

Detection of Duplication in Documents and WebPages Based Documents Syntactical Structures through an Improved Longest Common Subsequence

Mohamed Elhadi
*Department of Computer Science,
Sultan Qaboos University, Oman*
elhadi@squ.edu.om

Amjad Al-Tobi
*Center of Educational Technology,
Sultan Qaboos University, Oman*
amjad@squ.edu.om

doi: 10.4156/ijipm.vol1.issue1.16

Abstract

This paper reports on experiments performed to investigate the use of a combined Part of Speech (POS) and an improved Longest Common Subsequence (LCS) in the analysis and calculation of similarity between texts. The text's syntactical structures were used as a representation for the documents. An improved LCS algorithm was applied to such a representation in order to compare and rank the documents according to the similarity of their representative strings. The approach was applied in the detection of duplicate documents within a corpus, and in the filtering of search engine results. Obtained results were encouraging.

Keywords: POS, Longest Common Subsequence, Duplication Filtering, Syntactical Structure

1. Introduction

Deciding whether two documents are similar can be a difficult task especially for a large collection of documents. Devising techniques to automate such a task can bring much needed help in this age of e-material and Internet. A number of researchers have discussed similarity [1] in contexts of copy detection, file management, information retrieval and other areas. Copy detection, in particular, is considered as one of the most important applications [1,2,3]. It is sufficient to think of copy detection as “the process of measuring and calculating the degree of similarity between two or more text objects, based on clear and specific discriminators that can be measured and quantified evidently” [3]. Among other areas, copy detection techniques are mainly used in filtering results returned by search engines [8], plagiarism and file and document redundancy management [9].

Existing detection techniques suffer drawbacks such as valuable information loss resulting from the producing of documents' representatives and the separation between semantic and syntactic features [22,23]. An alternative method is described in this paper with aim of producing a document representation as a string composed of a sequence of the Part-Of-Speech (POS) tags contained from the document's text. The produced string will be further analyzed using an improved Longest Common Subsequence (LCS) algorithm in order to calculate similarity measures for a set of documents.

Representing a text document as a string of syntactical units differs from traditional methods that are based on the actual characters or words of the text of the document. In this alternative method, text is represented using meaningful units that are well-defined and clearly-related. The produced string has the power not only to represent the documents' syntactical content but also to capture some of the writing style of the authors and relationships determined by the order and structural make-up of the units of text. This mapping of text to tags will result in huge savings in space by reducing the original texts. This in turn can translate to savings in time and improvements in performance. The proposed approach has an important advantage by its tolerance to minor changes in the make-up of the text. Changing of a letter(s) in a word or the total replacements of words (from the same syntactical category) will not affect how the tagging results. In addition, the approach has less information-loss due to its retention of punctuations and other text features.

2. Related Work

String analysis applications, such as computational biology, commonly use string representations and processing tasks [1,11,12,13] for comparison of strands of DNA and Proteins. They aim to compare unknown or partially known sequences to a collection of known and interpreted ones [12,13,18]. Use of POS tags to represent a text document allows for the use of similar techniques as those used in biology.

Similarity calculation methods fall into the semantic and syntactic types with syntactic approaches receiving much more attention especially fingerprinting (hashing) [15], ranking (or information retrieval) [11] and hybrid techniques [8,24,25].

Fingerprinting, widely used technique in identifying similar documents [15], is popular due to its short comparison time [24]. Systems that use fingerprint suffer from failure to provide information about where the overlap has occurred, susceptibility to false positives, and difficulty in detecting overlap between documents with big differences in their sizes [26].

Information Retrieval (IR), another interdisciplinary area that attracted attention [11,31], consists of a corpus, one or more indexes of its content, a query interface, a search system, and a results interface. IR systems' functionality can be divided into three categories [31]: (a) content analysis which is usually done automatically by identifying all stored items and every search request by assigning it to one or more content indicators, which are designed to reflect the information content, or the property set, which characterizes the given information item [32,33]; (b) Information structure which deals with mapping the relationship between documents to enhance efficiency and effectiveness of retrieval process; and (c) Evaluation which works on measuring the effectiveness of the retrieval process [31].

