Information Retrieval
CS 6900

Lecture 04

Razvan C. Bunescu

School of Electrical Engineering and Computer Science

bunescu@ohio.edu

Tokenization: From Text to Tokens

- **Tokenization** = segmenting text into tokens:
  - **token** = a sequence of characters, in a particular document at a particular position.
  - **type** = the class of all tokens that contain the same character sequence.
    - "... to **be** or not to **be** ..."
    - "... so **be** it, he said ..."

    3 tokens, 1 type
  - **term** = a (normalized) type that is included in the IR dictionary.
    - *text* = "to sleep perchance to dream"
    - *tokens* = to, sleep, perchance, to, dream
    - *types* = to, sleep, perchance, dream
    - *terms* = sleep, perchance, dream (stopword removal).

Tokenization: From Text to Tokens

- Split on whitespace and non-alphanumeric?
  - Good as a starting point, but complicated by many tricky cases:
    - Appostrophes are ambiguous:
      - possessive constructions:
        - » the books’s cover => the book s cover
      - contractions:
        - » he’s happy => he is happy
        - » aren’t => are not
      - quotations:
        - » ‘let it be’ => let it be

Tokenization: From Text to Tokens

- Split on whitespace and non-alphanumeric?
  - Good as a starting point, but complicated by many tricky cases:
    - Whitespaces in proper names or collocations:
      - San Francisco => San_Francisco
        - how do we determine it should be a single token?
    - Hyphenations:
      - co-education => co-education
      - state-of-the-art => state of the art? state_of_the_art?
      - lowercase, lower-case, lower case => lower_case
      - Hewlett-Packard => Hewlett_Packard? Hewlett Packard?
    - Whitespaces and Hyphenations:
      - San Francisco-Los Angeles => San_Francisco Los_Angeles

Tokenization: From Text to Tokens

- Split on whitespace and non-alphanumeric?
  - Good as a starting point, but complicated by many tricky cases:
    - Whitespaces and Hyphenations:
      - split on hyphens and whitespaces, but use a phrase index.
    - Unusual strings that should be recognized as tokens:
      - C++, C#, B-52, C4.5,M*A*S*H.
    - URLs, IP addresses, email addresses, tracking numbers.
      - exclude numbers, monetary amounts, URLs from indexing?

- **Use same tokenization rules for queries and documents!**

Tokenization is Language Dependent

- Need to know the language of document/query:
  - **Language Identification**, based on classifiers trained on short character subsequences as features, is highly effective.
  - French (reduced definite article, postposed clitic pronouns):
    - l'ensemble, un ensemble, donne-moi.
  - German (compund nouns), need *compound splitter*:
    - Computerlinguistik
    - Lebensversicherungsgesellschaftsangestellter
  - East Asian languages, need *word segmenter*:
    - 莎拉波娃现在居住在美国东南部的佛罗里达。
      - Not always guaranteed a unique tokenization
    - Complicated in Japanese, with multiple alphabets intermingled.

Tokenization is Language Dependent

- Need to know the language of document/query:
  - Arabic and Hebrew:
    - 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

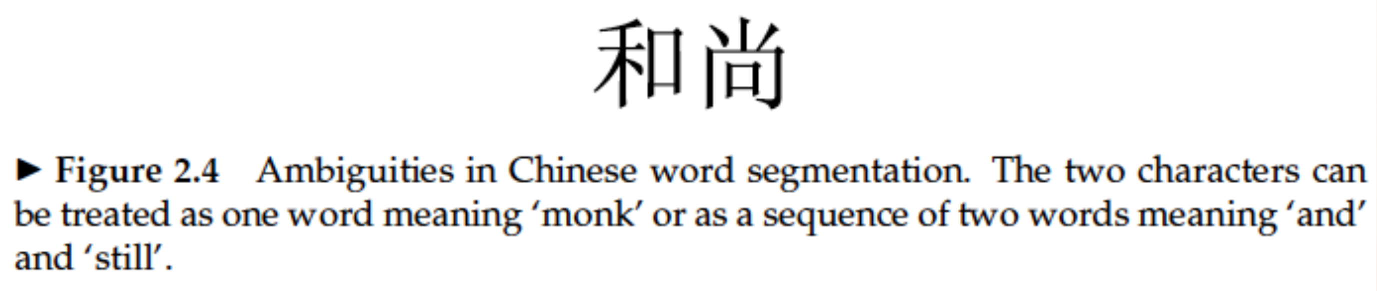
استقلت الجزائر في سنة 1962 بعد 132 عاما من الاحتلال الفرنسي.

← → ← → ← start

Algeria achieved its independence in 1962 after 132 years of French occupation.

