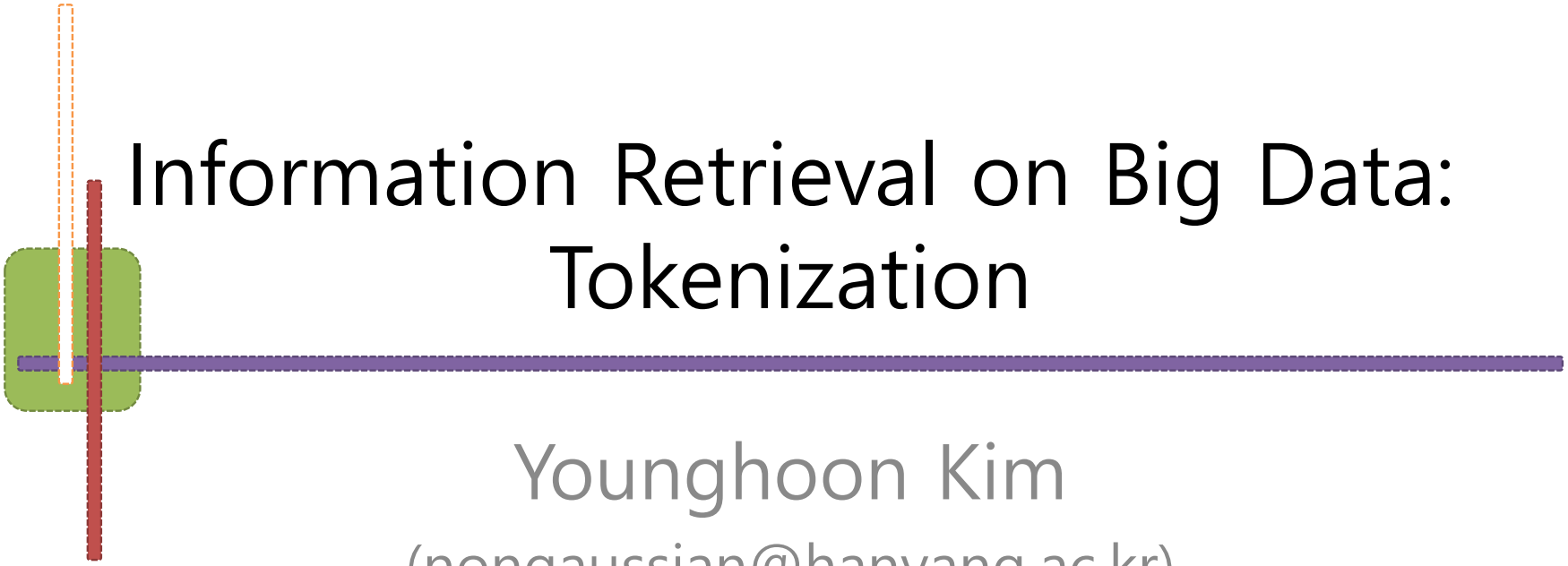


Information Retrieval on Big Data: Tokenization

Younghoon Kim

(nongaussian@hanyang.ac.kr)

Modified slides originally written by Jim Martin, Donald Patterson Min-Yen Kan, and Zhang & Helmer, used for the Stanford CS276 class and from the Stuttgart IIR class
https://nlp.stanford.edu/IR-book/newslides.html

Types of Documents

- What format is it in?
  - pdf/word/excel/html?
- What language is it in?
- What character set is in use?
  - (CP949, UTF-8, …)

What is a document?