Cranfield evaluation methodology of recall and precision [24] are used as bases on relevance of retrieved information in response to the query.

Attempts have been made to merge some of the above mentioned techniques [8,24,25] resulting in many string alignment and manipulation methods been proposed and used in various domains. LCS algorithm is one of the earlier algorithms used to define similarity of sequences in biology and file management [35]. It solves the similarity problem using a brute-force approach, where all possible subsequences of one of the strings is first found, then each is tested to see if it is a subsequence of the other string [36,37]. Alternatively dynamic programming is used to reduce the time and space complexity of the problem using 2-dimensional array structure to store the size of the longest common subsequence [37].

The running time and space of the algorithm are both $O(mn)$. The space can be reduced to $O(n)$ if the trace back of longest subsequence is not needed.

POS tagging is the task of marking up the words in a text as corresponding to a particular part of speech category such as Noun, Verb, Adjective. The annotation process reflects both the word's definition and context [38,34]. Many tagging approaches have been investigated and implemented [29]. The most widely discussed in the literature generally fall into Rule-Based, Probability-Based (Bi- and tri-gram, Markov) and Memory-Based approaches [34,27,30].

Basic POS tags are verbs, nouns, pronouns, adjectives, adverbs, prepositions, conjunctions and interjections [28,46]. It is possible; however, to find systems that use tens and even hundreds of tags [17].

TreeTagger is a tool for annotating text with POS [29,16]. The tagger has a 55 tags-set and has achieved high accuracy of up to 96.36%. [20,21].

3. LCS-Refinements

Classical LCS is quite slow. To improve its speed two refinements termed Flip property and Slot splitting property respectively were considered. Brief descriptions of both improvements are described next [38].

3.1. Flip Property

Every row in the 2-dimensional array containing the LCS holds the value of the LCS that has been computed so far. If we assume that we have two sequences $S_1 = "slcdefmn"$ and $S_2 = "abcdef"$, then the

Detection of Duplication in Documents and WebPages Based Documents Syntactical Structures
through an Improved Longest Common Subsequence
Mohamed Elhadi, Amjad Al-Tobi

LCS between the two sequences is $S_3 = "cdef"$ which has a length of 4. Traditionally the LCS is computed by comparing every letter in S_1 to all letters in the S_2 (See Table 1).

Table 1. The sizes computed by normal LCS

		a	b	c	d	e	f
	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
s	1	0	0	0	0	0	0
l	2	0	0	0	0	0	0
c	3	0	0	0	1	1	1
d	4	0	0	0	1	2	2
e	5	0	0	0	1	2	3
f	6	0	0	0	1	2	3
m	7	0	0	0	1	2	3
n	8	0	0	0	1	2	3

For sequences S_1 and S_2 , we compare “s” of S_1 to all letters in S_2 which will result in no match. This leads to copying all values (zeros) of the previous row to the “s” row. At second iteration, the algorithm will do the same with “l”. At the third iteration, “c” will be compared to all letters of S_2 , yielding a match at position 3, where a FLIP from 0 to 1 takes place with no further flips even if there was another “c” in the sequence. Therefore, the largest value of this row will be 1. At the fourth iteration, letter “d” will be compared to S_2 resulting in another FLIP from 1 to 2 at position 4 giving us 2 as the largest value of this row. There will be another FLIP from 2 to 3 at the fifth iteration at position 5 followed by another FLIP from 3 to 4 at the 6th iteration at position 6. Since there are no more matches, the rest of iterations result in no further flips.

So a FLIP property is when there is a flip resulting in a higher value for LCS at i^{th} row than the value at previous row. It means that there would be no need to check the rest of this row, if we can memorize the new flip value and its position.

For example, when there was a flip at the third iteration –when “c” was compared to the S_2 - we could abort this iteration after memorizing the flip value and its position. Taking advantage of this Flip property, we developed an algorithm that would run at 50% of the original algorithm’s time $O(mn)$ at the best case. This is a valuable savings for reasonably large sequences [3].

Table 2 shows an example that traces the run of the new version of the LCS_V1 algorithm in comparing sequence “abcdef” to itself.

