Extracting Key-Substring-Group Features for Text Classification

Dell Zhang
SCSIS
Birkbeck, University of London
London WC1E 7HX, UK
dell.z@ieee.org

Wee Sun Lee
Department of Computer Science and
Singapore-MIT Alliance
National University of Singapore
Singapore 117543
leews@comp.nus.edu.sg

ABSTRACT

In many text classification applications, it is appealing to take every document as a *string of characters* rather than a *bag of words*. Previous research studies in this area mostly focused on different variants of *generative* Markov chain models. Although *discriminative* machine learning methods like Support Vector Machine (SVM) have been quite successful in text classification with word features, it is neither effective nor efficient to apply them straightforwardly taking all substrings in the corpus as features. In this paper, we propose to partition all substrings into statistical *equivalence groups*, and then pick those groups which are important (in the statistical sense) as features (named *key-substring-group* features) for text classification. In particular, we propose a *suffix tree* based algorithm that can extract such features in *linear* time (with respect to the total number of characters in the corpus). Our experiments on English, Chinese and Greek datasets show that SVM with key-substring-group features can achieve outstanding performance for various text classification tasks.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.2.8 [**Database Management**]: Database Applications—*Data mining*; I.2.6 [**Artificial Intelligence**]: Learning; I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation; Feature evaluation and selection*

General Terms

Algorithms, Experimentation.

Keywords

Text Mining, Text Classification, Machine Learning, Feature Extraction, Suffix Tree.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
KDD'06, August 20–23, 2006, Philadelphia, Pennsylvania, USA.
Copyright 2006 ACM 1-59593-339-5/06/0008 ...$5.00.

1. INTRODUCTION

Text classification (or categorization) [27, 54] via *machine learning* [44] is a fundamental technique for information organization and management.

Traditionally machine learning methods for text classification treat every document as a *bag of words* [27, 40, 54]. However, in many text classification applications, it is appealing to take every document as a *string of characters* [63]. The string-based approaches to text classification have at least the following potential advantages.

- The sub-word features (e.g., morphological variants) and super-word features (e.g., phrasal effects) can be exploited automatically. This is particularly helpful to non-topical text classification applications, such as text genre classification [33, 38, 47, 57] and text authorship classification [25, 47, 57].

- The messy and rather artificial problem of defining word boundaries can be avoided. This is particularly attractive to text classification for oriental languages (Chinese, Japanese, etc.) [1, 23, 47], because many oriental languages do not utilize word delimiters as whitespace characters in western languages (English, French, etc.). Although it is possible to perform automatic word segmentation to separate words in oriental language documents, errors are likely to be brought to the following stage of text classification because no automatic word segmentation technique has perfect accuracy. Therefore it is desirable to avoid this intermediate step. As Vapnik has pointed out, "When solving a problem of interest, do not solve a more general problem as an intermediate step." [61].

- The non-alphabetical features (e.g., punctuation characters) can be taken into account. This is particularly useful to text classification for spam filtering, because many spam emails try to disguise themselves using non-alphabetical characters. For example, spam emails may use "V1a.gr.a" instead of "Viagra" to cheat the spam filters.

- Different types of documents (e.g, Web pages, emails, chat history) can be dealt with in a uniform way. This is probably beneficial if we want to add text classification function to "desktop search" tools (such as those from Google, Microsoft and Yahoo!) so that all documents in a PC can be categorized automatically.

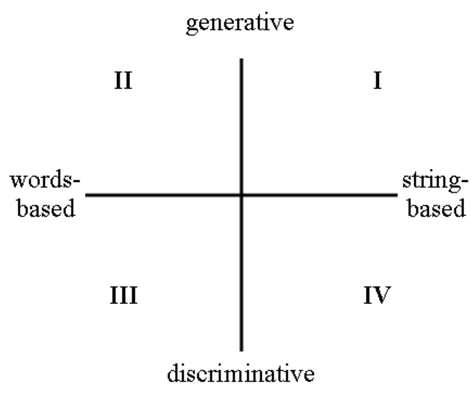

Figure 1: Different kinds of text classification.

Previous research studies on string-based text classification mostly focused on different variants of *generative* Markov chain models [42]. However, *generative* learning methods are often inferior to *discriminative* learning methods, in terms of classification performance.

Although *discriminative* machine learning methods like Support Vector Machine (SVM) [15, 24, 31, 53, 55] and AdaBoost [52, 51] have been quite successful in text classification with word features, it is neither effective nor efficient to apply them straightforwardly taking all substrings in the corpus as features.

In this paper, we propose to partition all substrings into statistical *equivalence groups*, and then pick those groups which are important (in the statistical sense) as features (named *key-substring-group* features) for text classification. In particular, we propose a *suffix tree* [21] based algorithm that can extract such features in *linear* time (with respect to the total number of characters in the corpus). Thus discriminative machine learning methods are enabled to work on string representations of documents. In other words, this paper focuses on quadrant IV in Figure 1. Our experiments on English, Chinese and Greek datasets show that SVM with key-substring-group features can achieve outstanding performance for various text classification tasks.

In the rest of this paper, we first survey related works (in section 2), then propose our idea and algorithm for extracting key-substring-group features (in section 3), later present experimental results to show the effectiveness and efficiency (in section 4), finally make concluding remarks (in section 5).

2. RELATED WORKS

2.1 The Generative Approach

The generative approach to string-based text classification assumes that each document is generated by a Markov chain model [42] according to its class. For each class C, a Markov chain model M_C is constructed using the collection of training documents, then a test document d is classified to $\arg\max_C \Pr[C|d]$, where $\Pr[C|d]$ can be computed using the Bayes's rule and $\Pr[d|C]$ given by M_C.

Markov chain models could be in fixed order/memory or variable order/memory.

