Spelling and Phonetic Correction

Spelling Correction

- When a user types “carot”, she/he may mean “carrot”
  - Spelling correction improves the effectiveness of information retrieval
- Among various alternative correct spellings for a misspelled query, choose the “nearest” one
  - How to define the “distance” between two words?
- When two correctly spelled queries are tied or nearly tied, select the one that is more common
  - More common in the document collection
  - More common among queries by other users
  - Classroom discussion: can you come up a new idea of “more common”?

Spelling Correction Functionalities

- On the query "carot", retrieve documents containing carot and any spell-corrected version of carot such as carrot and tarot
- Conduct query spelling correction only when the query term (carot) is not in the dictionary
- Conduct query spelling correction only when the query term (carot) returns fewer than a preset number of documents
- When the query term (carot) returns fewer than a preset number of documents, make a spelling suggestion to the user
  - "Do you mean carrot or tarot?"

Isolated-term or Context-sensitive

- Isolated-term correction corrects a single query term at a time, even when we have a multiple-term query
  - Edit distance
  - K-gram overlap
- Context-sensitive correction considers the whole multiple-term query in correction

Edit Distance (Levenshtein Distance)

- Given two character strings s1 and s2, the edit distance between them is the minimum number of edit operations required to transform s1 to s2
  - Edit operations: insertion, deletion, replacement
  - Different operations may carry different weights
  - Changing from a character to different other characters may carry different weights
- Classroom discussion: is the edit distance symmetric? That is, is the edit distance from s1 to s2 equal to the edit distance from s2 to s1? Why?

Edit Distance Computation

- Can be computed in time O(|s1| x |s2|) using dynamic programming

```
EDITDISTANCE(s1, s2)
 1  int m[|s1|, |s2|] = 0
 2  for i ← 1 to |s1|
 3  do m[i, 0] = i
 4  for j ← 1 to |s2|
 5  do m[0, j] = j
 6  for i ← 1 to |s1|
 7  do for j ← 1 to |s2|
 8      do m[i, j] = min{m[i − 1, j − 1] + if (s1[i] = s2[j]) then 0 else 1fi,
 9                       m[i − 1, j] + 1,
10                       m[i, j − 1] + 1}
11  return m[|s1|, |s2|]
```

Substitute a character

Inserting a character into s1

Inserting a character into s2

Example

		f		a		s		t	
	0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
c	**1** **1**	*1* *2*	*2* *1*	**2** **2**	3 **2**	**3** **3**	4 **3**	**4** **4**	5 **4**
a	**2** **2**	**2** 3	**2** **2**	*1* *3*	3 *1*	*3* *2*	4 *2*	4 **3**	5 **3**
t	**3** **3**	**3** 4	**3** **3**	3 4	**2** **2**	*2* *3*	3 *2*	*2* *3*	4 *2*
s	**4** **4**	**4** 5	**4** **4**	4 5	**3** **3**	**2** 4	3 **2**	*3* *3*	3 *3*

Spelling Correction as NN Search

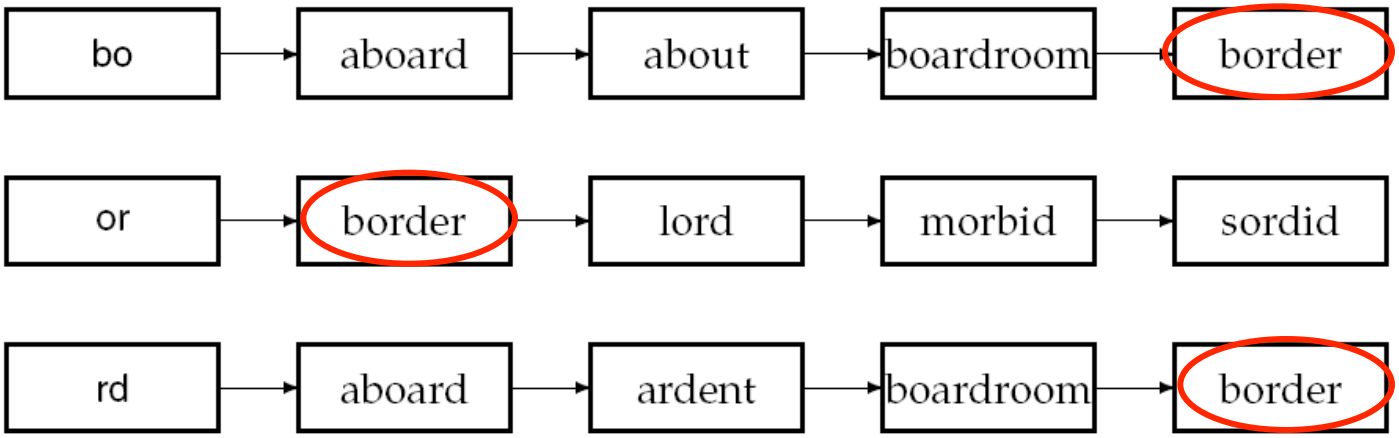
- Given a set S of terms in the vocabulary and a query q, find the strings in S of least edit distance from q
- A naïve method
  - Compute the edit distance from q to each string in S
  - Select the one of the minimum distance
  - Very costly! $O(|q| \times \Sigma_{s \in S} |s|)$