- We return from our query "documents" but there are often interesting questions of grain size:
- What is a unit document?
  - A file?
  - An email? (Perhaps one of many in a single mbox file)
    - What about an email with 5 attachments?
  - A group of files (e.g., PPT or LaTeX split over HTML pages)

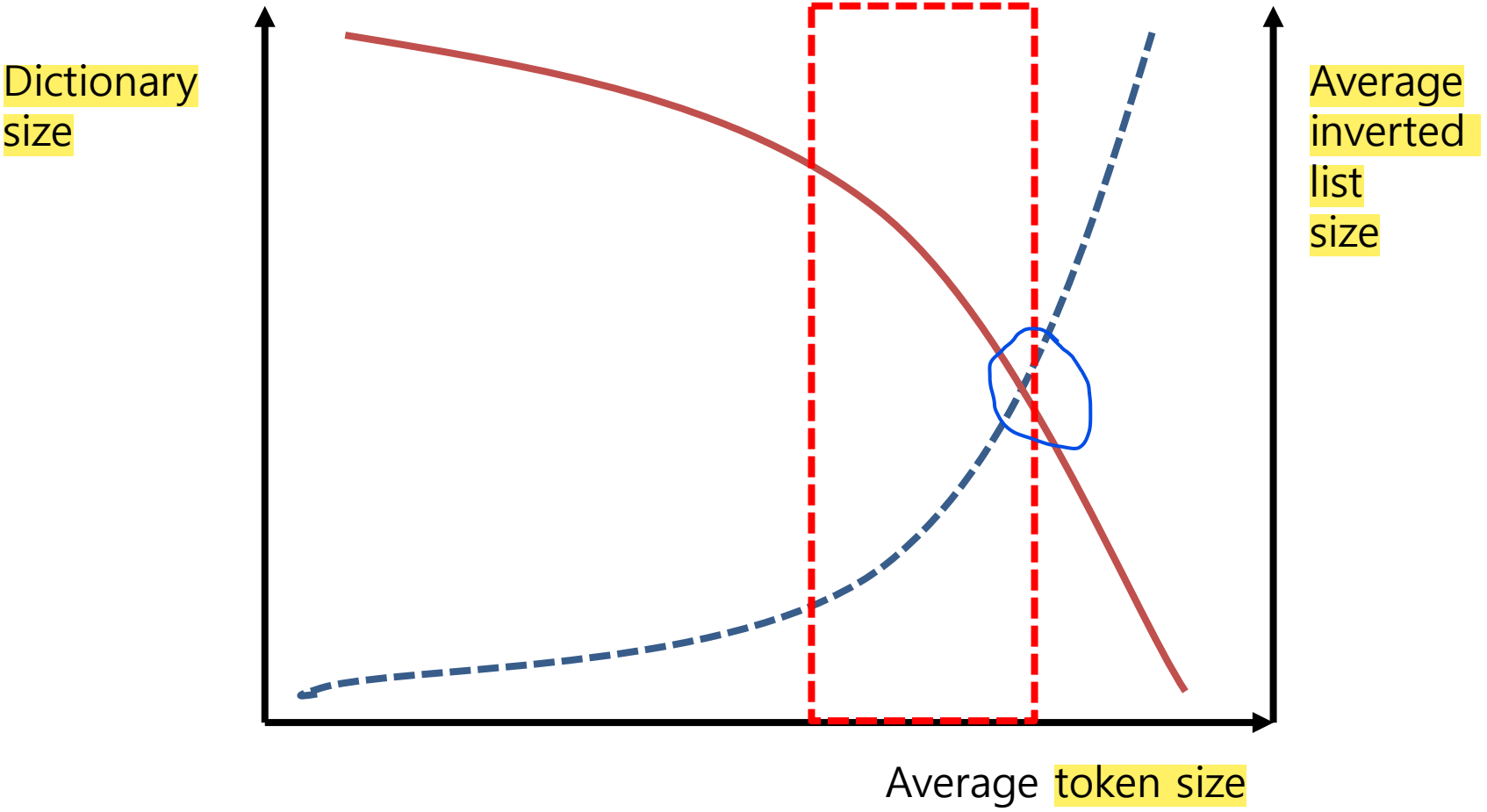
List of common problems

TOKENS

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 for an index entry, after further processing

Average Token Size vs. Dictionary Size



Tokenization

For *O'Neill*, which of the following is the desired tokenization?

- neill
- oneill
- o'neill
- o' neill
- o neill ?

And for *aren't*, is it:

- aren't
- arent
- are n't
- aren t ?