Markov chain models in fixed order n are usually called n-gram language models [20, 50] in the field of natural language processing. The n-gram language modeling technique has been used extensively in speech recognition for decades [32], and recently in information retrieval [48, 36, 70]. Most occurrences of the term "n-gram" in the natural language processing literature refer to a continuous segment of n words, but in this paper we focus on character-level n-grams, i.e., character strings of fixed length n. Peng et al. have tried character-level n-gram modeling for various text classification tasks [47]. To achieve a decent performance, one needs to choose an appropriate order n and employ a good smoothing method [71, 10].

Markov chain models in variable order adjust the memory length according to the context, hence they are much more flexible and robust than fixed order Markov chain models. The amnesic probabilistic automata [49] aka PST (probabilistic suffix tree or prediction suffix tree), text compression [3] methods including PPM (Predication by Partial Matching) [14] and PPM* [13] are all in the family of variable order Markov models. PST have been applied to gene/protein sequence clustering [66] and spam filtering [45]. Researchers have tried to do text classification by using PPM, PPM* and other text compression methods [5, 18, 43, 58, 64].

2.2 The Discriminative Approach

In contrast to generative learning methods, discriminative learning methods do not posit a generative model, but attempt to find the optimal classification function directly. Previous studies have consistently shown that discriminative learning methods, particularly SVM, can achieve better performance than generative learning methods in word-based text classification [17, 29, 67].

The discriminative approach to string-based text classification attempts to utilize the substrings of a document as its features. However, a document of length $|d|$ has $|d|(|d|+1)/2$ substrings in total. The number of distinctive substrings in a reasonable size corpus would be extremely large. Consequently it would be impractical to enumerate all substrings in a corpus explicitly. This problem prohibits the application of most discriminative learning methods to string-based text classification.

2.2.1 String Kernel

SVM can overcome the problem of high dimensionality via the so-called *kernel trick* "which permits the computation of dot products in high-dimensional feature spaces, using simple functions defined on pairs of input patterns." [53]. SVM only require to know the dot product between any pair of feature vectors which could be computed via kernel functions, thus remove the need to express high-dimensional feature vectors explicitly. The idea of *string kernel*[1] [55] is to define the kernel between a pair of documents to be the weighted sum of their common substrings (of any length) which is actually the dot product between their feature vectors, if every distinctive substring is considered as a feature. To our knowledge, SVM with string kernel is the only exist-

[1]There are two kinds of "string kernel" in the machine learning literature. One uses all the continuous substrings of documents as features (e.g., in [62]), while the other also uses non-continuous substrings, aka, sub-sequences (e.g., in [41]). To avoid confusion, we would like to call the latter sequence kernel. In this paper, In this paper we focus on string kernel (defined on continuous substrings) because they are computationally more efficient than sequence kernel, though we do compare with sequence kernel in our experiments.

ing technique that can perform string-based text classification in a discriminative way.

In spite of the success of SVM with words-based kernels, SVM with string kernel has not become popular for text classification tasks. On the one hand, the computational efficiency of string kernel is still not comparable to its words-based counterparts, though fast string kernel algorithms via dynamic programming [41], trie [55] or suffix tree [62] can accelerate the computation speed to some degree. On the other hand, string kernel has not shown text classification performance that is as good as those of words-based kernels [41]. The effectiveness of string kernel might be impaired by the following factors.

Firstly, it is known that string kernel leads to the *ridge problem* [24] when applied to natural language documents, i.e., kernel values between almost identical documents are much larger than those between different documents. A document of length $|d|$ has at least $|d|(|d|+1)/2$ matches of substrings with itself. However, even if two documents of length $|d|$ share all words (with average length $|w|$) but in different orders, we have approximately $|w|(|w|+1)/2 \times (|d|/|w|) = |d|(|w|+1)/2$ matches only, where $|w| \ll |d|$. Thus the Gram matrix has a dominant diagonal, which is generally considered problematic.

Secondly, string kernel utilizes all distinctive substrings in the corpus as features, therefore the *redundancy* of features is quite high, which is likely to hurt the performance of SVM in text classification [19].

Furthermore, string kernel weights different substring features only according to their lengths. Since the number of substring features used implicitly in string kernel is extremely large, it would be infeasible to apply effective feature selection techniques(using χ^2 or *information gain*, etc.) [68, 19], feature weighting techniques (such as TF×IDF [60, 2]), or advanced kernel functions [28, 35, 72].

3. KEY-SUBSTRING-GROUP FEATURES

A corpus $D = \{d_1, d_2, ..., d_m\}$ of size n (i.e., $\sum_{k=1}^{m} |d_k| = n$) contains about $n(n+1)/2$ substrings. Our most important insight comes from the fact that these substrings could be divided into statistical *equivalence groups*. All the substrings in such an equivalence group have exactly identical distribution in the corpus, so it is not necessary to distinguish among them for statistical machine learning. That is to say, a group of substrings could be taken in whole as a single feature. To further reduce the dimension of the feature space, we filter the substring-groups by some statistical criteria. The *suffix tree* [21] data structure, which is extensively used in string data indexing and querying, makes it possible to extract this sort of features very efficiently.

In order to explain this, we first review the suffix tree data structure (in section 3.1), then propose the idea of key-substring-group features (in section 3.2 and 3.3), finally present the feature extraction algorithm and its time complexity analysis (in section 3.4).

3.1 Data Structure

Definition 1. [21] Consider a string S of n characters: $S = c_1c_2...c_n$. The *suffix tree* T for S is a compacted trie [34] that stores all suffixes of S. Specifically, T is a rooted directed tree with exactly n leaves numbered 1 to n. Let r denote the root of T. Each internal node other than r has at least two children and each edge is labeled with a nonempty substring of S. No two edges out of a node can have edge-labels beginning with the same character. For any leaf i, the concatenation of the edge-labels on the path from r to leaf i exactly spells out the suffix of S that starts at position i, i.e., $S[i..n] = c_ic_{i+1}...c_n$. To guarantee that a suffix tree for any string S actually exists, every string is assumed to end with a special termination character \$.

