Is this German "mit"?

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)

Thesauri and soundex

- How we can handle synonyms and homonyms?
  - E.g., by hand-constructed equivalence classes
    - ***car*** = ***automobile*** ***color*** = ***colour***
  - We can rewrite to form equivalence-class terms
    - When the document contains ***automobile***, index it under ***car-automobile*** (and vice-versa)
  - Or in search engines we can expand a query
    - When the query contains ***automobile***, look under ***car*** as well
- What about spelling mistakes?
  - One approach is Soundex, which forms equivalence classes of words based on phonetic heuristics

LEMMATIZATION AND STEMMING

Lemmatization

- Reduce inflectional/variant forms to base form
- E.g.,
  - *am, are, is* → *be*
  - *car, cars, car's, cars'* → *car*
- *the boy's cars are different colors* → *the boy car be different color*
- Lemmatization implies doing "proper" reduction to dictionary headword form

Lemmatization is done by Morphological Parsing

- Morphemes:
  - The small meaningful units that make up words
  - Stems: The core meaning-bearing units
  - Affixes: Parts that adhere to stems, often with grammatical functions
- Morphological Parsers:
  - Parse *cats* into two morphemes *cat* and *s*

Stemming

- Reduce terms to stems
- "Stemming" suggests crude affix chopping
  - 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 Stemmer

- Commonest algorithm for stemming English

- Based on a series of rewrite rules run in sequence
  - A cascade (sequentially), in which output of each pass fed to next pass

- Some sample rules:

$$\begin{aligned} \text{ATIONAL} &\rightarrow \text{ATE} \quad \text{(e.g., relational} \rightarrow \text{relate)} \\ \text{ING} &\rightarrow \epsilon \quad \text{if stem contains vowel (e.g., motoring} \rightarrow \text{motor)} \\ \text{SSES} &\rightarrow \text{SS} \quad \text{(e.g., grasses} \rightarrow \text{grass)} \end{aligned}$$

Other stemmers

- Other stemmers exist:
  - Lovins stemmer
    - http://www.comp.lancs.ac.uk/computing/research/stemming/general/lovins.htm

      Single-pass, longest suffix removal (about 250 rules)
  - Paice/Husk stemmer
  - Snowball
- Full morphological analysis (lemmatization)
  - At most modest benefits for retrieval

Language-specificity

- The above methods embody transformations that are
  - Language-specific, and often
  - Application-specific
- These are “plug-in” addenda to the indexing process
- Both open source and commercial plug-ins are available for handling these

Does stemming help in retrieval?

- English: very mixed results. Helps recall for some queries but harms precision on others
  - E.g., operative (dentistry) ⇒ oper
- Definitely useful for Spanish, German, Finnish, …
  - 30% performance gains for Finnish!

Stop words

- With a stop list, you exclude from the dictionary entirely the commonest words. Intuition:
  - They have little semantic content: *the, a, and, to, be*
  - There are a lot of them
- However, for example Web SE do not use stoplists, due to the following reasons:
  - Good compression techniques means the space for including stop words in a system is very small
  - You need them for:
    - Phrase queries: “King of Denmark”
    - Various song titles, etc.: “Let it be”, “To be or not to be”
    - “Relational” queries: “flights to London”