Tokenization

- Issues in tokenization:
  - *Finland's capital :*
    Finland AND *s* ? *Finlands* ? *Finland's* ?
  - *Hewlett-Packard* : *Hewlett* and *Packard* as two tokens?
    - *state-of-the-art*: break up hyphenated sequence?
    - *co-education, coeducation*
    - *lowercase*, *lower-case*, *lower case* ?
    - It can be effective to get the user to put in possible hyphens
  - *San Francisco*: one token or two?
    - How do you decide it is one token?

Numbers

- *3/20/91*
- *Mar. 20, 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 error codes/stacktraces on the web
    - One answer is using n-grams
  - Will often index "meta-data" separately
    - Creation date, format, etc.

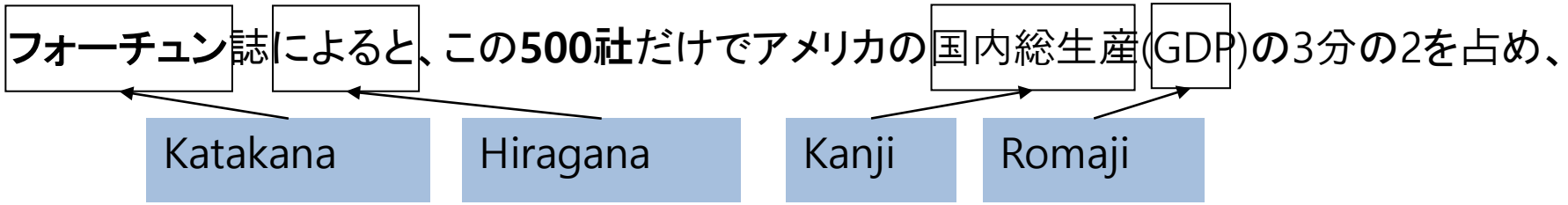
Tokenization: language issues

- French
  - *L'ensemble* (=weight) → one token or two?
    - *L* ? *L'* ? *Le* ?
    - Want *l'ensemble* to match with *un ensemble*
      - Until at least 2003, it didn't on Google
        - » Internationalization!

- 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: language issues

- Chinese and Japanese have no spaces between words:
  - 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
- Further complicated in Japanese, with multiple alphabets intermingled
  - E.g.,



End-user can express query entirely in hiragana!

Tokenization: language issues

- Korean
  - Spacing:
    - 아버지가방에들어가신다
    - 주택저당채권유동화회사법
  - Verb variations:
    - 먹다
    - 먹히다
    - 먹었다
    - 먹고
    - 먹어라
  - Honorific
    - 먹다 → 드시다
    - 자다 → 주무시다

Tokenization: language issues

농협 용인 육가공 공장 vs. 농협용 인육가공 공장

내동 생고기 vs. 내 동생 고기

https://kh-kim.gitbook.io/natural-language-processing-with-pytorch/00-cover/04-korean-is-hell

TERMS: THE THINGS INDEXED IN AN IR SYSTEM

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: ~30% of postings for top 30 words
- But the trend is away from doing this:
  - 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"

Normalization to terms

- We may need to "normalize" words in indexed text as well as query words into the same form
  - We want to match *U.S.A.* and *USA*
- Result is terms: a term is a (normalized) word type, which is an entry in our IR system dictionary
- We most commonly implicitly define equivalence classes of terms 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
- Most important criterion:
  - How are your 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***
  - ***Korean words originated from Chinese***
    - ***E.g., 학교 vs. 學校***
- Reduce all letters to lower case
  - e.g., General Motors, Fed vs. fed
  - Often best to lower case everything, since users will use lowercase regardless of 'correct' capitalization.
- Abbreviations
  - E.g., ASAP, TGIF, F.Y.I., DIY, IMO (IMHO), n/a

Normalization: Stemming

- For grammatical reasons,
  - documents are going to use different forms of a word, such as
    - *organize*, *organizes*, and *organizing*
    - *democracy*, *democratic*, and *democratization*
- The goal of both stemming and lemmatization is
  - to reduce such inflectional forms
- For example
  - The boy's cars are different colors ➔ the boy car be differ color