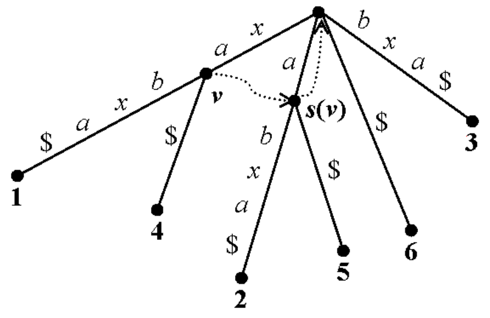

Figure 2: The suffix tree for string $xabxa\$$.

THEOREM 1. *[21] A suffix tree T has at most n internal nodes.*

Definition 2. [21] The *label of a path* in T from the root r that ends at a node is the concatenation, in order, of the substrings labeling the edges of that path. The path-label of a node v in T is the label of the path from r to v. A path from r that ends in the middle of an edge (u, v) splits the label on (u, v) at a designated point. Define the label of such a path as the path-label of u concatenated with the characters on edge (u, v) down to the designated point.

THEOREM 2. *[21] Given an arbitrary string P, we can find all occurrences of P in S in $O(|P|)$ time taking advantage of the suffix tree T for S [21]. This is done by matching the characters of P along the unique path in T until no more matches are possible or P is exhausted. In the former case, P does not appear anywhere in S. In the latter case, every leaf in the subtree below the point of the last match is numbered with a starting location of P in S, and every starting location of P in S numbers such a leaf. If the last match point is in the middle or at the lower end of an edge (u, v) (where u is the parent of v), we define $\overline{P} = u$ and $\underline{P} = v$.*

THEOREM 3. *[21, 62] If P is a substring of S, the occurrence frequency of P in S would be equal to the number of leaves in the subtree of T rooted at $\underline{P}$.*

Definition 3. [21] For an internal node v (other than r) in T with path-label $x\alpha$, where x denotes a single character and α denotes a (possibly empty) substring, if there is another node $s(v)$ with path-label α in T, then a pointer from v to $s(v)$ is called a *suffix link.*

For example, the suffix tree for string $xabxa\$$ is shown in Figure 2. The substring x must occur twice because the subtree under $\underline{x}$ (the node at the end of path xa) has two leaves. There are two suffix links: one from the node with path-label xa to the node with path-label a, and the other from the node with path-label a to the root node r.

THEOREM 4. *[21] For any node v (other than r) in T with path-label $x\alpha$ where x denotes a single character and α denotes a (possibly empty) substring, there must exist a unique node $s(v)$ with path-label α in T. If v is an internal node, we can jump from v to $s(v)$ in constant time via the suffix link between them. If v is a leaf, we can jump from v to $s(v)$ in constant time through the so-called "skip/count" trick.*

THEOREM 5. *[21] Ukkonen's algorithm [59] can construct the suffix tree T for a string S of length n, along with all its suffix links, in $O(n)$ time.*

Definition 4. [21] A *generalized* suffix tree can be constructed for a set of strings, i.e., a corpus, where each suffix of each string corresponds to a leaf.

3.2 Substring-Groups

The huge number of substrings in the training corpus D is an obstacle to making use of discriminative learning methods for text classification. However, by examining the suffix tree T for D, we see that all substrings of D could be clustered into a relatively small number of *equivalence groups*, according to their match points in T.

Definition 5. Each substring P in D must have a match point in T, hence correspond to a node $\underline{P}$. We define the *substring-group* SG_v corresponding to a node v in T to be the set of substrings $\{P_j | \underline{P_j} = v\}$.

Our idea is to take the substring-groups rather than individual substrings as features which would have been computationally expensive. The reduction of the computation over substrings to a computation over groups is made possible by the following theorems.

THEOREM 6. *The substrings in a group SG_v have exactly identical distribution over D. That is to say, if a substring $P \in SG_v$ occurs in a document $d \in D$ for f times, any other substring $P' \in SG_v$ must also occur in that document d for f times.*

PROOF. This could be proved straightforwardly using Definition 1 and Theorem 3. □

Almost all machine learning [44] methods for text classification only require the statistics of features, such as term frequency (TF) and document frequency (DF), to train classifiers. According to the above Theorem, it is not necessary to distinguish among substrings in one group for (statistical) machine learning. That is to say, such a substring-group could be taken in whole as a single feature. Note that this strategy would also help to solve the ridge problem and the high-redundancy problem of string kernel (Section 2.2.1).

Since all substrings in a group SG_v have the same occurrence frequency in D, we also call that frequency the occurrence frequency of SG_v, denoted by $freq(SG_v)$. From Theorem 3, we know that $freq(SG_v)$ equals to number of leaves in the subtree of T rooted at v.

THEOREM 7. *The substring-groups corresponding to the nodes in T (other than r) partition the set of all substrings in D.*

PROOF. There are two parts of this argument: every substring belongs to at most one group; and every substring belongs to at least one group. The former could be proved by contradiction using Definition 1, and the latter is implied by Theorem 2. □

THEOREM 8. *Suppose the corpus $D = \{d_1, d_2, ..., d_m\}$ is of size n (i.e., $\sum_{k=1}^{m} |d_k| = n$). Then there are n trivial groups whose substrings occur only once in D, and at most $n - 1$ non-trivial groups.*

PROOF. It can be easily shown that the trivial groups correspond to leaves in T, and the non-trivial groups correspond to internal nodes (other than r) in T. The suffix tree T for D has n leaves according to Definition 1, and at most $n - 1$ internal nodes (other than r) according to Theorem 1. □

Although there are about $n(n+1)/2$ substrings in a given corpus D of size n, there are only n trivial substring-groups and $n - 1$ non-trivial substring-groups. Moreover, it is actually not necessary to include trivial substring-groups as features, because they occur only once in the corpus so they won't be useful in learning classifiers.

