Digital Innovations & Contemporary Research in SCIENCE, ENGINEERING & TECHNOLOGY
Vol. 6. No. 1, March, 2018

Article Citation Format
A.A. Ojugo & D. Allenotor (2018): Text Mining Identification and Detection Using the Exact String Matching Algorithm: A Comparative Analysis
Journal of Digital Innovations & Contemp Res. In Sc., Eng & Tech. Vol. 6, No. 1. Pp 169-180

Article Progress Time Stamps
Article Type: Research Article
Manuscript Received: 12th Dec, 2017
Review Type: Blind
Final Acceptance:: 29th March, 2018
DOI Prefix: 10.22624

Text Mining Identification and Detection Using the Exact String Matching Algorithm: A Comparative Analysis

[1]A.A. Ojugo and **[2]D. Allenotor**
[1,2]Department of Mathematics/Computer Science
Federal University of Petroleum Resources
Effurun, Delta State, Nigeria
ojugo.arnold@fupre.edu.ng, allenotor.david@fupre.edu.ng

ABSTRACT

Text mining is a new burgeoning field in computer science research that tries to solve the crisis of information overload by combining techniques from data mining, machine learning, natural language processing, information retrieval, and knowledge management. Increase in document collection, often litters the collection with high rate of change. These pose search, performance and optimization challenges for various components of text-mining system. Users must better leverage their burgeoning textual data resources via tools that rely on a building up networks of interconnected objects via various relationships in order to discover patterns and trends. The study adopts a string-based matching algorithm which aims to extract, discover, and link together sparse evidence from vast amounts of data sources, to represent and evaluate the significance of the related evidence, and to learn patterns to guide the extraction, discovery, and linkage of entities.

Keywords— Bag of words, string, pattern matching and text-mining,

1. INTRODUCTION

Scientific researchers often employ literatures in their quest for new knowledge frontiers, as a major source of information during the course of their work. These, help to provide them with new ideas as well as to supplement their empirical studies, field work and result finding. However, some feel that this can be taken further: that new information, or at least new hypotheses, can be derived directly from the literature by researchers who are expert in information-seeking but not necessarily in the subject matter itself. Subject-matter experts can only read a small part of what is published in their fields and are often unaware of developments in related fields. Information researchers, students and other users do often seek useful linkages between related literatures, which is previously unknown – especially, if there is little explicit cross-reference between literatures (Willet, 1988; Witten and Frank, 2000). It should be noted that automatic text mining can aspire various studies via the example below.

We hear words like:

Birds of the same feather flock together
Birds fly towards the South in search of water
Birds choreograph in their motion etc

Reynolds (1987) extracted such information from titles in his quest for how agents react in response to their environment. This led to his agent-based model for which we have flock of bird flying in tight formation, which will collectively form an image with a goal movement as a single organism. Yet the flock choreographing in such grace has no group-leader bird. Instead, each bird reacts to the movement of its immediate neighbors, to result in hypnotic patterned rhythm and highly-nonlinear. Modeling such elegance not governed by any system is tedious or difficult due to its dynamism, complexity and nonlinear nature. But, it can be modeled as an aggregation of local feats interactions via 3-simple rules: (a) separation – each bird does not get too close to another, (b) alignment – each bird matches its direction and speed to nearest bird, and (c) cohesion – each bird stores in memory the perceived center and its immediate neighbor (Reynolds, 1987). It models each bird as an agent with local feats interaction to yield a highly realistic flight formation via simple rules – to result in theory of Agent Based Modeling (Ojugo et al, 2014).

By analyzing chains of causal implication in literature related to agent, their social relations and agent interactions, new hypotheses for causes of agent-models were discovered – some of which have received supporting experimental and empirical evidence (Macy and Willer, 2002). Thus, a plausible and new hypothesis emerged on social graphs that help predict tie strengths and other feats via combination of text fragments and the information researcher's background. However, these must be investigated via non-textual means (Breslauer et al, 1992).

