Introduction to

Information Retrieval

Lecture 2: Preprocessing

Recap of the previous lecture

- Basic inverted indexes:

  - Structure: Dictionary and Postings

BRUTUS	→	1	2	4	11	31	45	173	174
CAESAR	→	1	2	4	5	6	16	57	132	. . .
CALPURNIA	→	2	31	54	101

  - Key step in construction: Sorting

- Boolean query processing

  - Intersection by linear time "merging"

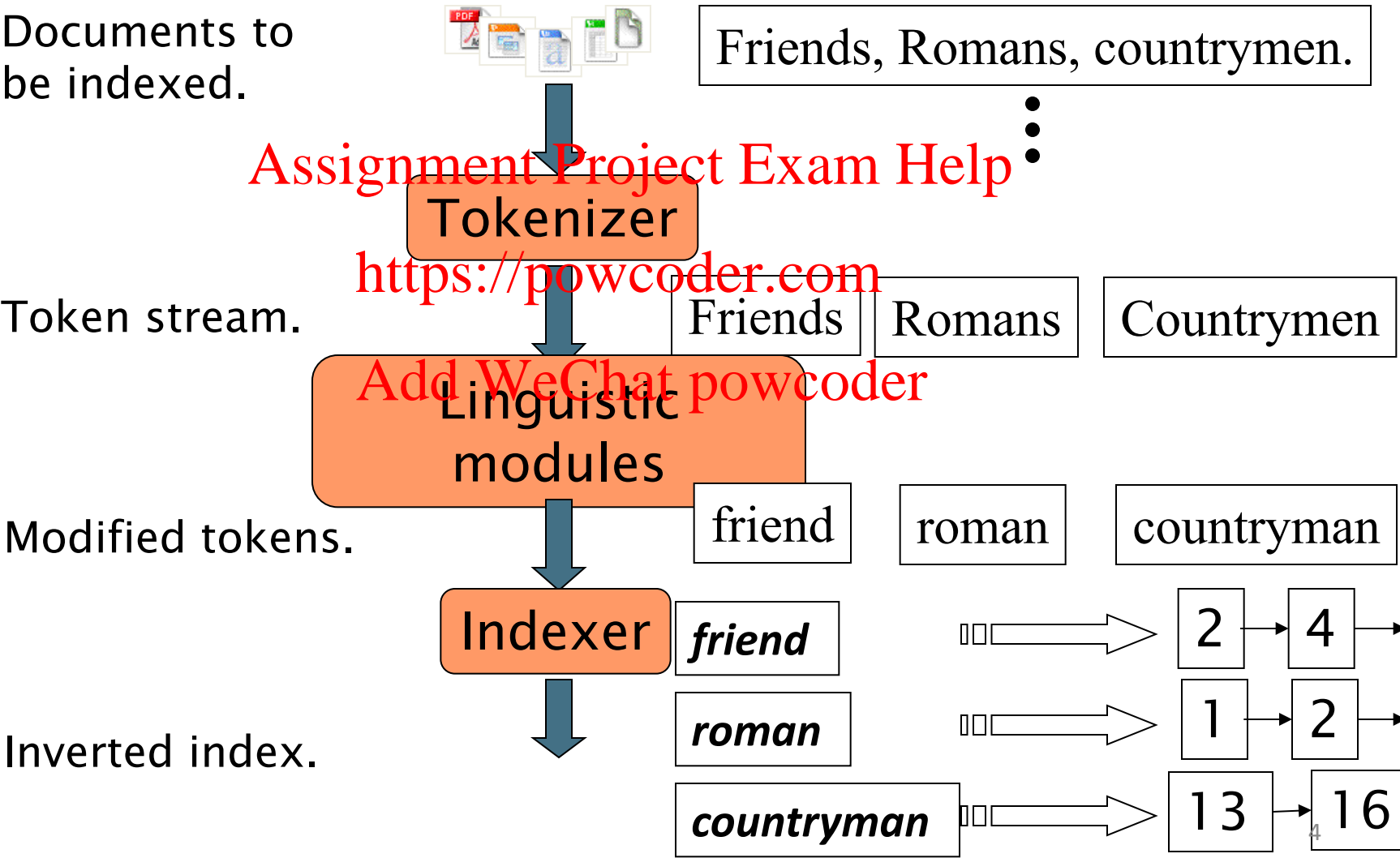 - Optimizations

  - Positional index

Plan for this lecture

Elaborate basic indexing

- Preprocessing to form the term vocabulary
  - Documents
  - Tokenization
  - What *terms* do we put in the index?

Recall the basic indexing pipeline

Documents to be indexed.

Friends, Romans, countrymen.

Tokenizer

Token stream.

| Friends | Romans | Countrymen |

Linguistic modules

Modified tokens.

| friend | roman | countryman |

Indexer

Inverted index.

friend → 2 → 4 →

roman → 1 → 2

countryman → 13 → 16

4

Parsing a document

- What format is it in?

  - pdf/word/excel/html?

- What language is it in?

- What character set is in use?

Each of these is a classification problem, which we will study later in the course.