To sum up, we can significantly reduce the potential dimension of feature space by taking the substring-groups rather than individual substrings as features.

These distribution of substring-groups seem to meet the Zipf's Law [74, 60, 30] like words in natural language documents as shown in our experiments (see section 4), which implies that it is plausible to regard them as *pseudo-words*.

3.3 Key-Substring-Groups

To further reduce the dimension of the feature space, we filter the substring-groups by the following criteria.

`-l` : the minimum frequency. A substring group SG_v is not taken as a feature, if it occurs less than `l` times in the corpus.

`-h` : the maximum frequency. A substring group SG_v is not taken as a feature, if it occurs more than `h` times in the corpus.

`-b` : the minimum number of branches (children). A substring group SG_v is not taken as a feature, if its corresponding node v has less than `b` branches (children).

`-p` : the maximum parent-child conditional probability. A substring group SG_u is not taken as a feature, if the probability $Pr[SG_v|SG_u] = freq(SG_v)/freq(SG_u) \geq$ `p`, where u is the parent node of v.

`-q` : the maximum suffix-link conditional probability. A substring group $SG_{s(v)}$ is not taken as a feature, if the probability $Pr[SG_v|SG_{s(v)}] = freq(SG_v)/freq(SG_{s(v)})$ $\geq$ `q`, where the suffix link of v points to $s(v)$.

The first and second criteria (`-l` and `-h`) are due to the fact that in natural language documents the words with very low or very high frequencies usually have little discrimination power [60].

The third criterion `-b`, which is actually equivalent to the number of possible unique characters following SG_v, partially reflects the *contextual dependency* of the substrings in that group. The more unique characters following a substring, the more contextual independent it is, hence the more suitable to be taken as a feature.

The last two criteria (-p and -q) aim at removing highly redundant features. In fact, the probability $Pr[SG_v|SG_u]$ is proportional to the *mutual information* [12, 46] between SG_v and SG_u, while the probability $Pr[SG_v|SG_{s(v)}]$ is proportional to the *mutual information* [12, 46] between SG_v and $SG_{s(v)}$. If two substring groups have a high mutual information, it should be sufficient to include just one of them as a feature.

The frequency information required by the above criteria can be computed efficiently using T, (Theorem 3).

Filtering substring groups by these five criteria not only reduces the dimension of feature space so as to improve efficiency, but also helps to solve the ridge problem and the high-redundancy problem of string kernel (Section 2.2.1). Moreover, since the number of key-substring-group features is relatively small, some effective feature weighting techniques (such as TF×IDF [60, 2]) and advanced kernel functions [28, 35, 72] can be applied easily, as in words-based text classification.

We call the selected substring-groups *key-substring-groups* because they are statistically critical. We propose to use these *key-substring-groups* as features to facilitate discriminative learning for string-based text classification.

The above feature selection criteria are unsupervised: they do not exploit the class labels of documents. It is possible to use more effective supervised feature selection criteria like χ^2 or *information gain* [68, 19] afterward.

Another good unsupervised feature selection criterion is the document frequency (DF) [68]. We are able to count the DF for every substring in linear time taking advantage of constant-time Least Common Ancestor (LCA) algorithm [4, 21], as in [65], though we omit the details here due to the space limit.

Our idea about key-substring-groups has been partially inspired by the works on suffix tree (or suffix array or PAT tree) based key-phrase extraction [8, 11, 37] and document clustering [69, 73]. The essential difference is that we do not care about whether the extracted key-substring-groups are semantically meaningful or not. In contrast, our concern is the utility of key-substring-groups as features for text classification.

3.4 Algorithm

We propose a feature extraction algorithm for key-substring-group features based on suffix tree [21], as shown in Figure 3. It converts each document in the training[2] corpus to a *bag of key-substring-groups*. The source code will be made open[3].

Theorem 9. *The key-substring-group extraction algorithm has linear time complexity $O(n)$, where n is the number of characters in the corpus.*

Proof. The construction of the suffix tree T using Ukkonen's algorithm takes $O(n)$ time, according to Theorem 5. Then, each of the three preparation sub-routines perform a

[2]With some little trick as in the matching statistics algorithm [9, 62], the proposed feature extraction algorithm could also handle the unseen test documents, though we omit the details here due to the space limit.

[3]http://www.dcs.bbk.ac.uk/∼dell/

Input: a training corpus D; a set of parameters l, h, b, p and q.
Output: a bag of key-substring-groups for each document $d \in D$.

```
// using Ukkonen's algorithm
construct the suffix tree T for D;
// assuming the root node of T is r

void count_freq(tree T, node v)
{
  freq[v] = stree_get_num_leaves(T, v);
  for (each child c of node v) {
    count_freq(T, c);
    freq[v] += freq[c];
  }
}

count_freq(T, r);  // recursively

void select_feature(tree T, node v)
{
  selected[v] = 1;
  if (v is the root of T) {
    selected[i] = 0;
  }
  else {
    if ((freq[id] < l) || (freq[id] >= h))
      selected[v] = 0;
    if (stree_get_num_children(T, v) < b)
      selected[v] = 0;
      node p_v = stree_get_parent(T, v);
      if (freq[v]/freq[p_v] >= p)
        selected[p_v] = 0;
      node s_v = stree_get_suffix_link(T, v);
      if (freq[v]/freq[s_v] >= q)
        selected[s_v] = 0;
  }
  for (each child c of node v)
    select_feature(c);
}

select_feature(T, r);  // recursively

void accumulate_feature(tree T, node v)
{
  if (selected[v])
    add i to feature_set[v];
  for (each child c of node v) {
    if (selected[v])
      add i to feature_set[c];
    accumulate_feature(T, c);
  }
}

accumulate_feature(T, r);  // recursively

// d --> a bag of key-substring-groups
for (each d in D) {
  // match the whole string d to T to get to
  // the node where the first suffix ends
  stree_match(T, d, length(d), &v, &pos);
  output feature_set[v];
  for (int i=2; i<=length(d); i++) {
    next_v = stree_get_suffix_link(T, v);
    // next_v is the node where the i-th suffix ends
    v = next_v;
    output feature_set[v];
  }
}
```