1.1 Definition

Text mining is a new, exciting area in computer science that tries to solve the crisis of information overload by combining techniques from data mining, machine learning, knowledge management, information retrieval and natural language processing. Similarly, link detection is a rapidly evolving approach that involves analysis of text that shares and builds on many of the key elements of text mining. These, provides new tools to users so that they can better leverage their quest and burgeoning textual data resources. Link detection relies on a process of building up networks of interconnected objects through various relationships in order to discover patterns and trends. Major tasks of link detection are to extract, discover, and link together sparse evidence from vast amounts of data sources, to represent and evaluate significance of the related evidence, and to learn patterns to guide extraction, discovery, and linkage of entities (Feldman and Sanger, 2007).

Text mining is broadly defined as a knowledge-intensive process in which a user interacts with a document collection over-time by using a suite of analysis tools. Analogous to data mining, text mining seeks to extract useful data from sources via identification and exploration of interesting patterns. Many of the source document collections show interesting patterns (not often found among formalized database record); But; are in the unstructured textual data in these documents collection. Thus, text mining derives much of its inspiration and direction from seminal research on data mining – making data cum text mining systems evince in many architectural similarities. For instance, both types of systems rely on preprocessing routines, pattern-discovery algorithms, and presentation-layer elements such as visualization tools to enhance the browsing of answer sets.

Further, text mining adopts many of the specific types of patterns in its core knowledge discovery operations that were first introduced in data mining research (Porter, 2002).

1.2 String Matching

The exact string matching algorithm simply aims to match a pattern to the text. The problem however, arises if the strings can be found repeatedly anywhere in the text (i.e. the number of strings to find) is potentially large. Also, in some cases and applications, finding similar string instead of exact strings, is be needed. This is equal to a string search using wildcards. For example, instead of searching for the string perl.exe, one can search for p*rl.exe to get results pearl.exe, perl.exe, parl.exe etc (Witten, Moffat and Bell, 1999). Basically, 2-mechanisms based on two different ideas on how to handle character substitution or insertions (Witten and Bainbridge, 2003):

a. Substitution Error allows the replacement of one or more characters in a key search string. So, instead of searching the whole key string, only some positions are considered. This method cannot find character insertions or deletions.
b. Resemblance – here, the similarities of the two strings are measured by dividing the number of characters they have in common by the length of the longer string. Formally, this equals the division of the intersection by the union. Resemblance thus, can handle character insertion and deletion (where it exists). It drawback is that the character positions are not taken into account. So the two strings perl.exe and lexe.rpe have resemblance one. Clearly, this is not what we want. To overcome this, the order or position numbers can be introduced.

2. LEVERAGING ON FORMALIZED TEXT-MINING OPERATIONS

Data mining assumes data from various sources have been stored in a structured format, so that much of its preprocessing focuses on two critical tasks: (a) data scrub and normalizing, and (b) creating extensive numbers of table joins. As opposed to this in text mining, its preprocessing operations center on the identification and extraction of representative feats for natural language documents. It thus, transforms unstructured data stored in the document collections, into a more explicitly structured intermediate format. Due to the centrality of natural language text to its goal(s), text mining draws on the advances that explores and exploits techniques from the areas of information retrieval, information extraction, and corpus-based computational linguistics (Feldman and Dagan, 1995). It transforms different elements contained in a natural language document from the usual irregular and implicitly unstructured representation – into an explicitly structured representation.

However, given the potentially large database and collection of words, phrases, sentences, typographical element errors, layout artifacts that even a short document may have – not to mention the potentially vast number of different senses that each of these elements may have in various contexts and combinations, an essential task of text-mining is its focus in the identification of a simplified subset of the documents features that can be used to represent a particular document. These have often been classified as the representational model and all efforts to develop an efficient representational model is daunting as each document usually contains large number of features. And the large number of features contained therein a document affects every aspect of the text mining system's approach, design, optimization and performance (Smith, 2002; Montes-y-Gomez et al, 2001b).