Table 2. The size table computed by LCS_V1

		a	b	c	d	e	F
	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
a	1	0	1				
b	2	0	1	2			
c	3	0	1	2	3		
d	4	0	1	2	3	4	
e	5	0	1	2	3	4	5
f	6	0	1	2	3	4	5

3.1. Slot-Splitting Property

The second refinement makes use of the dot plot technique [14] to compare two given sequences only at the regions that are marked to be similar. A closer look at LCS revealed that no row in the

result table can be computed unless the previous row has been already calculated. Moreover, each row can be seen as if it is made of a number of slots. Every table starts with an initial row only containing one slot that is made of zeros.

When there is no match, then the previous row will be copied. But when a match is encountered within a slot then this slot will be split further. Normally if there are many matches lying within one slot, only the first match will contribute in splitting that slot; whereas, the following matches will have no effect. At the typical implementation of the LCS algorithm, the length of the longest sequence will be the number of slots in the last row minus 1.

Here are the main steps in computing the size of the longest common sequence with an example:

- Store the position(s) of every character in the first sequence using a lookup table access.
- Look for every character in the second sequence in the lookup table. If it has an entry, then get the hit positions list.

If we assume the two sequences $S_1 = "slcdefmn"$ and $S_2 = "abcdef"$. The algorithm creates a lookup table for sequence S_1 . Then it scans sequence S_2 to check if it has an entry in the lookup table. If there is no entry for any character, as the case with "s" and "l", then the algorithm moves to the next character saving considerable time.

When the algorithm checks for "c" it finds that this letter has a match at position 3. Since the previous row is initialized to form one slot (position 0-∞), a break is inserted in the new row at the match position. The new row contains two slots, where first slot (position 0-2) and second slot (position 3-∞). This row, which is made of two slots, is the base row, which is used to compute the next row. When the algorithm processes the next character "d" it will result in a further split that produces a row with 3 slots (positions 0-2, 3-3 and 4-∞).

In the fifth iteration, where "e" is checked, results in a row with 4 slots (positions 0-2, 3-3, 4-4 and 5-∞). Checking letter "f" results in a row with 5 slots (positions 0-2, 3-3, 4-4, 5-5 and 6-∞). The last two iterations just ignore the last two characters – "m" and "n" - because there is no entry for them in the table.

This algorithm ignores computing unnecessary rows where there are no matches between the two strings. Also it takes advantage of the Flip properly refinement by stopping further computation as the score reaches a higher value than already computed.

Time complexity of finding match positions is $O(m+n)$. Experimental validations and analysis for these two refinements are presented in next section.

4. Approach and Experiments

Three sets of experiments were performed using the proposed ideas. Each set is described following an overview of the proposed methodology.

4.1. Methodology

The overall methodology as illustrated by Figure 1 is made of the following steps.

- Syntactical Processing Phase: A given text document will be processed to produce a representation for further processing.
- TreeTagger is used to tag the text to produce representation strings. Tags are mapped into single characters to save space and reduce comparison costs.
- String Optimization Phase: This phase is used to adjust the accuracy of the system by changing the size and type of the tags-set that is used to annotate a given text.
- Matching and Ranking Phase: Do further processing and analysis of the string using the refined and improved LCS algorithm.

Detection of Duplication in Documents and WebPages Based Documents Syntactical Structures through an Improved Longest Common Subsequence

Mohamed Elhadi, Amjad Al-Tobi

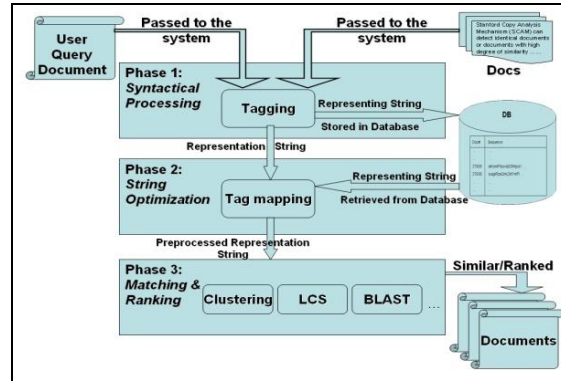


Figure 1. Proposed Approach

4.2. Experiments and Results

The experiments and their results are described after presentation of validation of the LCS algorithm's new refinements. Two areas were used in the experimentation, namely, duplicate detection using text corpus and web duplication detection.

