This thesis focuses on the problem of text retrieval allowing errors, also called “approximate” string matching. The problem is to find a pattern in a text, where the pattern and the text may have “errors”. This problem has received a lot of attention in recent years because of its applications in many areas, such as information retrieval, computational biology and signal processing, to name a few.

The aim of this work is the development and analysis of novel algorithms to deal with the problem under various conditions, as well as a better understanding of the problem itself and its statistical behavior. Although our results are valid in many different areas, we focus our attention on typical text searching for information retrieval applications. This makes some ranges of values for the parameters of the problem more interesting than others.

We have divided this presentation in two parts. The first one deals with on-line approximate string matching, i.e. when there is no time or space to preprocess the text. These algorithms are the core of off-line algorithms as well. On-line searching is the area of the problem where better algorithms existed. We have obtained new bounds for the probability of an approximate match of a pattern in a random text, and used these results to analyze many old and new algorithms. We have developed new algorithms for this problem which are currently among the fastest known ones, being even the fastest algorithms for almost all the interesting cases of typical text searching. Finally, we extended our results to the simultaneous search of multiple patterns, obtaining the best existing algorithms when a moderate number of them is sought (less than 100, approximately).

The second part of this thesis addresses indexed approximate string matching, i.e. when we are able to build an index for the text beforehand, to speed up the search later. The ultimate index for approximate string matching is yet to appear and the current development is rather immature, but we have made progress regarding new algorithms as well as better understanding of the problem. For the restricted case of indices able to retrieve only whole words on natural language text, we have obtained new analytical results on their asymptotic complexity, which allowed us to develop an index that is sublinear in space and query time simultaneously, something that did not exist before. For this kind of index we also presented improved search algorithms. For general indices able to find any occurrence (not only words), we have developed new indexing schemes which are a tradeoff between efficiency and space requirements. Also, inspired in on-line techniques, we have proposed a hybrid between existing indexing schemes and obtained very promising results.

It is worth to mention that in almost all cases we have complemented the development of the new algorithms with their worst-case and average-case complexity analysis, as well as a thorough experimental validation and comparison against the best previous work we were aware of.
As a whole, we believe that this work constitutes a valuable contribution to the development and understanding of the problem of approximate text searching.

Contents

1 Introduction 1
  1.1 History and Motivation 1
  1.2 The Problem 4
  1.3 Overview of the Thesis 5
    1.3.1 General Part 5
    1.3.2 On-line Searching 6
    1.3.3 Indexed Searching 7

2 Notation and Basic Concepts 9
  2.1 Definition of the Problem 10
  2.2 Dynamic Programming Algorithm 12
  2.3 A Graph Reformulation 15
  2.4 A Reformulation Based on Automata 15
  2.5 Filtering Algorithms 16
  2.6 Bit-Parallelism 18
  2.7 Suffix Trees and DAWGs 19
  2.8 Suffix Automata 21
  2.9 Natural Language and Its Statistics 21
  2.10 Inverted Files or Inverted Indices 24
  2.11 Suffix Arrays 26

3 Related Work and Our Contributions 29
  3.1 On-line Searching 29
    3.1.1 Taking Advantage of the Dynamic Programming Matrix 31
      3.1.1.1 Improving the Worst Case 31
      3.1.1.2 Improving the Average Case 33
    3.1.2 Searching with a Deterministic Automaton 34
    3.1.3 Filtering 36
      3.1.3.1 Moderate Patterns 36
      3.1.3.2 Very Long Patterns 39
    3.1.4 Bit-Parallel Algorithms 41
      3.1.4.1 Parallelizing Non-deterministic Automata 42
      3.1.4.2 Parallelizing the Dynamic Programming Matrix 43
  3.2 Variants on the On-line Problem 44
    3.2.1 Extended Patterns and Different Costs 44
    3.2.2 Multiple Patterns 45
  3.3 Indexed Searching 47
3.3.1 Word-Retrieving Indices 48
3.3.2 Simulating Text Traversal 50
  3.3.2.1 Minimum Redundancy 50
  3.3.2.2 Depth-First Search 51
3.3.3 Filtration Indices 52
  3.3.3.1 All \( q \)-grams on the Text 53
  3.3.3.2 Sampling the Text 54

4 Basic Tools 55
  4.1 Statistics of the Problem 55
    4.1.1 Probability of Matching 55
      4.1.1.1 An Upper Bound 56
      4.1.1.2 A Lower Bound 58
      4.1.1.3 Experimental Verification 58
    4.1.2 Active Columns 59
  4.2 Partitioning Lemmas 61
  4.3 Hierarchical Verification 62
    4.3.1 Pattern Splitting 62
    4.3.2 Superimposed Searching 65

