Wildcard Queries
Inverted Indexes

Query “Brutus” AND “Calpurnia”

<table>
<thead>
<tr>
<th>Brutus</th>
<th>→</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>11</th>
<th>31</th>
<th>45</th>
<th>173</th>
<th>174</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caesar</td>
<td>→</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>16</td>
<td>57</td>
<td>132</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>→</td>
<td>2</td>
<td>31</td>
<td>54</td>
<td>101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Dictionary     Postings
### Vocabulary Lookup

- Given an inverted index and a query, we need to determine whether each query term exists in the vocabulary
  - If so, identify the pointer to the corresponding postings
- Hashing or search trees?
  - How many keys (terms)?
  - Is the number of keys static or changing a lot?
  - Operations on the keys, insertions only or insertions + deletions?
  - Relative frequencies of key accesses?
Hashing

- No easy way to find minor variants of a query term
  - Minor variants could be hashed to very different buckets
- Cannot find all terms with the same prefix
- For web search, the vocabulary size keeps growing
  - A hash function may become insufficient after several years
Search Trees

- Easy to find all terms with the same prefix
- Balancing search trees
  - Logarithmic search time
  - Cost: rebalancing
B-trees

- Every internal node has a number of children in interval \([a, b]\)
- Good for disk-based data storage
When Are Wildcard Queries Useful?

- A user is uncertain about the spelling of a query term
  - S*dney → uncertain about Sydney or Sidney
- A user is aware of multiple variants of spelling a term and (consciously) seeks documents containing any of the variants
  - Color versus colour
- A user searches documents containing variants of a term that would be caught by stemming, but is unsure whether the search engine conducts stemming
  - “judicia*” → judicial versus judiciary
- A user is uncertain about the correct rendition of a foreign word or phrase
  - “Universit* Stuttgart”
Trailing Wildcard Queries

- A trailing wildcard query has only one * symbol at the end of the search string
  - Example: mon*
- Trailing wildcard queries can be answered efficiently using a search tree
  - Walk down the tree following the symbols m, o, and n in turn
  - Enumerate the set $W$ of terms in the dictionary with the prefix mon
  - Use $|W|$ lookups on the inverted index to retrieve all documents containing any term in $W$
Leading Wildcard Queries

• A leading wildcard query has only one * symbol at the beginning of the query
  – Example: *mon

• A leading wildcard query can be answered efficiently using a reverse search tree
  – Each root-to-leaf path corresponds to a term in the dictionary written backwards
  – The term “lemon” is represented by a path root-n-o-m-e-l
A Little More General Case

• How to answer queries containing only one * symbol but can be in any position
  – Example: se*mon?
• Rewrite the query to se* AND *mon
• Use two search trees
  – A search tree to answer query se*, find the set $W$ of terms
  – A reverse search tree to answer query *mon, find the set $R$ of terms
  – $W \cap R$ is the set of terms satisfying the query
General Wildcard Queries

- A general wildcard query can have any number of * symbol at any position
- Framework
  - Rewrite a given wildcard query q as a Boolean query Q on a specially constructed index, such that the answer to Q is a superset of the set of vocabulary terms matching q
  - Check each term in the answer to Q against q, discarding those vocabulary terms that do not match q
- Two methods: permuterm indexes and k-gram indexes
Permuterm Indexes

- Use a special symbol $ to mark the end of a term
  - Term “hello” is represented as hello$
- A permuterm index contains various rotations of each term augmented with $ all linked to the original vocabulary term
  - The permuterm vocabulary: the set of rotated terms in the permuterm index
Query Answering – One * Symbol

• Rotate a wildcard query so that the * symbol appears at the end of the string
  – Example: rotate m*n to n$m*

• Look up the string in the permuterm index
  – Find terms n$ma and n$moro → man and moron are the answers
Query Answering – Multiple *’s

- Example query: q = fi*mo*er
- Conduct query Q = er$fi
- Check each term returned from Q against q, only search the inverted index for those terms satisfying q
- Cost: the permuterm index is quite large since it contains all rotations of each term
  – On average 10 times for English documents
Discussion

• For query $q = f^*mo^*er$, we can run queries $Q1 = er$f$ and Q2=mo and obtain the intersection of the answers
  – Is the method good? Why?
• For query $q = b^*etro^*t$
  – Run query $Q1 = t$b$
  – Run query $Q2 = etro^*$
  – Which way is better? Why?
K-gram Indexes

- A k-gram is a sequence of k characters
  - Use symbol $ to denote the beginning and end of a term
  - 3-grams of “castle”: $ca, cas, ast, stl, tle, le$
- A k-gram index contains all k-grams that occur in any term in the vocabulary
- Each postings list points from a k-gram to all vocabulary terms containing that k-gram
Query Answering

- Example query re*ve
- Run the Boolean query $re \text{ AND } ve$
- False positive may happen
  - Query red*
  - Run Boolean query $re \text{ AND } red$
  - Term \text{ retired} is an answer
- Postfiltering: check terms returned from the Boolean query against the original query
More on Wildcard Queries

- Wildcard queries can be quite expensive
  - The added lookups in the special index, filtering
- Most commonly, the capability of wildcard queries is hidden behind an “advanced query” interface
  - Most users never use
  - Do not encourage users to invoke wildcard queries when they do not require it
  - Reduce the processing load on a search engine
Summary

- Vocabulary lookup: hashing versus search trees
- Wildcard queries are powerful in search
  - Permuterm indexes
  - K-gram indexes