STEMMING AND LEMMATIZATION

Stemming

- Reduce terms to their "roots" before indexing
- "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

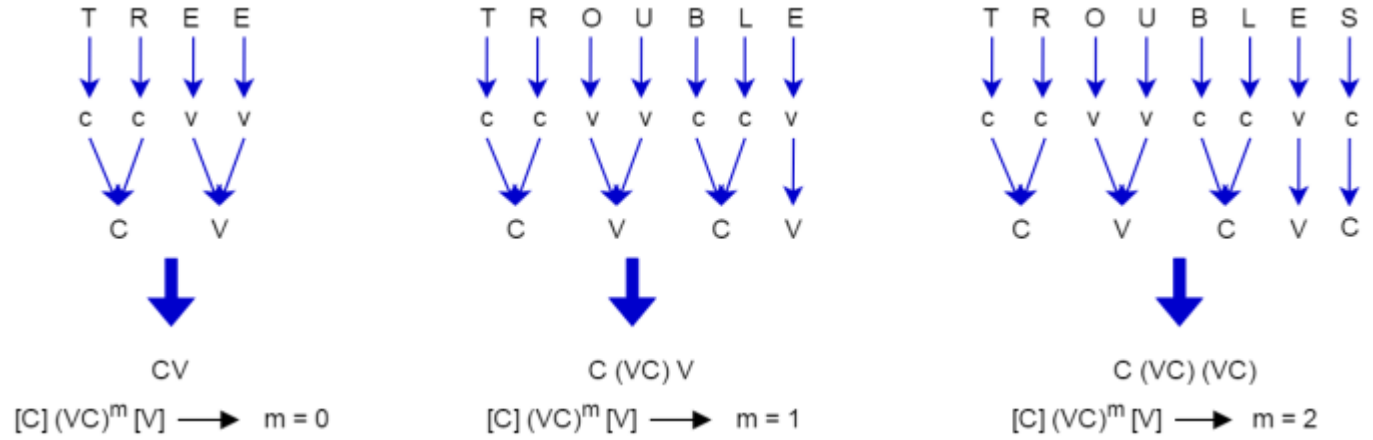
Porter's algorithm

- Developed by a British Computer Scientist named Martin F. Porter
- Commonest algorithm for stemming English
  - Results suggest it's at least as good as other stemming options
- Conventions + 5 phases of reductions
  - phases applied sequentially
  - each phase consists of a set of commands

Consonants and Vowels

- A consonant will be denoted by **c** and a vowel by **v**
- **c+ → C, v+ → V**
- Any word can be represented by

$$[C](VC)^m[V]$$



Exercise

- Determine m of the following:
  - by
  - oats
  - private
  - ivy

Rules

- The 5 steps of Porter's algorithm is represented by the rules of
  - Replacing (or removing) a suffix will be given in the form as

$$\textbf{(condition) S1} \rightarrow \textbf{S2}$$

  - E.g.,

$$(m > 1)\ \text{EMENT} \rightarrow$$

The rule changes
REPLACEMENT → REPLAC

Conditions

- The conditions may contain the following:
  - *S: the stem ends with S (and similarly for the other letters)
  - *v*: the stem contains a vowel
  - *d: the stem ends with a double consonant (e.g. -TT, -SS)
  - *o: the stem ends cvc, where the second c is not W, X or Y (e.g. -WIL, -HOP)

Step 1a

	Rule	Example
1	SSES → SS	CARESSES → CARESS
2	IES → I	PONIES → PONI TIES → TI
3	SS → SS	CARESS → CARESS
4	S →	CATS → CAT

Step 1b

	Rule	Example
1	(m > 0) EED → EE	FEED → FEE AGREED → AGREE
2	(*v*) ED →	PLASTERED → PLASTER (Note that BLED → BLED)
3	(*v*) ING →	MOTORING → MOTOR (Note that SING → SING)

If the second or third of the rules in Step 1b is successful, the following is performed.