2.1 Structural Design of the Documents Feature

Text mining algorithm and system operate on feature-based representations of documents with two important goals. First, it seeks to achieve a correct calibration of the volume and the semantic level of features to portray meaning of a document accurately, which tends to incline text mining preprocessing operations toward selecting or extracting relatively more features to represent documents.

The second goal is to identify features in a way that is most computationally efficient and practical for pattern discovery that emphasizes streamlining of representative feature sets which is supported by validation, normalization, or cross-referencing features against controlled vocabularies or external knowledge sources like dictionaries, thesauri, ontologies or knowledge bases to assist in generating smaller representative sets of more semantically rich features (Blake and Pratt, 2001).

To achieve these, it uses four (4) commonly used features (Feldman and Sanger, 2007):

a. Characters – these are individual component-level letters, numerals, special character and spaces, contained therein to form the basic building block for higher semantics known as bag-of-characters such as **bigram** or **trigram**)
b. Words (are cluster of characters concatenated together to form words, selected from the native document. It often consists of a subset of representative features filtered for items like meaningless numeric, symbolic characters and stop words that forms bag-of-words etc.
c. Terms are words and multi-word phrases selected directly from the corpus of native document via term extraction methodologies. Term level feats can only be made up of specific words and expressions found within the native document for which they are generally representative of. E.g. "Buhari is the President of Nigeria" contains words like "Buhari,", "is", "the", "President", and "Nigeria" as well as multiword such as "President of Nigeria" etc.
d. Concepts are features generated for a document by means of manual, statistical, rule-based or hybrid categorization methodologies – that often uses complex preprocessing routines that identifies single words, multi-words, whole clauses, or even larger syntactical cum semantic units that are related to the specific phrase or concept identifier. Many of these concept categorization methodologies involves the use of cross-referencing against an external knowledge source (for statistical methods – the source may be annotated collection of training documents). Thus, concepts can include words and expressions not found in the native document.

2.2 Search for Patterns and Trends

Though, text mining preprocessing operations plays critical role in transforming unstructured content into a more tractable concept-level data representation, the core functionality of a text mining system resides in the analysis of co-occurrence patterns across documents in a collection. Thus, it relies on algorithmic and heuristic approaches to consider distributions, frequent sets, and various associations of concepts at an inter-document level in an effort to enable a user to discover the nature and relationships of concepts reflected in the collection as a whole.

Text mining methods are largely based on a large-scale brute-force search directed at large, high-dimensionality feature sets that yields very large numbers of patterns. This, results in an overabundance problem with respect to identified patterns that is usually more severe than that encountered in data mining applications aimed at structured data sources.

A main operational task for text mining systems is to enable a user to limit pattern overabundance by providing refinement capabilities that key on various specifiable measure or degree of "interestingness" for search results. This also prevents users from getting overwhelmed by too many uninteresting results. The problem of pattern overabundance exists in all knowledge discovery activities. It is simply heightened when interacting with large collections of text documents, and, therefore, text mining operations must necessarily be conceived to provide not only relevant but also manageable result sets to a user.

Text mining also builds on various data mining approaches first specified in Lent, Agrawal, and Srikant (1997) to identify trends in data. In text mining, trend analysis relies on date-and-time stamping of documents within a collection so that comparisons can be made between a subset of documents relating to one period and a subset of documents relating to another. Trend analysis across document subsets attempts to answer certain types of questions.

For instance, in relation to a collection of news stories (Montes-y-Gomez et al, 2001a) suggests that trend analysis concerns itself with the following:

a. What is the general trend of the news topics between two periods (represented by two different document subsets)?
b. Are the news topics nearly the same, or are they widely divergent across the two periods?
c. Can emerging and disappearing topics be identified?
d. Did any topics maintain the same level of occurrence during the two periods?

