Information Retrieval and Web Search

Lecture 02

Lecture by Dr. Inbal Budowski-Tal
Typeset by Steven Karas

2018-03-21
Last edited 20:51:29 2018-03-21

Disclaimer These lecture notes are based on the lecture for the course Information Retrieval and Web Search, taught by Dr. Inbal Budowski-Tal at IDC Herzliyah in the spring semester of 2017/2018. Sections may be based on the lecture slides written by Dr. Inbal Budowski-Tal.

Agenda

- Term vocabulary
- Postings lists

There will not be a lecture next week, the week after that, the week after that, and the week after that. This means the next lecture will be

The first homework will be posted soon. The homeworks are intended to be easy, for example just running an algorithm.

1 Recap

1.1 Inverted Index

This data structure is the most important thing we learned last week. Basically, provides a query index from terms to a set of documents they appear in.

Set intersection Last week, we presented an algorithm for linked list intersection.

Index construction When constructing an inverted index, terms are preprocessed. Preprocessing should normalize terms as much as possible, for example normalizing the case of the terms.

1.2 Boolean search

Searches for the appearances of terms using boolean operators to string queries together.

2 Documents

Previously, we assumed that we know what a document is and that we can machine read a document.

Realistically, we need to parse the documents, which may come from different input formats, and may contain different types of information (computer code, English text, German text, molecular data, genetic sequencing data, etc). Some formats may have different encodings as well (character sets, etc).

Determining these is a classification problem, which may be covered much later in the course[1, Chapter 13].

Format/Language considerations A single index typically only includes terms from several languages. Documents may contain multiple languages. Answering the question of what a document is nontrivial.

3 Terms

Word A delimited string of characters as it appears in the text.

Term A normalized word (case, morphology, spelling, etc); induces an equivalence class of words.

Token An instance of a word or term occurring in a document.

Type An equivalence class of tokens. The number of distinct words in a text.

3.1 Normalization

We need to normalize terms, whether they come from the query or when parsing documents. We usually implicitly define equivalence classes of terms.

The alternative to doing this is to perform asymmetric expansion, which is more powerful, but less efficient.

Normalization in some contexts may differ, for example w.r.t. capitalization rules in German.

Tokenization problems

- Hyphenated phrases, such as "Los Angeles-based company" or "co-education"
- Numbers: dates, amounts, units, etc.
- Chinese: no whitespace, which can cause ambiguous parsing
- Compound words in Dutch, German, Swedish, etc. (e.g. Lebensversicherungsgesellschaft-sangestellter)
- Japanese: multiple alphabets. Kanji, Hiragana, Katakana, and Latin.
- Arabic: affixes and RTL
- Hebrew: optional vowels; RTL and bidirectional text
- Diacritics: accents, umlauts, etc.

3.2 English

3.2.1 Case folding

English words are sometimes capitalized based on upon sentence structure.

As a rule of thumb, it's a good idea to downcase everything as users tend not to correctly capitalize when searching.

3.2.2 Stop words

Stop words are extremely common words that appear to provide very little value and don't contribute much information to the query/document.

Lucene provides an example list of stop words for English.

3.2.3 More equivalence classes

Soundex[1, Chapter 3] provides phonetic equivalence between terms.

Thesauri[1, Chapter 9] provides semantic equivalence between terms.

3.2.4 Lemmatization

Lemmatization reduces inflectional or variant forms to the base form. Clustering is an alternative approach that is more generally applicable.

3.2.5 Stemming

This is a crude heuristic process that chops off the ends of words in the hope of approximating lemmatization.

An example input/output comparing three different stemmers can be found on slide 38.

Porter Algorithm This is the most common algorithm for stemming English. Results suggest that it's at least as good as other stemming options.

Applies some conventions and has 5 phases of reductions. Each phase is applied sequentially. Each phase consists of a set of commands, for example: Delete the final *ement* if what remains is longer than 1 character.

Of the rules in a phase, apply the one that applies to the longest suffix.

3.3 Example:

These are some possible normalizations, but there may be others, and they are subject to system design considerations.

- In June, the dog likes to chase the cat in the barn
  ⇒ june dog like chase cat barn
- Mr. O'Neill thinks that the boys' stories about Chile's capital aren't amusing
  ⇒ mister o'neill think that boy story about chile capital amusing

A full example

a good wine is a
wine that you like

In this example, there are 9 words, 9 tokens, but only 7 types and 7 terms.

4 Skip pointers

The basic idea of skip pointers are to augment linked lists to improve the average case performance of taking the intersection of two such lists. The skip pointer is set for some nodes in the list, and skips over several (or many) elements in the list.

In practice, the ideal number of skip pointers to use is $\sqrt{n}$, as this provides a good tradeoff.

5 Phrase queries

A phrase query is searching for multiple terms as a single phrase without splitting them into tokens. The practical impact of this is that simply storing the documents in the inverted index is no longer sufficient.

5.1 Biword index

Index every consecutive pair of tokens in the document as a phrase. Each of these biwords is now a vocabulary term.

Multi-word phrases can be expanded to take all the biword phrases out of the query phrase. However, a second pass must be done to ensure each of these biword matches are consecutive, and not disparate.

Extensions This can be extended by doing part-of-speech tagging, and building the biwords from nouns/adjectives, and skip over articles/prepositions (or similar combinations).

These extended biwords can be included in the term vocabulary and the index (in addition to the regular biwords).

Drawbacks Using a biword index can lead to false positives, which must be filtered out of the hit list. Also, the index has many more terms ($O(n^2)$), and twice as many appearances.

5.2 Positional index

Stores the position of each appearance of a term in a document in the index for that term. [1]

This allows us to perform proximity searches as well, where we find all documents that contain terms within a token distance of each other. Pseudocode for an algorithm that does this can be found on slide 59.

[1] We will be required to execute a phrase search on a positional index in one of the homework assignments

Biword indexes and positional indexes can be combined, and many biwords are extremely common. For common biwords, it may be beneficial to include them as terms in the index, and do positional search for the rest.

6 Next lecture

Will take place on April 25th.

References

[1] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.