Experiments on Approximate String Matching Algorithms

Qi Li, Jiawei Wu

EECS 405

Introduction

- Approximate String Search
  - Q-gram
  - Distance functions
- VGRAMS
  - Chen Li, et.al, VGRAM: Improving Performance of Approximate Queries on String Collections Using Variable Length Grams, VLDB 2007.
- Set-Similarity Joins
  - Arasu, et. al, Efficient Exact Set-Similarity Joins, VLDB 2006
  - Talking about SSjoin and PartEnum
- Merging and Filtering Algorithms
  - Chen Li, et. al., "Efficient Merging and Filtering Algorithms for Approximate String Searches", ICDE 2008

Overview

1. VGRAM and Exact Set-Similarity Join: PartEnum
    a. VGRAM
    b. PartEnum
    c. PartEnum with VGRAM

2. Merging and Filtering Algorithms
    a. ScanCount
    b. MergeSkip
    c. DivideSkip
    d. Length filter
    e. Charsum filter

VGRAM

- Description
  - Variable gram length for q-gram

- Advantages
  - Overcoming q length dilemma
    - Great for suitable, large, datasets
    - Improved speed, size
- Disadvantages
  - Storage is more verbose
    - Speed and Time for maintaining structures
  - Volatile grams
    - Fewer choice and more overhead on distance functions

Implementations of VGRAM

- Gram Dictionary
  - Trie and inverse trie
  - Hashmap

- Gram Pruning
  - Frequencies analysis
  - Keeping long grams that has high presence compared to its parent (eg. “tion” as to “tio”)
  - Keeping short grams that is not of high enough frequency
  - Perform recursively to improve gram dictionary quality

- Distances
  - NAG Vectors: *NAG(s,k)*
    - Maximum # of affected grams with *k* edit operations
    - Not computed exactly;
    - Preferably bounded as small as possible; rooms for optimization

PartEnum

- Description
  - ➔ Set-Similarity join (SSJoin)
    - Traditionally probabilistic
  - ➔ Signature based, Partitioning + Enumeration Hybrid method.
  - ➔ Tailored towards a sweetspot between speed and correctness.

- Advantages
  - ➔ Exact SSJoin
    - Produces very few false positives
  - ➔ Good control parameters adapting to different datasets
    - Part & Enum ratio, gram lengths, edit distance, etc.

PartEnum with VGRAM

- Idea
  - ➔ Use VGRAMs as atomic elements in PartEnum
  - ➔ Reduced size and increased efficiency
  - ➔ Use *NAG(s,k)* to compare with VGRAM hamming distance.

- Implementation
  - ➔ VGRAM dictionary construction
  - ➔ Transform strings set into VGRAMs set
    - String ➔ vector(vgrams,locations)
  - ➔ Signature generation
  - ➔ Distance eval
    - $\mathit{Hamming}(\mathrm{VGRAMS}(s_1), \mathrm{VGRAMS}(s_2)) <= \mathit{NAG}(s_2,k) + \mathit{NAG}(s_1,k)$

PartEnum Results

- Current Datasets
  - ➔ Movie names from IMDB (english words, nonuniform length, difficult to sanitize)
  - ➔ Protein gene code (more random)
  - ➔ Gene code (size 20000) PartEnum sample results:

```
Parameters: ed=2 q=2 n1=2 n2=5
GramsID Size: 20249
Signature Size: 2274824
Build and save time: 0.911453
Search time: 0.00702548
Strings similar to dsfnk: dsn1 dstn dstyk ifnk sfn
```

```
Parameters: ed=2 q=2 n1=3 n2=5
GramsID Size: 20249
Signature Size: 3263224
Build and save time: 1.48798
Search time: 0.00726104
Strings similar to dsfnk: dsn1 dstn dstyk ifnk sfn
```

- VGRAM-PartEnum
  - ➔ Ongoing effort on top of legacy Flamingo library
  - ➔ Full implementation not yet functional, near completion;
  - ➔ Freedom of implementation choices

Merging and Filtering Algorithms

Description:

- ➔ Algorithms
- ➔ Filters

Advantage:

- ➔ ScanCount, MergeSkip and DivideSkip is faster on solving T-occurrence problems
- ➔ Can use multiple kinds of similarity functions
- ➔ Can improve performance by filters

Disadvantage:

- ➔ Required fixed length of q-gram.
- ➔ With edit distance threshold increasing, query time increases significantly.

Merging Algorithms

- HeapMerger - pop list-heads to a heap, push from heap and count occurrences of each element
- ScanCount - uses one counter for every possible stringid (scan count array), traverses inverted lists and increments counts in the array
- MergeOpt - separate long lists from short lists, for short lists use heap merge, for long lists do binary search on candidates from short lists
- MergeSkip - like heapmerger, but uses the merging-threshold to skip elements on the lists
- DivideSkip - combines MergeOpt and MergeSkip, skipping is used for the short lists, binary search is done on the long lists

Merging Algorithms

```
Gram Length = 3    edit distance threshold = 1 (10 out of 20183, Protein gene code)
----- MergeOpt ----
Average Query Time: 4.58e-05 ms

----- HeapMerge ----
Average Query Time: 5.69e-05 ms

----- ScanCount ----
Average Query Time: 4.36e-05 ms
----------------------
```

```
----- MergeSkip ----
Average Query Time: 3.54e-05 ms

----- DivideSkip ----
Average Query Time: 1.81e-05 ms
```

```
Gram Length = 3    edit distance threshold = 2 (10 out of 20183, Protein gene code)
----- MergeOpt -----
Average Query Time: 0.0001848 ms

----- HeapMerge ----
Average Query Time: 0.0003358 ms

----- ScanCount ----
Average Query Time: 0.0001324 ms
----------------------
```

```
---- MergeSkip ----
Average Query Time: 0.0001337 ms

----- DivideSkip ----
Average Query Time: 0.0001257 ms
```

Filtering Algorithms

- Length Filter (intermediate level)
  - Partition the data strings based on their length
  - Partition the string collection
- Charsum Filter (leaf node)
  - Partition the data strings based on their charsums (sum of characters in the string).
  - Partition the inverted lists
- Continue on doing data test with filters adding to merging algorithms

Conclusions

1. Scancount, MergeSkip and DivideSkip is more efficient than MergeOpt and Heap.

2. DivideSkip always gives the best result.

3. When edit distance threshold grows, the time consumed by all algorithms increases significantly (around 10^(threshold growth)).

Conclusions

- PartEnum
  - Very adaptable overall algorithm for lots of pertinent tasks
    - Approximate strings; Set-joins; etc.
  - Signature-based scheme able to achieve good speed
  - Hybrid partitioning and enumerating provides good controls

- VGRAM
  - Good idea with a large variety of different implementation decisions
  - Adaptable to any variants using edit distances.

- PartEnum with VGRAM
  - Larger construction overhead
    - NAG Calculations
    - Depends on optimization
  - VGRAM Speedup on query time; better signature generations
  - VGRAM index sizing improvement
    - Highly benefited in case of the right datasets
    - Human languages, etc.

Q&A