3. MATERIALS AND METHODOLOGY

Given a problem for which there exists the alphabet(s) $\sum$, a string of text T (search space – from where we search for the possible solution) and its length $|T| = n$, and the string that is searched known as the pattern P and its length $|P| = m$. The problem is to search for the given pattern P amongst the search space string text T – so that a solution can be found using shift methods. It is clear that the text T must have at least the same length as the pattern P as $n \geq m$; otherwise, the problem is intractable. Thus, the exact string matching problem consists of finding all valid shifts with which a given pattern P occurs in a given text T. A position $\mathbf{s} \in \{0,...,n - m\}$ in T is a valid shift if P occurs with shift **s** in T – expressed mathematically as (Guo, 2014, Klaib et al, 2007):

$$pattern[\mathbf{i}] = text[\mathbf{s}+\mathbf{i}],\ i \in \{0,..., m - 1\} \quad (1)$$

In all other cases, **s** is an invalid shift. If **s** is a valid shift, then we have that $P\{0...m - 1\} = T\{\mathbf{s}...\mathbf{s} + m - 1\}$ (2).

3.1 Naïve Algorithm

The naïve algorithm yields a solution that makes comparison character by character of the text T [s...s + m – 1] for all $s \in \{0,..., n - m +1\}$ and the pattern P{0...m – 1} such that it returns all the valid shifts found. The solution is quite effective in that given the pattern to be search for P from the text T, we need **m**-operations of comparison by shift. Thus, for all text – we need $(\mathbf{n} - \mathbf{m} + 1) * m$ operation(s). Generally, because **m** is very small compared to **n** – it yields a time complexity O(**nm**). Its pseudocode is given on the next page

Pseudocode

```
// Return an array which contains all valid shifts in text (str)
NaiveMethod(text, pattern)
{
   n = length(text);
   m = length(patter,);
   limit = n – m;
   j = 0, k = 0;
   arrayOfValidShift[];

   for(i = 0; i <= limit; i++)
   {
      j = 0;
      k = i;
      for (j = 0; j <= m AND str[k] == pat[j]; j++)
         k++;
      if (j >= m)
         Add i to arrayOfValidShift;
   }
   return arrayOfValidShift;
}
```

3.2 Knuth-Morris-Pratt (KMP) Algorithm

As in fig 1 below

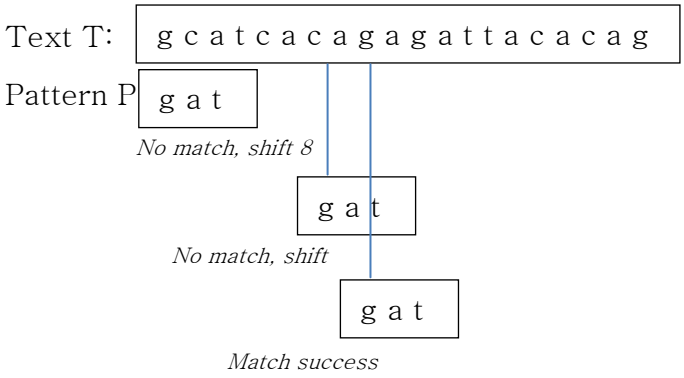

Fig. 1: Knuth–Morris–Pratt String

KMP is a linear time algorithm, more accurately O(**n**+**m**). Its main characteristic is that each time a match between pattern and a shift in text fails, the algorithm will use the data given by a specific table, obtained by a preprocessing of the pattern, to avoid re-examine the characters that have been previously checked, thus limiting the number of comparison required. So KMP consists of two parts, a **searching** part which consists to find the valid shifts in the text, where the time complexity is O(N), obtained by comparison of the pattern and the shifts of the text, and **preprocessing** part which consists to preprocesses the pattern.

The goal of the preprocessing of pattern consists to obtain a table that gives the next position in the pattern to be processed after a mismatch. For a pattern P[0...**m** – 1], the table of result of the preprocessing will give for each character j contained in the pattern a value which is defined as the substring that is in the same time the longest prefix of the pattern and the suffix of the substring of pattern p'[0...j]. The complexity of the preprocessing part is O(**m**), applying the same searching algorithm to the pattern itself.