But these tasks are often done heuristically …

Complications: Format/language

- Documents being indexed can include docs from many different languages
  - A single index may have to contain terms of several languages.
- Sometimes a document or its components can contain multiple languages/formats
  - French email with a German pdf attachment.
- <u>What is a unit document</u>?
  - A file?
  - An email? (Perhaps one of many in an mbox.)
  - An email with 5 attachments?
  - A group of files (PPT or LaTeX as HTML pages)

6

Assignment Project Exam Help

Add WeChat powcoder

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

TOKENS AND TERMS

Tokenization

- <u>Input</u>: "***Friends, Romans and Countrymen***"

- <u>Output</u>: Tokens
  - ***Friends***
  - ***Romans***
  - ***Countrymen***

- A token is an instance of a sequence of characters

- Each such token is now a candidate for an index entry, after <u>further processing</u>
  - Described below

- But what are valid tokens to emit?

Tokenization

- Issues in tokenization:

  - **Finland's capital** →
    Finland? Finlands? Finland's?

  - **Hewlett-Packard** → **Hewlett** and **Packard** as two
    tokens?

    - **state-of-the-art**: break up hyphenated sequence.

    - **co-education**

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

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

        - ■ 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

南京市长江大桥

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
  - Dates/amounts in multiple formats

フォーチュン**500**社は情報不足のため時間あた**$500K(約6,000万円)**

Katakana   Hiragana   Kanji   Romaji
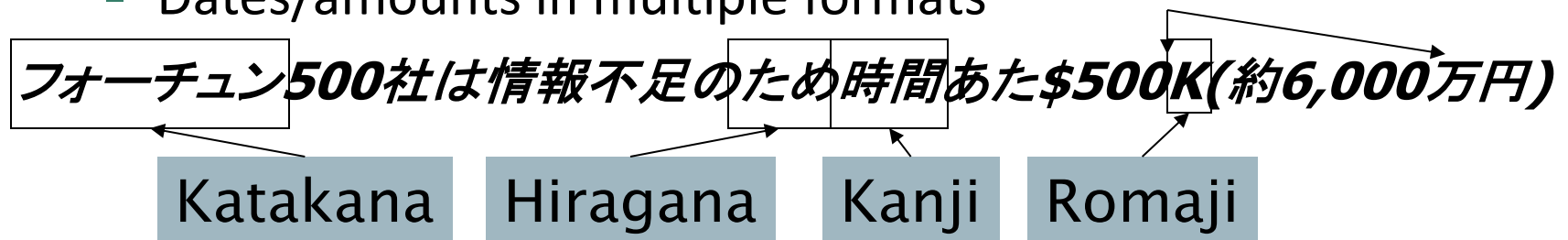
End-user can express query entirely in hiragana!

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**

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

- ← → ← →   ← start

- '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

13

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 (lecture 5) means the space for including stopwords in a system is very small
  - Good query optimization techniques (lecture 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" vs. "flights from London"

14

Normalization to terms

- We 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: criteria
  - 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*
    - *Japanese use of kana vs. Chinese characters*

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

    ***Morgen will ich in MIT** ...*

    > Is this German "mit"?

- Crucial: Need to "normalize" indexed text as well as query terms into the same form

Case folding

- Reduce all letters to lower case

    - exception: upper case in mid-sentence?

        - e.g., *General Motors*

        - *Fed* vs. *fed*

        - *SAIL* vs. *sail*

    - Often best to lower case everything, since users will use lowercase regardless of 'correct' capitalization…

- Google example:

    - Query *C.A.T.*

    - #1 result is for "cat" (well, Lolcats) *not* Caterpillar Inc.

I keepz ur beerz till I getz toona

Normalization to terms

- An alternative to equivalence classing is to do asymmetric expansion

- An example of where this may be useful
  - Enter: **window**   Search: **window, windows**
  - Enter: **windows**   Search: **Windows, windows, window**
  - Enter: **Windows**   Search: **Windows**

- Potentially more powerful, but less efficient

Thesauri and soundex

- Do we 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 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
- More in lectures 3 and 9

Lemmatization

- Reduce inflectional/variant forms to base form

- E.g.,

  - *am, are, is $\rightarrow$ be*

  - *car, cars, car's, cars'* $\rightarrow$ *car*

- *the boy's cars are different colors $\rightarrow$ the boy car be different color*

- Lemmatization implies doing "proper" reduction to dictionary headword form

Stemming

- Reduce terms to their "roots" before indexing

- "Stemming" suggest 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

Other stemmers

- Other stemmers exist, e.g., Lovins stemmer
  - http://www.comp.lancs.ac.uk/computing/research/stemming/general/lovins.htm
  - Single-pass, longest suffix removal (about 250 rules)

- Full morphological analysis – at most modest benefits for retrieval

- Do stemming and other normalizations 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!

Language-specificity

- Many of the above features 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

Dictionary entries – first cut

ensemble.french
時間.japanese
MIT.english
mit.german
guaranteed.english
entries.english
sometimes.english
tokenization.english

These may be grouped by language (or not…).
More on this in ranking/query processing.

25

Resources for today's lecture

- IIR 2

- MG 3.6, 4.3; MIR 7.2

- Porter's stemmer:
  http://www.tartarus.org/~martin/PorterStemmer/