	Rule	Example
1	AT → ATE	CONFLAT(ED) → CONFLATE
2	BL → BLE	TROUBL(ED) → TROUBLE
3	IZ → IZE	SIZ(ED) → SIZE
4	(*d and not (*L or *S or *Z)) → single letter	HOPP(ING) → HOP (Note FALL(ING) → FALL)
5	(m=1 and *o) → E	FIL(ING) → FILE (Note FAIL(ING) → FAIL)

Step 1c

	Rule	Example
1	(*v*) Y → I	HAPPY → HAPPI (Note that SKY → SKY)

Step 2

	Rule	Example
1	(m>0) ATIONAL → ATE	RELATIONAL → RELATE
2	(m>0) TIONAL → TION	CONDITIONAL → CONDITION
3	(m>0) ENCI → ENCE	
4	(m>0) ANCI → ANCE	
5	(m>0) IZER → IZE	DIGITIZER → DIGITIZE
6	(m>0) ABLI → ABLE	
7	(m>0) ALLI → AL	
8	(m>0) ENTLI → ENT	
9	(m>0) ELI → E	
10	(m>0) OUSLI → OUS	
11	(m>0) IZATION → IZE	
12	(m>0) ATION → ATE	
13	(m>0) ATOR → ATE	
14	(m>0) ALISM → AL	
15	(m>0) IVENESS → IVE	
16	(m>0) FULNESS → FUL	
17	(m>0) OUSNESS → OUS	
18	(m>2) ALITI → AL	
19	(m>0) IVITI → IVE	SENSITIVITI → SENSITIVE
20	(m>0) BILITI → BLE	

Step 3

	Rule	Example
1	(m>0) ICATE → IC	FORMALIZE → FORMAL
2	(m>0) ATIVE →	ELECTRICITI → ELECTRIC
3	(m>0) ALIZE → AL	ELECTRICAL → ELECTRIC
4	(m>0) ICITI → IC	GOODNESS → GOOD
5	(m>0) ICAL → IC	
6	(m>0) FUL →	
7	(m>0) NESS →	

Step 4

	Rule	Example
1	(m>1) AL →	ALLOWANCE → ALLOW
2	(m>1) ANCE →	INFERENCE → INFER
3	(m>1) ENCE →	REPLACEMENT → REPLAC
4	(m>1) ER →	
5	(m>1) IC →	
6	(m>1) ABLE →	
7	(m>1) IBLE →	
8	(m>1) ANT →	
9	(m>1) EMENT →	
10	(m>1) MENT →	
11	(m>1) ENT →	
12	(m>1 and (*S or *T)) ION →	
13	(m>1) OU →	
14	(m>1) ISM →	
15	(m>1) ATE →	
16	(m>1) ITI →	
17	(m>1) OUS →	
18	(m>1) IVE →	
19	(m>1) IZE →	

Step 5

- Step 5a

	Rule	Example
1	(m>1) E →	PROBATE → PROBAT
	(m=1 and not *o) E →	CEASE → CEAS

- Step 5b

	Rule	Example
1	(m > 1 and *d and *L) → single letter	CONTROLL → CONROL

Example

- Input = CHARACTERIZATION
  - ➔ $C(VC)^3$
- Porter algorithm
  - The suffix will not match any of the cases found in step 1.
  - So it will move to step 2.
  - The stem of the word has m > 0 (since m = 3) and ends with "**IZATION**".
  - Hence in step 2, "**IZATION**" will be replaced with "**IZE**".
  - Then the new stem will be **CHARACTERIZE**.
  - Step 3 will not match any of the suffixes and hence will move to step 4.
  - Now m > 1 (since m = 3) and the stem ends with **"IZE"**.
  - So in step 4, **"IZE"** will be deleted (replaced with null).
  - No change will happen to the stem in other steps.
  - Finally the output will be **CHARACTER**.

Quiz

- Input = STEMMING

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

Does stemming help?

- 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!