3.3 Aho-Corasick

Zink (2014) A trie is a digital tree (also called a radix or prefix tree) with a graph-like structure such that in searching for a string, the structure can be searched by prefixes. Thus, it is an ordered tree data structure that is used to store a dynamic set or associative array where the keys are usually strings. However, the Aho-Corasick algorithm builds a trie of all the characters in the key strings that should be found – such that to find all key strings in one pass over the payload, it pre-computes failure pointers that points from already recognized characters to all possible suffixes. So if a failure occurs – the trie follows the failure pointers to the next character, if it is recognized. Fig 2 shows a 2-phase building of Aho-Corasick algorithm trie using the key strings {track, crack, race, trace}. As is seen, all key strings can be found with one-pass over the search string (the packet payload). On a hit, read string is recognized.

However, on a miss or failure (where the read string is not recognized) – traversing the trie starts anew so that the next character is not part of a key string already read and the trie follows the failure pointer to identify the new string. In the case where such occurs, with the key string not recognized, it implies that it is a genuine connection (Zink, 2009). Basically, the algorithm trie forms a state transition machine resembling the characters read. Thus, it is called Aho-Corasick automaton. It is efficient in space consumption – since the trie is already compressed and in complexity. It provides a linear search time. Its only drawback is the large number of failure pointers that can arise.