Figure 3: The key-substring-group feature extraction algorithm.

recursive traverse (depth-first-search) of T. Since T has n leaves and at most $n-1$ internal nodes according to Definition 1 and Theorem 1, a traverse takes $O(n)$ time. The conversion from a document d to a *bag of key-substring-groups* takes $O(|d|)$ time, because it processes $|d|$ suffixes each in $O(1)$ time according to Theorem 4. The trick here is to take full advantage of the suffix links to move from one suffix to the next suffix. Therefore the total time for conversion is $O(\sum_{k=1}^{m} |d_k|) = O(n)$. In summary, the time complexity of the proposed key-substring-group extraction algorithm is $O(n)$. $\square$

The above theorem indicates that the feature extraction algorithm is efficient and scalable, which has been confirmed by our experiments.

4. EXPERIMENTS

To evaluate our proposed approach, we conducted four experiments using SVM with key-substring-group features: (1) English text topic classification; (2) Chinese text topic classification; (3) Greek text authorship classification; and (4) Greek text genre classification. In each experiment, we first extracted the key-substring-group features by the proposed linear-time algorithm[4]. The algorithm's parameters `l`, `h`, `b`, `p` and `q` were tuned in the following way. Initially we took the default values for those parameters which would not filter any nontrivial substring-group out. Then we reduced the number of features gradually by adjusting the parameters, while keeping an eye on the cross-validation performance on the training data. Finally we selected the parameter configuration that could achieve a good trade-off between the number of features and the cross-validation performance. After the key-substring-group features were extracted, we weighted every feature value using the classic TF×IDF [60, 2] scheme, and normalized all feature vectors to unit length. LibSVM[5] [7] was employed as the implementation of SVM[6]. We chose the *linear kernel* and set all its other parameters to their default values. In all these experiments, our proposed approach worked very well.

4.1 English Text Topic Classification

We did the English text topic classification experiment on the ten largest categories of the Reuters-21578 dataset[7] (with the "ApteMod" training/testing split). Note that this dataset is widely regarded as the home-ground for words-based text classification but not string-based text classification.

All documents were pre-processed simply by lowercasing every alphabetical characters (letters) and converting every block of consecutive non-alphabetical characters to one "space" character ␣. Such pre-procession could be done easily *on-the-fly*.

[4]In English or Greek text, words are separated by whitespace characters or punctuations. We unconditionally discarded every key-substring-group that does not start from the beginning of a word, in experiments (1), (3) and (4). This trick slightly increased the program speed without affecting the performance.

[5]http://www.csie.ntu.edu.tw/∼cjlin/libsvm/

[6]LibSVM uses the *one-vs-one* ensemble method [26] for mutual-exclusive multi-class classification tasks, i.e., in experiments (2), (3) and (4).

[7]http://www.daviddlewis.com/resources/testcollections/reuters21578/

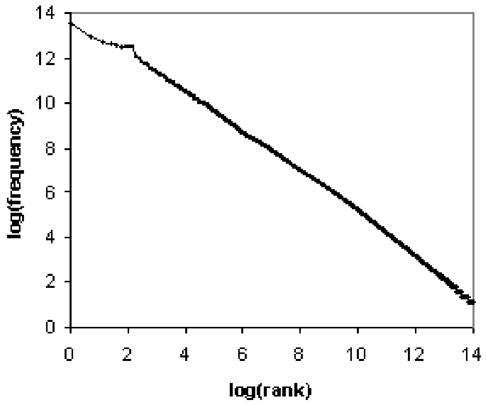

Figure 4: The log(rank) $\sim$ log(frequency) plot of the substring-groups in the Reuters-21578 top10 dataset.

	F_1	BEP
earn	98.3%	97.9%
acq	96.8%	97.4%
money-fx	84.0%	84.4%
grain	92.5%	94.6%
crude	89.4%	89.4%
trade	90.2%	91.5%
interest	81.5%	87.0%
ship	81.9%	85.9%
wheat	81.8%	86.5%
corn	87.1%	83.9%
macro-avg.	88.3%	89.8%
micro-avg.	93.9%	94.4%

Table 1: The performance of linear kernel SVM with key-substring-group features on the Reuters-21578 top10 dataset.

The log(rank) $\sim$ log(frequency) plot of substring-groups, i.e., suffix tree nodes, is approximately a straight line, as shown in Figure 4. It indicates that the distribution of substring-groups roughly follows Zipf's law [74], like words in natural language documents do [60, 30].

We set the feature extraction parameters as `-l 80 -h 8000 -b 8 -p 0.8 -q 0.8`. The dataset contains $4,460,825$ characters, therefore the total number of substrings would be more than 9×10^{12}. However, with the above setting of parameters, only 6,055 key-substring-group features were extracted and then utilized for text classification.

The text classification performance of our proposed approach (linear kernel SVM with key-substring-group features), measured by macro-averaged and micro-averaged [67] F_1 measure and precision/recall Break Even Points (BEP) [2, 31], is shown in Table 1, and compared with other state-of-art approaches in Table 2. Our approach achieved the best performance on this dataset.

