Lect 7: Normalization

Dr. Subrat Kumar Nayak
Associate Professor
Dept. of CSE, ITER, SOADU

Definitions

- Word – A delimited string of characters as it appears in the text.
- Term – A “normalized” word (case, morphology, spelling etc); an equivalence class of words.
- Token – An instance of a word or term occurring in a document.
- Type – The same as a term in most cases: an equivalence class of tokens.

Normalization

- ***Token normalization*** is the process of canonicalizing tokens so that matches occur despite superficial differences in the character sequences of the tokens.
- The most standard way to normalize is to implicitly create ***equivalence classes***, which are normally named after one member of the set.
- Example**:** ***anti-discriminatory*** and ***antidiscriminatory*** are both mapped onto the term **antidiscriminatory**, in both the document text and queries, then searches for one term will retrieve documents that contain either.

Normalization

- An alternative to creating equivalence classes is **to maintain relations between** unnormalized tokens.
- This method can be extended to hand-constructed lists of synonyms such as ***car*** and ***automobile***
- These term relationships can be achieved in two ways.
  - The usual way is to index unnormalized tokens and to maintain a query expansion list of multiple vocabulary entries to consider for a certain query term. A query term is then effectively a disjunction of several postings lists.
  - The alternative is to perform the expansion during index construction. When the document contains automobile, we index it under car as well
- Use of either **of these methods is considerably less efficient** than equivalence classing, as **there are more postings to store and merge**.
  - The **first method** adds a query expansion dictionary and requires more processing at query time, while **the second method** requires more space for storing postings.
- Traditionally, **expanding the space required for the postings** lists was seen as more **disadvantageous**, but with modern storage costs, the increased flexibility that comes from distinct postings lists is appealing.

Normalization

- 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

- deleting hyphens to form a term

anti-discriminatory, antidiscriminatory

- Alternatively: do asymmetric expansion
  - window → window, windows
  - windows → Windows, windows, window
  - Windows (no expansion)
- More powerful, but less efficient

Normalization: Other Languages

- **Diacritics :** Diacritics on characters in English have a fairly marginal status, and we might well want ***cliché*** and ***cliche*** to match, or ***naïve* and *naïve***.
- This can be done by normalizing tokens to remove diacritics. In many other languages, diacritics are a regular part of the writing system and distinguish different sounds.
- **Accents:** Occasionally words are distinguished only by their accents.
- Example: For instance, in Spanish, ***peña*** is 'a cliff', while ***pena*** is 'sorrow'.
- Example2:French ***résumé*** vs. ***resume*.**
- **Umlauts:** 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/** equate all words to a form **without diacritics.**
- ***Tuebingen, Tübingen, 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 identically

Case-folding

- Reduce all letters to lower case
- exception: upper case in mid-sentence?
- e.g., General Motors
- The same task can be done more accurately by a machine learning sequence model which uses more features to make the decision of when to case-fold. This is known as *truecasing*.
- Fed vs. fed
- SAIL vs. sail
- Often best to lower case everything, since users will use lower case regard less of 'correct' capitalization…
- Google example:
- Query C.A.T.
- #1 result is for "cats" (well, Lolcats)not

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)
  - Homonyms: Jaguar, BalckBery or Blackberry
  - 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.