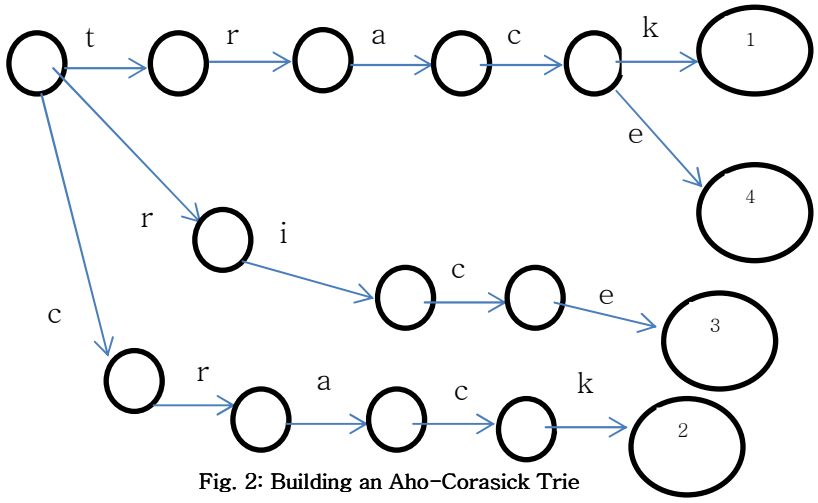

Fig. 2: Building an Aho-Corasick Trie

3.4 Boyer Moore Algorithm (BMA)

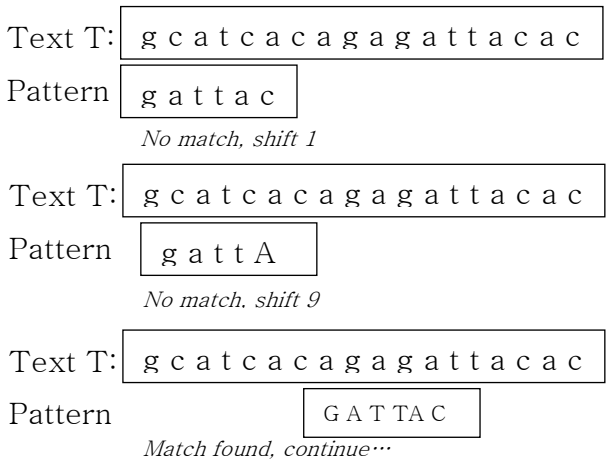

Fig. 3: Boyer Moore algorithm

Ojugo (2017) BMA is one of the most efficient exact string matching algorithms available as it actually does find matches in a sub-linear search time. It achieves this by simply scanning through the key string from left to right. On a miss, the key string is shifted a pre-computed number of characters to the right until a match of the current character occurs. Then, the next character not yet matched is considered. Since the length of the key string and the position of the current character is known, the number of characters on which the match of the key string can occur can be computed. Fig. 3 shows example with characters matched depicted using upper-cases. What is clear on inspecting the sample is that the algorithm can find exact string matches in sub-linear search time. Its demerit(s) includes that: (a) in the search for multiple key strings in a payload – the algorithm is quite inefficient, and (b) each key string has to be stored in its entirety.

3.5 Problem Statement

A major/key element in text mining is its focus on document collection (grouping of text-based documents) as either static (in which the documents remains unchanged) or dynamic (in which the document collection is characterized by its inclusion of new or updated documents over-time). But, challenges arise in malware detection (where codes to be detected are dynamic and always changing with insertion of whitespaces etc) as well as in the scenario below, we note that:

1. Large document collections and those with very high rate of document change often pose performance search and optimization challenges for the various components of a text-mining system. How do we best resolve such issues?
2. How can we best transform an irregular and implicitly unstructured representation into an explicitly structured representation?
3. Given a potentially massive collection of words, phrases, sentences, typographical errors, layout artifacts that any given document can feature as well as the potentially vast number of different senses that each of these may have in various contexts and combinations, how can we easily present it to users such that they can make meaning more easily from the document?

3.5 Study Sample

The goal here, is to use string matching algorithm to find a pattern P from text T using various algorithms as we measure various properties for the different string matching algorithm. For the study, we use the textbook "Introductory University Computer Science, edited by Allenotor, Ojugo and Oyemade", with text T = 7,320,103 and the pattern P = "program".

4. FINDINGS AND DISCUSSION

4.1 Findings

Using a simple Python implementation of various matching algorithms, we compared algorithms based these feats:

Classification Accuracy

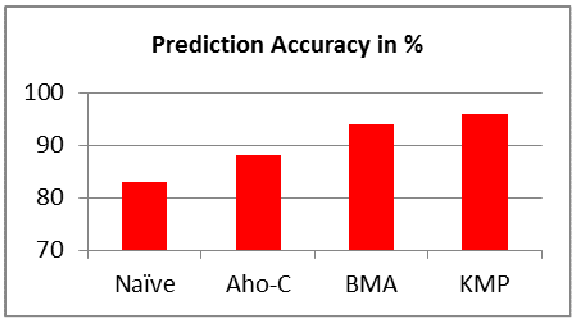

Fig. 4: Prediction Accuracy of Algorithms in percentage

Fig 4 shows prediction accuracy for algorithms explained with graph representation.

Processing Speed

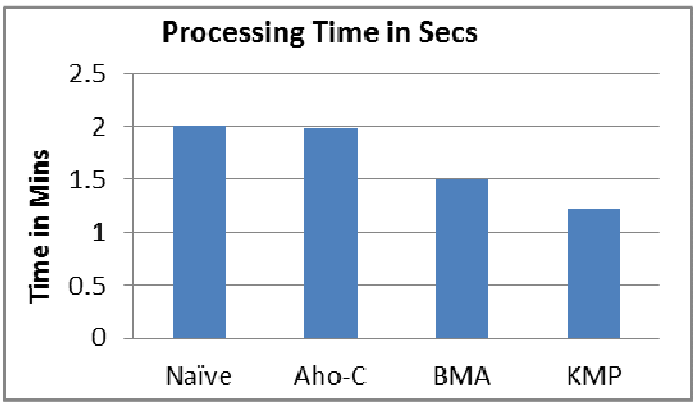

Fig. 5: Processing time in Seconds

Fig 5 shows processing time for the various string matching algorithms.