To examine the robustness of our approach, we vary the chosen parameter configuration by changing one parameter value at a time and show its influence to the number of features and the text classification performance in Table 3. It seems that a decent performance could be achieved across a wide range of parameter settings.

The linear time key-substring-group feature extraction al-

		ma-F_1	mi-F_1	ma-BEP	mi-BEP
words-based	Naïve Bayes	——	——	69.8% [29]	72.0% [29]
	Rocchio	——	——	79.1% [29]	79.9% [29]
	Decision Tree (C4.5)	——	——	77.8% [29]	79.4% [29]
	k-NN	——	——	82.1% [29]	82.3% [29]
	Language Modeling (word n-gram)	——	81.5% [47]	——	——
	SVM (polynomial kernel)	——	——	——	86.0% [29]
	SVM (rbf kernel)	——	——	——	86.4% [29]
	SVM (linear kernel)	85.3% [41]	——	87.1% [17]	92.0% [17]
	SVM (word sequence kernel)	80.6% [6]	90.5% [6]	83.1% [6]	91.5% [6]
string-based	Language Modeling (character n-gram)	——	80.3% [47]	——	——
	Data Compression (PPMC Order 2)	——	——	74.3% [18]	——
	DVMM (VMM with discriminative feature selection)	——	——	——	87.0% [56]
	SVM (character sequence kernel)	77.3% [41]	——	——	——
	SVM (character n-gram, i.e., fixed-length string kernel)	80.6% [41]	——	——	——
	SVM (linear kernel, key-substring-group features)	**88.3%**	**93.9%**	**89.8%**	**94.4%**

Table 2: Comparing the experimental results of our proposed approach — linear kernel SVM with key-substring-group features — and some representative existing approaches.

gorithm did show very high efficiency in practice. The program implemented in C++ took less than 30 seconds on an ordinary Pentium-4 PC to process the Reuters-21578 top10 dataset. LibSVM also ran very fast, because the number of features was controlled to be relatively small.

4.2 Chinese Text Topic Classification

We did the Chinese text topic classification experiment on the TREC-5 People's Daily News dataset, as in [22, 23]. This dataset is a subset of the Mandarin News Corpus announced by the Linguistic Data Consortium (LDC) in 1995. There are six topic categories: (1) Politics, Law and Society; (2) Literature and Arts; (3) Education, Science and Culture; (4) Sports; (5) Theory and Academy; (6) Economics. Each category contains 500 training documents and 100 test documents. All documents were cleaned by removing SGML tags and merging continuous whitespace characters.

Setting the feature extraction parameters as `-l 20 -h 8000 -b 8 -p 0.8 -q 0.8`, we got a classification accuracy of **87.3%** (524/600) and a micro-averaged [67] F_1 measure [2] of **87.3%**. For comparison, He et al. reported a micro-averaged F_1 measure of about 82% also using SVM but after word segmentation [22, 23]; Peng et al. reported a micro-averaged F_1 measure of 86.7% using character-level n-gram language model [47].

4.3 Greek Text Authorship Classification

We did the Greek text authorship classification experiment on a dataset used by Stamatatos et al. [57]. This dataset consists of 200 Greek articles written by 10 different modern Greek authors: (1) S. Alaxiotis; (2) G. Babiniotis; (3) G. Dertilis; (4) C. Kiosse; (5) A. Liakos; (6) D. Maronitis; (7) M. Ploritis; (8) T. Tasios; (9) K. Tsoukalas; and (10) G. Vokos. There are 20 articles from each author — 10 articles for training and another 10 articles for testing (as in [57]).

Since this dataset is not large, we just used the default feature extraction parameter configuration which would not filter any nontrivial substring-group out. The classification accuracy we obtained is **92%** (92/100). For comparison, Stamatatos et al. reported a classification accuracy of 72% using deep natural language processing [57]; Peng et al. re-

`-l`	#f	ma-F_1	mi-F_1	ma-BEP	mi-BEP
3	18386	87.7%	93.8%	89.9%	94.5%
50	8294	87.8%	93.8%	90.1%	94.4%
80	6055	88.3%	93.9%	89.8%	94.4%
100	5255	88.4%	93.9%	89.9%	94.4%
500	1722	85.0%	92.4%	86.8%	92.8%
`-h`	#f	ma-F_1	mi-F_1	ma-BEP	mi-BEP
500	4333	84.0%	92.0%	86.1%	92.7%
1000	5069	87.6%	93.2%	88.9%	93.9%
5000	5945	88.4%	94.0%	90.1%	94.6%
8000	6055	88.3%	93.9%	89.8%	94.4%
20000	6173	88.1%	93.7%	89.3%	94.2%
`-b`	#f	ma-F_1	mi-F_1	ma-BEP	mi-BEP
2	9396	88.4%	94.1%	90.5%	94.8%
8	6055	88.3%	93.9%	89.8%	94.4%
16	2866	87.6%	93.8%	89.9%	94.5%
20	1391	87.4%	93.3%	88.8%	93.8%
`-p`	#f	ma-F_1	mi-F_1	ma-BEP	mi-BEP
0.2	999	85.5%	90.8%	86.9%	91.0%
0.4	3628	88.2%	94.1%	89.4%	94.5%
0.8	6055	88.3%	93.9%	89.8%	94.4%
1.0	6397	88.0%	93.8%	90.0%	94.5%
`-q`	#f	ma-F_1	mi-F_1	ma-BEP	mi-BEP
0.2	4609	88.6%	93.9%	89.8%	94.4%
0.4	5349	88.4%	93.9%	89.7%	94.3%
0.8	6055	88.3%	93.9%	89.8%	94.4%
1.0	6406	88.4%	94.0%	89.9%	94.5%

Table 3: The influence of key-substring-group feature extraction parameters to the number of features (#f) and the text classification performance.

ported a classification accuracy of 90% using character-level n-gram language model [47].

