CADT

Cambodia Academy of Digital Technology

“Text Preprocessing, Inverted Index and Boolean Retrieval”

Lecturer

Mr. Natt KORAT

Lecturer-Researcher,
Institute of Digital Research & Innovation (IDRI),
Cambodia Academy of Digital Technology (CADT)

Today pipeline

- Text -> tokens -> remove stop words -> stem/lem -> invert index -> Boolean query
- Each arrow is one part of Lab 2

Tokenization

- Split raw text into tokens: words, usually lowercased
- Example:
  - “Cheap hotel deals” -> [“cheap”, “hotel”, “deals”]
  - “ទិសរ្សិកសាលាស៊ីអេឌីធី” -> ['ទិសរ្សិក', 'សាលា', 'ស៊ី', 'អេឌីធី']
- Tokenizer tools:
  - NLTK: https://www.nltk.org/
  - Khmer-NLTK: https://pypi.org/project/khmer-nltk/
  - Khmer-segmenter: https://pypi.org/project/khmer-segmenter/

What tokenization decides

- Keep number or not?
- Split hyphense?
- Keep case (proper nouns)?
- Every choice trades recall against precision

Stop words

- Very frequent words that carry little content on their own: the, is, of, and, to.
- Remove them shrinks the index and speeds up ranking
- Tools:
  - **English**: you can use nltk toolkit
  - Khmer?
    - https://github.com/back-kh/KSWv2-Stop-Word-Dictionary-for-Khmer-Document.git

When stop-words removal hurts

- Phrase queiries: “to be or not to be”
- Names/titles: “The Who”, “Vitamin A”

Document: I do not like apple

Lowercase: i do not like apple

Tokenize: [i, do, not, like, apple]

Remove stopwords: [like, apple]

Stemming and lemmatization

- Stemming cuts words
- lemmatization understands word
- Both are used to do the text-normalization

Stemming:

- playing -> play
- Played -> play
- studies -> studi
- sonnection -> connect

Lemmatization:

- Running -> run
- Better -> good
- Was -> be
- Studies -> study

Stemming trade-off

- Over-stemming: unrelated words collapse together (e.g. university / universe -> univers)
- Under-stemming: related words stay separated (e.g. ran / running)

The inverted index

- A dictionary: term -> sorted list of document ids that contain it (a postings list)
- Sorted lists are the KEY: they let us merge instead of scan

```
dataset     → [D3]
Khmer       → [D3]
learning    → [D1, D2]
machine     → [D1, D2]
recognition → [D2]
speech      → [D1, D2, D3]
```

Building the index

- For every document:
  - Preprocess every unique terms -> add the document id to its posting list -> sort every list
- Paid once: used for every future query

Building the index

- A common term (e.g. “language”) has a long list
- A rare term (e.g. “pagerank”) has as short, informative list

Term frequency across douments will matter again for ranking (session 3)

Boolean retrieval

- AND, OR, NOT combine posting lists into a single anser set
- *A* AND *B*: documents with both terms
- *A* OR *B*: documents with either
- NOT *A*: document without *A*

Recap

1. *Tokenize, remove stop words, stemming. Each step is a trade-off*
2. *An inverted index stores sorted postings lists*
3. *Boolean queries are answered by merging postings, not scaning text*