Convergence Time

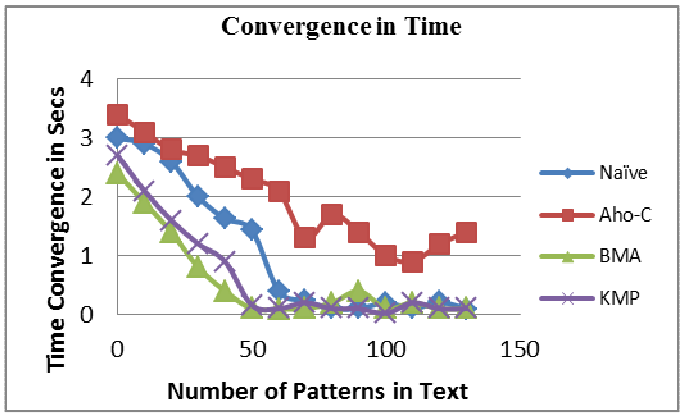

Fig. 6: Convergence time of matches

Fig 6 shows the convergence time of matches in which the various algorithms took in finding the patterns within the text.

4.2 Discussion

We provide a summary table 1 for tabular comparison:

Table 3: Comparative Result

Items	Naïve	Aho-Corasick	BMA	KMP
Text	7,320,103	7,320,103	7,320,103	7,320,103
Pattern	program	program	program	program
Convergence	3.11secs	2.96secs	1.39secs	1.19secs
Matches |T|	1902	1829	2003	1967

Looking at the results, we conclude that BMA yields the best algorithm result in majority of cases – because, the more long its pattern is, the more its advantage becomes significant. In tandem with Guo (2014), the reason being that it skips more characters during a shift so that its complexity is sub-linear yielding O(N=M). But, this advantage disappears if pattern is very small. KMP and Naïve algorithm obtain similar results (though, KMP is always a bit better). KMP is also a very good choice if the length of pattern is very short. We also notice that the results for shorter patterns show more similarities between the pattern and substrings for the given text even when there are no matches. These similarities require more comparisons and must be further processed to ascertain a true result.

5. RECOMMENDATIONS / CONCLUSION

In many applications today, finding the appropriate content in minimal time is critical and quite important. Thus, string algorithms play a vital role in this. Data mining and text mining techniques are today employed by different groups that are working both on software and hardware levels to make pattern searching faster. Researchers must continue in their implementation and quest to improve string algorithm usage in different applications. BMA and KMP are recommended in many cases with their reduced complexity and computation time.

They have been successfully used in many applications. They may not be the best optimal algorithm for a case; But, they have proven to be better than most algorithms. Most text and data mining applications today use BMA and/or KMP for their effective and efficient functionality – due to their lesser search and processing time complexity. Other algorithms depends upon the type of input and is efficient for certain or particular application.

REFERENCES

[1] Allenotor, D., Ojugo, A.A and Oyemade, D.A., (2016). Introductory University Computer Science, Published Lecture notes in Computer Science for University Students, ISBN: 978-978-41320-1-X, His Bride Publishers: Asaba.

[2] Blake, C., and Pratt,W. (2001). Better Rules, Fewer Features: A Semantic Approach to Selecting Features from Text. In Proceedings of the 2001 IEEE International Conference on Data Mining. San Jose, CA, IEEE Computer Society Press, New York: 59–66.

[3] Breslauer, D., Colussi, D and Toniolo, L., (1992). Tight Comparison Bounds for the String Prefix Matching Problem, Stiching Mathematisch Centrum, Amsterdam, 1-9, 1992.

[4] Chaudhary, R., Rasool, A and Khare, N., (2012). Variations of Boyer-Moore string matching algorithm: a comparative analysis, International Journal of Computer Science and Information Security, Vol. 10, No. 2, pp 95 – 101.

[5] Feldman, R., and Dagan, I. (1995). Knowledge Discovery in Textual Databases (KDT). In Proceedings of 1st International Conference on Knowledge Discovery and Data Mining. Montreal, Canada, AAAI Press, Menlo Park, CA: 112–117.