4.4 Greek Text Genre Classification

We did the Greek text genre classification experiment on a dataset used by Stamatatos et al. [57]. This dataset consists of 200 Greek articles in 10 different genres: (1) press editorial; (2) press reportage; (3) academic prose; (4) official documents; (5) literature; (6) recipes; (7) curriculum vitae; (8) interviews; (9) planned speeches; (10) broadcast news. There are 20 articles in each genre — 10 articles for training and another 10 articles for testing (as in [57]).

Since this dataset is not large, we just used the default feature extraction parameter configuration which would not filter any nontrivial substring-group out. The classification accuracy we obtained is **94%** (94/100). For comparison, Stamatatos et al. reported a classification accuracy of 82% using deep natural language processing [57]; Peng et al. reported a classification accuracy of 86% using character-level n-gram language model [47].

5. CONCLUSIONS

The motivation of this work is to make discriminative learning methods work for string-based text classification. Our proposed key-substring-group feature extraction technique solves the effectiveness and efficiency problems of string kernel, and opens numerous promising directions. SVM with key-substring-group features has exhibited very promising text classification performance, independent of language and task. Notably it can outperform traditional words-based text classification methods even on the Reuters-21578 top10 dataset, which is widely regarded as the home-ground of the latter. Therefore it is reasonable to believe that key-substring-group features are able to excel in various non-traditional text classification tasks that are naturally suitable to string-based approaches, e.g., spam filtering [5, 45]. Obviously key-substring-group features could also be used for text clustering [16, 69]. It is possible to go even further to consider applications in other areas like gene/protein sequence classification/clustering [39, 66].

6. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments.

7. REFERENCES

[1] A. N. Aizawa. Linguistic techniques to improve the performance of automatic text categorization. In *Proceedings of the 6th Natural Language Processing Pacific Rim Symposim (NLPRS)*, pages 307–314, Tokyo, Japan, 2001.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[3] T. C. Bell, J. G. Cleary, and I. H. Witten. *Text Compression*. Prentice-Hall, 1990.

[4] M. A. Bender and M. Farach-Colton. The LCA problem revisited. In *Proceedings of the 4th Latin American Symposium on Theoretical Informatics (LATIN)*, pages 88–94, 2000.

[5] A. Bratko and B. Filipič. Spam filtering using compression models. Technical Report IJS-DP-9227, Department of Intelligent Systems, Jožef Stefan Institute, Ljubljana, Slovenia, 2005.

[6] N. Cancedda, E. Gaussier, C. Goutte, and J.-M. Renders. Word-sequence kernels. *Journal of Machine Learning Research*, 3(Feb):1059–1082, 2003.

[7] C.-C. Chang and C.-J. Lin. LIBSVM : a library for support vector machines, 2001.

[8] C.-H. Chang and S.-C. Lui. IEPAD: Information extraction based on pattern discovery. In *Proceedings of the 10th International Conference on World Wide Web (WWW)*, pages 681–688, Hong Kong, 2001.

[9] W. I. Chang and E. L. Lawler. Sublinear approximate string matching and biological applications. *Algorithmica*, 12(4/5):327–344, 1994.

[10] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics (ACL)*, pages 310–318, Morristown, NJ, USA, 1996. Association for Computational Linguistics.

[11] L.-F. Chien. PAT-tree-based keyword extraction for Chinese information retrieval. In *Proceedings of the 20th Annual ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 50–58, Philadelphia, PA, 1997.

[12] K. W. Church and P. Hanks. Word association norms, mutual information and lexicography. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 76–83, Vancouver, BC, Canada, 1989.

[13] J. G. Cleary and W. J. Teahan. Unbounded length contexts for PPM. *The Comput Journal*, 40(2-3):67–75, 1997.

[14] J. G. Cleary and I. H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communication*, 32(4):396–402, 1984.

[15] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.

[16] D. R. Cutting, J. O. Pedersen, D. R. Karger, and J. W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 318–329, Copenhagen, Denmark, 1992.

[17] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the 7th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 148–155, Bethesda, MD, 1998.

[18] E. Frank, C. Chui, and I. H. Witten. Text categorization using compression models. In *Proceedings of the Data Compression Conference (DCC)*, page 555, Snowbird, UT, 2000.

[19] E. Gabrilovich and S. Markovitch. Text categorization with many redundant features: Using aggressive feature selection to make SVMs competitive with C4.5. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, pages 321–328, Banff, Alberta, Canada, 2004.

[20] J. Goodman. A bit of progress in language modeling, extended version. Technical report, Microsoft Research, 2001.

[21] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

[22] J. He, A.-H. Tan, and C.-L. Tan. A comparative study on Chinese text categorization methods. In *PRICAI'2000 International Workshop on Text and Web Mining*, pages 24–35, Melbourne, Australia, 2000.

[23] J. He, A.-H. Tan, and C. L. Tan. On machine learning methods for Chinese document categorization. *Applied Intelligence*, 18(3):311–322, 2003.

[24] R. Herbrich. *Learning Kernel Classifiers: Theory and Algorithms*. MIT Press, Cambridge, MA, USA, 2001.

[25] D. Holmes and R. Forsyth. The federalist revisited: New directions in authorship attribution. *Literary and Linguistic Computing*, 10(2):111–127, 1995.

[26] C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[27] P. Jackson and I. Moulinier. *Natural Language Processing for Online Applications: Text Retrieval, Extraction, and Categorization*. John Benjamins Publishing Co, 2002.

[28] T. Jebara, R. Kondor, and A. Howard. Probability product kernels. *Journal of Machine Learning Research*, 5:819–844, 2004.