Some Heuristic Approaches

- Consider only dictionary terms beginning with the same letter
- Consider the set of all rotations of the query string q
  - Use a permuterm index by omitting the end-of-word symbol $
  - If q = mase, we consider sema, emas, …
- To find mare and mane that are close to mase, for each rotation, we omit a suffix of up to l characters, where l is a parameter

K-Gram Indices

- Heuristic: if two words have many common k-grams, they may be similar to each other
  - If there are multiple candidates, find the one with the least edit distance
- Example: query "bord"
  - Suggest "border"



K-gram Algorithm

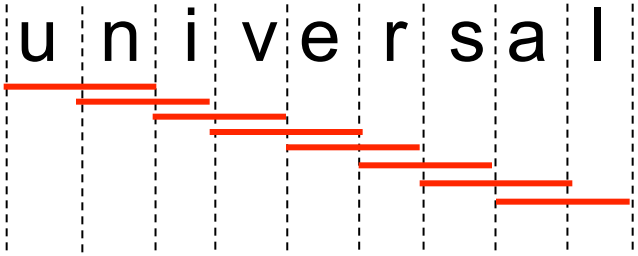
- Search k-gram postings in the merge-sort way
- When a term t is met, compute the Jaccard coefficient between q and t on the fly
  - For sets A and B, the Jaccard coefficient is $|A \cap B| / |A \cup B| = |A \cap B| / (|A| + |B| - |A \cap B|)$
  - For term "boardroom" and query "bord", since "boardroom" appears in 2 posting lists of 2-grams of "bord", the Jaccard coefficient is 2 / (8 + 3 – 3) = 3 / 8
    - "boardroom" and "bord" have 8 and 3 2-grams, respectively
  - If the Jaccard coefficient passes a threshold, add t into the candidate set

Variable length grams

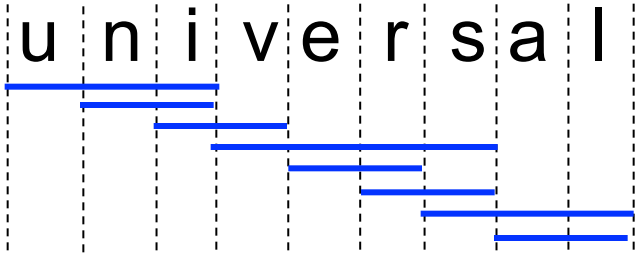
- Large k values → shorter posting lists and smaller number of common grams between similar strings
- Variable length grams
  - Removing k-grams which are common for many words
  - Reducing false positives
  - Reducing index size

K-gram versus Vgram

- Fixed-length 2-grams



- Variable-length grams



[2,4]-gram dictionary

ni
ivr
sal
uni
vers

Using Vgram

- How to determine the terms in the dictionary?
  - Using the frequency information
- How to use vgram to correct spelling errors?
  - Using vgram matches/mismatches to compute the upper bound of edit distance
- Details in [Chen et al., VLDB'07]
  - Not required in exam

Context-Sensitive Correction

- Query "flew form Vancouver"
  - Should be corrected to "flew **from** Vancouver"
- Using frequent combinations of words in query logs
  - Using bi-words statistics

Phonetic Correction

- How to correct misspelling caused by typing a query that sounds like the target term?
  - Example: Hermann and Herman
- "phonetic hashing" – similar-sounding terms are hashed to the same value
  - First developed in international police departments in early 20th century

Soundex Algorithm

- Idea
  - Vowels are viewed as interchangeable in transcribing names
  - Consonants with similar sounds (e.g., D and T) are put in equivalence classes → related names often have the same soundex codes
- Algorithm
  - Turn every term to be indexed into a four-character reduced form, build an inverted index from these reduced forms to the original terms called the soundex index
  - Do the same with the query terms
  - Search the soundex index

Four-Character Code

- The first character is a letter of the alphbet and the other three are digits between 0 and 9
- Algorithm
  - Retain the first letter of the term
  - Change all occurrences of the following letters to '0': A, E, I, O U, H, W, and Y
  - Change letters to digits as follows
    - B, F, P, V → 1
    - C, G, J, K, Q, S, X, Z → 2
    - D, T → 3
    - L → 4
    - M, N → 5
    - R → 6
  - Repeatedly remove one out of each pair of consecutive identical digits
  - Remove all zeros from the resulting string, pad the resulting string with trailing zeros and return the first four positions, which will consist of a letter followed by three digits
- Example: Hermann → H655, Herman → H655, matched!

Summary

- Spelling correction is important in IR systems
- NN search using edit-distance
- K-gram and vgram
- Phonetic correction using soundex algorithms

To-do List

- Read Section 6.2.2