I On-line Searching 67

5 A Bit-Parallel Algorithm 70
  5.1 A New Parallelization Technique 70
  5.2 A Linear Algorithm for Small Patterns 72
    5.2.1 A Simple Filter 75
    5.2.2 The Code 75
  5.3 Handling Extended Patterns 75
  5.4 Partitioning Large Automata 77
  5.5 Partitioning the Pattern 79
  5.6 Superimposing the Subpatterns 80
  5.7 Analysis and Optimization 81
    5.7.1 The Simple Algorithm 82
    5.7.2 Automaton Partitioning 82
      5.7.2.1 Search Cost 82
      5.7.2.2 Practical Tuning 83
      5.7.2.3 Improving Register Usage 85
    5.7.3 Pattern Partitioning 87
      5.7.3.1 Search Cost 87
      5.7.3.2 Optimal Selection for \( j \) 88
    5.7.4 Superimposition 89
      5.7.4.1 Optimizing the Amount of Superimposition 89
      5.7.4.2 Optimal Grouping and Aligning 91
5.8 Combining All the Techniques
   5.8.1 A Theoretical Approach
   5.8.2 A Practical Heuristic and a Searching Software
5.9 Experimental Comparison

6 Filtering and Automata Algorithms
6.1 Reduction to Exact Search
   6.1.1 The Original Algorithm
   6.1.2 Applying Hierarchical Verification
   6.1.3 Optimizing the Partition
   6.1.4 Experimental Comparison
   6.1.5 Extensions
6.2 A Counting Filter
   6.2.1 A Simple Counting Filter
   6.2.2 Analysis
      6.2.2.1 Exact Analysis
      6.2.2.2 A Simpler Formula
   6.2.3 A Sampling Technique
   6.2.4 Experiments
      6.2.4.1 Maximum Error Ratio
      6.2.4.2 Comparison among Algorithms
6.3 A Suffix Automaton Approach
   6.3.1 Adapting the NFA
   6.3.2 The Search Algorithm
   6.3.3 Analysis
   6.3.4 Experimental Results
6.4 A Partial Deterministic Automaton
   6.4.1 Lazy Automata
   6.4.2 The Algorithm
   6.4.3 Analysis
   6.4.4 Experiments
      6.4.4.1 Automaton Growth
      6.4.4.2 Comparison Against Other Algorithms
   6.4.5 Working with Limited Memory
      6.4.5.1 Victim Selection
      6.4.5.2 Victim Replacement

7 Multiple Patterns
7.1 Superimposed Automata
   7.1.1 Handling Longer Patterns
7.2 Partitioning into Exact Searching
7.3 A Counting Filter
7.4 Analysis
II Indexed Searching

8 Word-Retrieving Indices

8.1 Vocabulary Statistics
  8.1.1 Combining Heaps’ and Zipf’s Laws
  8.1.2 Vocabulary Matching
  8.1.3 Experiments

8.2 Full Inverted Indices
  8.2.1 Retrieval Times
  8.2.2 Experimental Results
  8.2.3 Differential Pointers

8.3 Block Addressing Inverted Indices
  8.3.1 Average Space-Time Trade-offs
    8.3.1.1 Query Time Complexity
    8.3.1.2 Space Complexity
    8.3.1.3 Combined Sublinearity
  8.3.2 Analyzing the Web
  8.3.3 Experimental Validation
    8.3.3.1 Fixed Block Size
    8.3.3.2 Fixed Number of Blocks
    8.3.3.3 Sublinear Space and Time

8.4 Improving the Search Algorithms
  8.4.1 Vocabulary Search
    8.4.1.1 Searching in General Metric Spaces
    8.4.1.2 The Vocabulary as a Metric Space
    8.4.1.3 Experimental Results
  8.4.2 Block Search

9 Sequence-Retrieving Indices

9.1 An Index Based on Sampling
  9.1.1 Indexing Text Substrings
  9.1.2 Analysis
    9.1.2.1 Building the Index
    9.1.2.2 Index Space
    9.1.2.3 Retrieval Time
  9.1.3 Experiments

9.2 An Index Based on Suffix Trees
  9.2.1 Using the Bit-parallel Automaton
10 Conclusions
10.1 Results Obtained 197
11.5 Future Work 198
10.3 Open Questions 199

Bibliography 201

Author’s Address

Gonzalo Navarro
Dept. of Computer Science
University of Chile
Blanco Encalada 2120
Santiago, Chile

Email: gnavarro@dcc.uchile.cl
Phone: +56-2-6892736
Fax: +56-2-6895531
Web: http://www.dcc.uchile.cl/~gnavarro
(thesis postscript available from there)