4.2.1. LCS Refinements' Validation

In order to validate the LCS refinements a set of sequences with different sizes have been compared to each other in terms of run time. A total set of 68 sequences were selected with lengths ranging from 29 to 3551 characters [38].

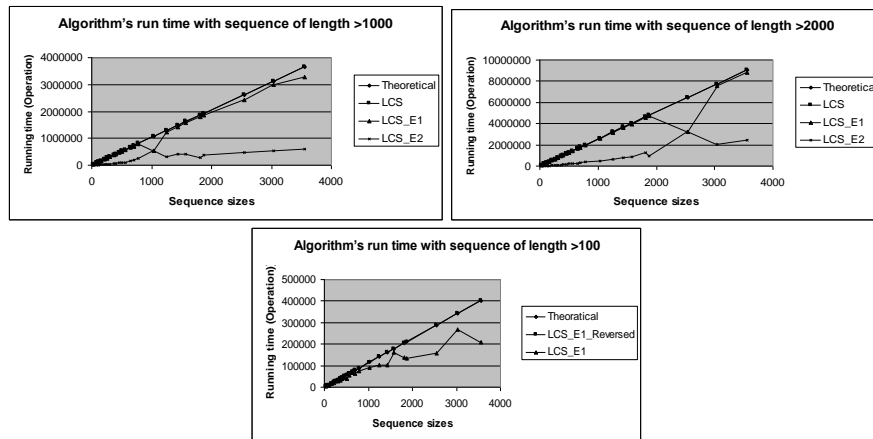


Figure 2. LCS run times using different lengths

Figure 2 shows the theoretical and actual (LCS) run times of the original algorithm and the actual run times of our first and second refinements. As figures show, the actual and theoretical run time of the original algorithm demonstrated similar behavior, whereas, the running time of the second refinement of the algorithm showed the best result. First refinement performance falls between the original and second refinement's run time. The run time of the first refinement reaches its best when it drops sharply recording about 50% of the theoretical and actual run time of the original algorithm. This happens when the sequence compares to itself.

4.2.2. Duplication Detection Experiments

The aim here is to investigate the effect of applying our method in detecting duplicate documents within large corpus. The intuition is that the existence of duplication should be captured and manifested as high similarity measures between relevant documents and vice versa.

A group of documents taken from Reuter's collection were selected and used in this experiment [7]. Reuter has defined a number of categories based on using region, industry and topic. Each document is categorized by Reuter as belonging to one or many regions, industries or/and topics.

Two sets of documents were created for this experiment. The first set contains 994 documents based on querying the Reuter's collection on topic category-wise. One topic class is queried at a time and the first 50 returned documents were selected with documents that are categorized to more than one topic selected. Therefore, this collection may contain duplicate documents.

The second set contains 935 documents and is based on querying the Reuter's collection but only documents that are categorized in the queried topic are chosen. The implication here is that, the single-topic set would contain no duplicates, because each selected document is categorized to one topic category only and each topic is queried just once.

Each of the above two sets was further divided into three equally sized groups for ease of analysis. The following procedure was used for this collection (for both sets) as Figure 3 demonstrates.

- Preprocessing phase: Convert document from XML to text.
- Syntactical Processing Phase: Tagging each document.
- String Optimization Phase: Map tags into single character.
- Matching and Ranking Phase: Compare each string to all others within the same tags-set. The results were divided into three different similarity score ranges: *Range 1*:0% to 30%; *Range 2*:30% to 60%; and *Range 3*:60% to 100%. These ranges were selected after manual checking of random samples in which it was found that, documents having similarity less than 30% tend to be not similar; whereas, those with similarity above 60% are mostly duplicates of each other. Documents that are ranging between 30% and 60% are mostly non-duplicate but with some suspicious ones.
- Compute the percentage of documents in each range for every group.

As shown in Table 3 and as expected, there were lots of duplicates in the multi-topic set. It was surprising, however, to see the amount of duplication in the single-topic set.