Language Dependent Processing

- Compound Splitting for German:
  - usually implemented by finding segments that match against dictionary entries.
- Word Segmentation for Chinese:
  - ML sequence tagging models trained on manually segmented text:
    - *Logistic Regression, HMMs, Conditional Random Fields*.
  - Multiple segmentations are possible:



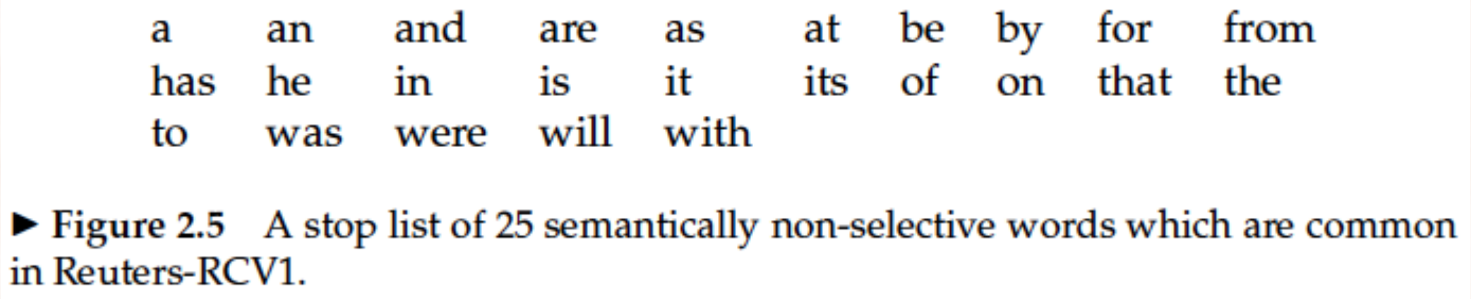
和尚

► **Figure 2.4** Ambiguities in Chinese word segmentation. The two characters can be treated as one word meaning 'monk' or as a sequence of two words meaning 'and' and 'still'.

From Tokens to Terms: Stop words

- Exclude from the dictionary the most common words.
  - They have little semantic content: *the, a, and, to, be*
  - There are a lot of them: ~30% of postings for top 30 words.

- **Stop words** = list of most common words:
  - sort tokens by *collection frequency*.
  - select most common types, often hand-filtered based on semantic content.



a	an	and	are	as	at	be	by	for	from
has	he	in	is	it	its	of	on	that	the
to	was	were	will	with					

► **Figure 2.5** A stop list of 25 semantically non-selective words which are common in Reuters-RCV1.

From Tokens to Terms: Stop words

- But the trend is away from doing this:
  - From large stop lists (200-300), to small stop lists (7-12), to none.
  - Good compression techniques (IIR 5) means the space for including stop words in a system is very small.
  - Good query optimization techniques (IIR 7) mean you pay little at query time for including stop words.
  - 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"

From Tokens to Terms: Normalization

- **Token Normalization** = reducing multiple tokens to the same canonical term, such that matches occur despite superficial differences.
  1. Create equivalence classes, named after one member of the class:
     - {anti-discriminatory, antidiscriminatory}
     - {U.S.A., USA}
       - but what about C.A.T vs. CAT?
  2. Maintain relations between unnormalized tokens:
     - o can be extended with lists of synonyms (car, automobile).
     1. Index unnormalized tokens, a query term is expanded into a disjunction of multiple postings lists.
     2. Perform expansion during index construction.

From Tokens to Terms: Normalization

- Accents and diacritics in French:
  - ***résumé*** vs. ***resume.***
- Umlauts in German:
  - ***Tuebingen*** vs. ***Tübingen***
- Most important criterion:
  - 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***

From Tokens to Terms: Normalization

- **Case-Folding** = reduce all letters to lower case:
  - allow Automobile at beginning of sentences to match automobile.
  - allow matching user typed ferrari to match Ferrari in documents.
  - but may lead to unintended matches:
    - the Fed vs. fed.
    - Bush, Black, General Motors, Associated Press, ...
- **Heuristic** = lowercase only some tokens:
  - words at beginning of sentences.
  - all words in a title where most words are capitalized.
- **Truecasing** = use a classifier to decide when to fold:
  - trained on many heuristic features.