[6] Feldman, R and Sanger, J., (2007). The text mining handbook: advanced approaches in analyzing unstructured data, Cambridge University Press, ISBN-13: 978-0-521-83657-9

[7] Guo, M., (2014). Algorithms for string matching, [online]: www.google.com/ retrieved on December 21 2016.

[8] Klaib, A.F., Zainol, Z., Ahamed, N.H., Ahmad, R and Hussin, W., (2007). Application of Exact String Matching Algorithms towards SMILES Representation of Chemical Structure, World Academy of Science, Engineering and Technology, Vol. 34, pp 36 – 40.

[9] Lent, B., Agrawal, R and Srikant, R., (1997). Discovering trends in text databases. In Proceedings of the 3rd Annual Conference on Knowledge Discovery and Data Mining (KDD-97) D. Heckerman, H. Mannila, D. Pregibon, and R. Uthrysamy, eds. Newport Beach, CA, AAAI Press, Menlo Park, CA: 227–230.

[10] Macy M and Willer, J., (2002). From factor to actors: computational sociology and agent based model, Annual Review Sociology, Vol. 28, pp 143–166

[11] Montes-y-Gomez, M., Gelbukh, A., and Lopez-Lopez, A. (2001a). Discovering Association Rules in Semi-Structured Data Sets. In Proceedings of the Workshop on Knowledge Discovery from Distributed, Dynamic, Heterogeneous, Autonomous Data and Knowledge Source at 17th International Joint Conference on Artificial Intelligence (IJCAI'2001). Seattle, AAAI Press, Menlo Park, CA: 26–31.

[12] Montes-y-Gomez, M., Gelbukh, A., and Lopez-Lopez, A. (2001b). Mining the News: Trends, Associations and Deviations." Computacion y Sistemas, Vol. 5, No. 1, pp 14–25.

[13] Ojugo, A.A., (2017). Introductory text mining for graduate studies, Lecture notes in Computer Science for PG Students, www.fupre.edu.ng/mcs/cs-notes/data_text_mining/notes.html

[14] Ojugo, A.A., Yoro, R.E., Eboka, A., Yerokun, M., Anujeonye, C.N and Efozia, F.N., (2015). Predicting Behavioural Evolution on a Graph-Based Model, Advances in Networks, Vol. 3, No. 2, 2015, pp. 8-21, doi: 10.11648/j.net.20150302.11

[15] Porter, A. (2002). Text Mining. Technology Policy and Assessment Center, Georgia Institute of Technology.
[16] Reynolds C.W., (1987). Flocks, herds and schools: distributed behavioral model, Computer Graphics, 21, p25–34
[17] Sedgewick, R and Wayne, K., (2014). Algorithms – 4th Edition,
[18] Singla, N and Garg, D., (2012). String matching algorithm and their applicability in various applications, International Journal of Soft Computing and Engineering, Vol. 1, Issue 6, pp 218 – 222.
[19] Smith, D., (2002). Detecting and browsing events in unstructured Text, In Proceedings of the 25th Annual ACM SIGIR Conference. Tampere, Finland, ACM Press, New York: 73–80.
[20] Willett, P. (1988) "Recent trends in hierarchical document clustering: A critical review." Information Processing and Management, Vol. 24, No. 5, pp. 577–597.
[21] Witten, I.H., Moffat, A. and Bell, T.C. (1999) Managing gigabytes: compressing and indexing documents and images. Morgan Kaufmann, San Francisco, CA.
[22] Witten, I.H. and Frank, E. (2000) Data mining: Practical machine learning tools and techniques with Java implementations. Morgan Kaufmann, San Francisco, CA.
[23] Witten, I.H. and Bainbridge, D. (2003) How to build a digital library. Morgan Kaufmann, San Francisco, CA.
[24] Zink, T., (2009). Network security algorithms, Konstanzer Online Publikations-System, www.nbn-resolving.de/urn:nbn:de:bsz:352-175988, last retrieve December 10, 2016.