[29] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning (ECML)*, pages 137–142, Chemnitz, Germany, 1998.

[30] T. Joachims. A statistical learning model of text classification for support vector machines. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 128–136, New Orleans, LA, 2001.

[31] T. Joachims. *Learning to Classify Text using Support Vector Machines*. Kluwer, 2002.

[32] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, 2000.

[33] B. Kessler, G. Nunberg, and H. Schtze. Automatic detection of text genre. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL) and 8th Conference of the European Chapter of the Association for Computational Linguistics (ECACL)*, pages 32–38, Madrid, Spain, 1997.

[34] D. Knuth. *The Art of Computer Programming*. Addison-Wesley, 3rd edition, 1997.

[35] J. D. Lafferty and G. Lebanon. Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research*, 6(Jan):129–163, 2005.

[36] J. D. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 111–119, Orleans, LA, 2001.

[37] M.-J. Lee and L.-F. Chien. Automatic acquisition of phrasal knowledge for English-Chinese bilingual information retrieval. In *Proceedings of the 21st Annual ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 351–352, Melbourne, Australia, 1998.

[38] Y.-B. Lee and S. H. Myaeng. Text genre classification with genre-revealing and subject-revealing features. In *Proceedings of The 25th Annual International ACM SIGIR Conference on Research and Development in Information (SIGIR)*, pages 145–150, Tampere, Finland, 2002.

[39] C. S. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Proceedings of the 7th Pacific Symposium on Biocomputing (PSB)*, pages 566–575, Lihue, HI, 2002.

[40] D. D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the 10th European Conference on Machine Learning (ECML)*, pages 4–15, Chemnitz, Germany, 1998.

[41] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research (JMLR)*, 2(Feb):419–444, 2001.

[42] C. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999.

[43] Y. Marton, N. Wu, and L. Hellerstein. On compression-based text classification. In *Proceedings of the 27th European Conference on IR Research (ECIR)*, pages 300–314, Santiago de Compostela, Spain, 2005.

[44] T. Mitchell. *Machine Learning*. McGraw Hill, international edition, 1997.

[45] R. M. Pampapathi, B. Mirkin, and M. Levene. A suffix tree approach to email filtering, 2005.

[46] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw Hill, New York, 2nd edition, 1984.

[47] F. Peng, D. Schuurmans, and S. Wang. Augmenting Naive Bayes text classifier with statistical language models. *Information Retrieval*, 7(3-4):317–345, 2004.

[48] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 275–281, Melbourne, Australia, 1998.

[49] D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25(2-3):117–149, 1996.

[50] R. Rosenfeld. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278, 2000.

[51] R. E. Schapire. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, Berkeley, CA, 2002.

[52] R. E. Schapire and Y. Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[53] B. Scholkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[54] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[55] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[56] N. Slonim, G. Bejerano, S. Fine, and N. Tishby. Discriminative feature selection via multiclass variable memory Markov model. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, pages 578–585, Sydney, Australia, 2002.

[57] E. Stamatatos, N. Fakotakis, and G. Kokkinakis. Automatic text categorisation in terms of genre and author. *Computational Linguistics*, 26(4):471–495, 2000.

[58] W. J. Teahan and D. J. Harper. Using compression based language models for text categorization. In J. Callan, B. Croft, and J. Lafferty, editors, *Workshop on Language Modeling and Information Retrieval*, pages 83–88, Carnegie Mellon University, 2001.

[59] E. Ukkonen. On-line construction of suffix-trees. *Algorithmica*, 14:249–260, 1995.

[60] C. van Rijsbergen. *Information Retrieval*. Butterworths, London, UK, 2nd edition, 1979.

[61] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 2nd edition, 2000.

[62] S. Vishwanathan and A. Smola. Fast kernels for string and tree matching. In K. Tsuda, B. Scholkopf, and J. Vert, editors, *Kernels and Bioinformatics*. MIT Press, Cambridge, MA, 2004.

[63] I. H. Witten. Applications of lossless compression in adaptive text mining. In *Proceedings of the 34th Annual Conference on Information Sciences and Systems (CISS)*, Princeton University, New Jersey, 2000.

[64] I. H. Witten, Z. Bray, M. Mahoui, and W. J. Teahan. Text mining: A new frontier for lossless compression. In *Proceedings of the 1999 Data Compression Conference (DCC)*, pages 198–207, Snowbird, Utah, 1999.

[65] M. Yamamoto and K. W. Church. Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. *Computational Linguistics*, 27(1):1–30, 2001.

[66] J. Yang and W. Wang. CLUSEQ: Efficient and effective sequence clustering. In *Proceedings of the 19th International Conference on Data Engineering (ICDE)*, pages 101–112, Bangalore, India, 2003.

[67] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 42–49, Berkeley, CA, 1999.

[68] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning (ICML)*, pages 412–420, Nashville, TN, 1997.

[69] O. Zamir and O. Etzioni. Grouper: A dynamic clustering interface to web search results. *Computer Networks*, 31(11-16):1361–1374, 1999.

[70] C. Zhai. *Risk Minimization and Language Modeling in Information Retrieval*. PhD thesis, Carnegie Mellon University, 2002.

[71] C. Zhai and J. D. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214, 2004.

[72] D. Zhang, X. Chen, and W. S. Lee. Text classification with kernels on the multinomial manifold. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 266–273, Salvador, Brazil, 2005.

[73] D. Zhang and Y. Dong. Semantic, hierarchical, online clustering of web search results. In *Proceedings of the 6th Asia Pacific Web Conference (APWEB)*, Hangzhou, China, 2004.

[74] G. K. Zipf. *Human Behaviour and the Principle of Least Effort*. Addison-Wesley, Cambridge, MA, 1949.