From Tokens to Terms: Normalization

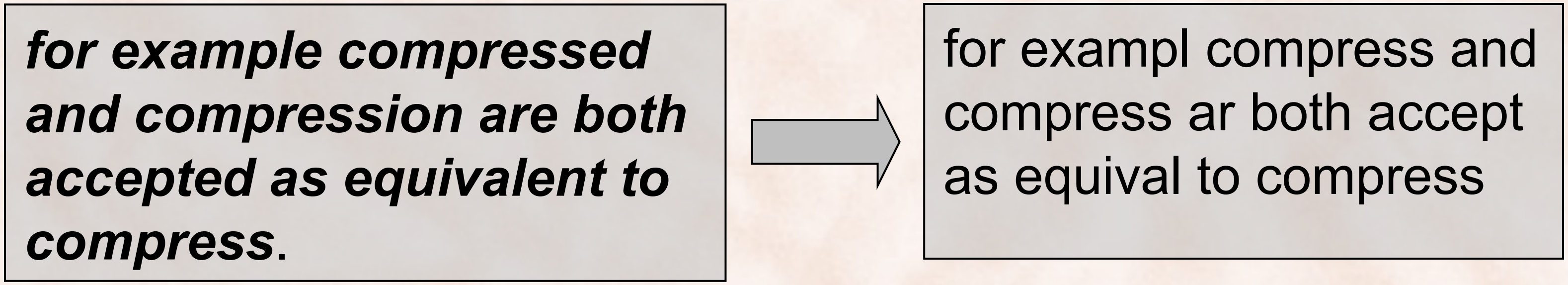
- British vs. American spellings:
  - colour vs. color.
- Multiple formats for dates, times:
  - 09/30/2013 vs. Sep 30, 2013.
- Asymmetric expansion:
  - Enter: ***window*** Search: ***window, windows***
  - Enter: ***windows*** Search: ***Windows, windows, window***
  - Enter: ***Windows*** Search: ***Windows***

Lemmatization and Stemming

- **Lemmatization** = reduce a word to its base/dictionary form, i.e. its lemma:
  - is, am, are => be
  - car, cars => car
- Lemmatization commonly only collapses the different *inflectional* forms of a lemma:
  - saw => see (if verb), or saw (if noun).

From Tokens to Terms: Stemming

- **Stemming** = reduce *inflectional* and sometimes *derivationally* related forms of a word to a common base form i.e. the *stem*.
  - automate, automates, automatic, automation => automat
  - see, saw => s
- Crude affix chopping that is language dependent:



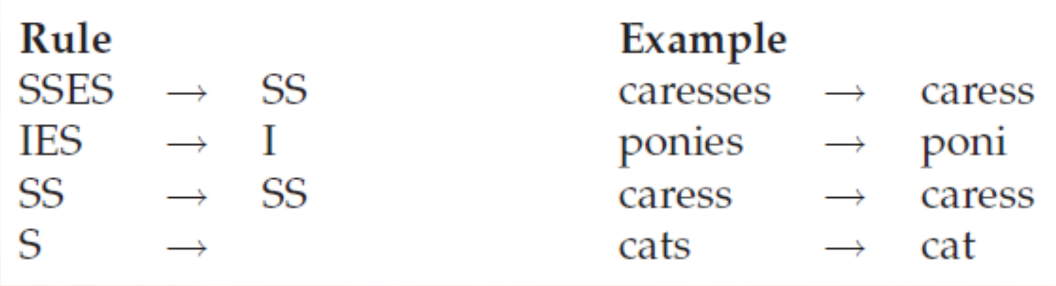
***for example compressed and compression are both accepted as equivalent to compress*.**

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

Porter’s Algorithm

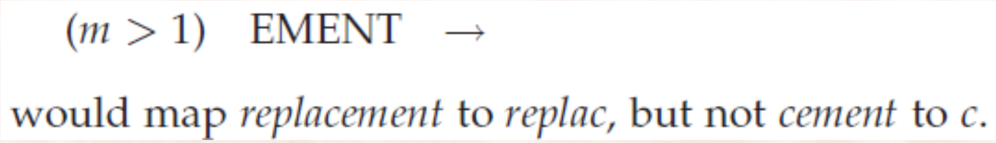
http://www.tartarus.org/˜martin/PorterStemmer/

- The most common stemmer for English:
  - at least as good as other stemming options.
  - 5 phases of word reductions, applied sequentially.
  - conventions for rule selection and application:
    - select the reduction rule that applies to the longest suffix:



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

    - check the number of syllables, for suffix determination:



$(m > 1)$ EMENT →

would map *replacement* to *replac*, but not *cement* to *c*.

Other Stemming Algorithms

- **Lovins** stemmer, **Paice/Husk** stemmer, **Snowball**:
  - http://www.cs.waikato.ac.nz/˜eibe/stemmers/
  - http://www.comp.lancs.ac.uk/computing/research/stemming/
- Stemming is language- and often application-specific:
  - open source and commercial plug-ins.
- Does it improve IR performance?
  - mixed results for English: improves recall, but hurts precision.
    - operative (dentistry) ⇒ oper
  - definitely useful for languages with richer morphology:
    - Spanish, German, Finish (30% gains).