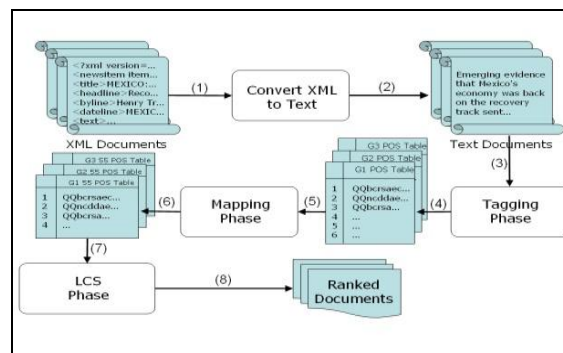


Figure 3. Phases of Experiment 2

It was thought that, there would be no duplicates in this set, because every document has got its own unique ID and it belongs to only one topic. Results have shown that duplications existed in this set which was also confirmed with manual verification suggesting that the proposed system can be effectively used to detect duplicate and near-duplicate documents.

Detection of Duplication in Documents and WebPages Based Documents Syntactical Structures
through an Improved Longest Common Subsequence
Mohamed Elhadi, Amjad Al-Tobi

Table 3. % of number of comparisons

Range	Type/Set	Set 1	Set 2	Set 3	Full
< 30%	RC-Set-Multi	99.62	99.34	99.39	99.45
	RC-Set-Single	99.95	99.94	99.89	99.92
30% -60%	RC-Set-Multi	0.12	0.25	0.30	0.22
	RC-Set-Single	0.03	0.04	0.10	0.06
> 60%	RC-Set-Multi	0.27	0.40	0.31	0.33
	RC-Set-Single	0.03	0.02	0.01	0.02

4.2.3. Web Collection Set Experiment

The aim here was the investigation of the utility of the proposed approach in filtering out duplicate WebPages returned as results by a search engine.

A set of documents, which were returned by AltaVista search engine as a result of submitting the phrase of “Perl Tutorial” was used in this experiments. It is worth noting that AltaVista may already use some duplication detection mechanism which means that our system may return limited results. As such any indication of duplications suggested by the system would be taken favorably.

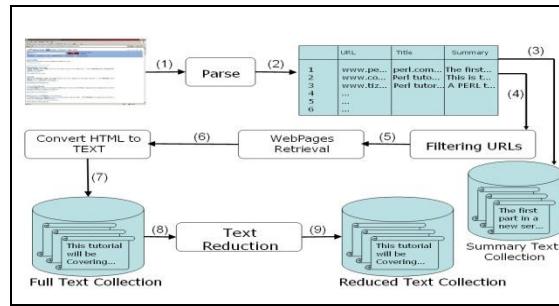


Figure 4. Pre-Processing Phase

Any non-text or broken links were removed first resulting in only a subset (of 433 documents) of the first 500 results to be selected and used in the experiment. Three main pieces of information were extracted as the search results pages were parsed; (1) title of the result page, (2) its URL and (3) the reported summary/description by the search engine. Each page’s URL was then downloaded and processed to extract the complete row text content. Finally, each text document will be reduced by selecting number of sentences (max of 10) that contain the query’s keyword(s). Figure 4 and Figure 5 illustrate the steps of this experiment:

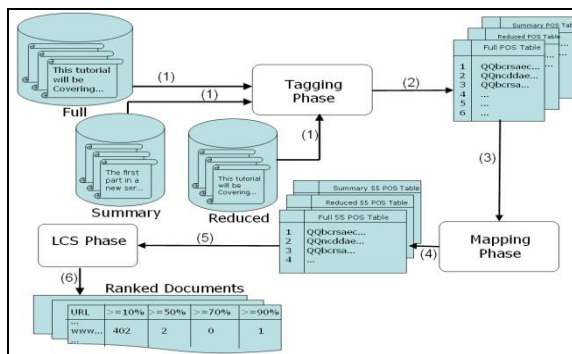


Figure 5. Phases 1-3 of Experiment 3

Although, using the summary has yielded a much faster execution time, it was not good to use the summary for measuring the similarity between the web documents.

This may be attributed to the ungrammatical pieces and parts of different sentence(s) produced. Moreover, our context reduced text has shown better results in performance as well as accuracy compared to the full text.

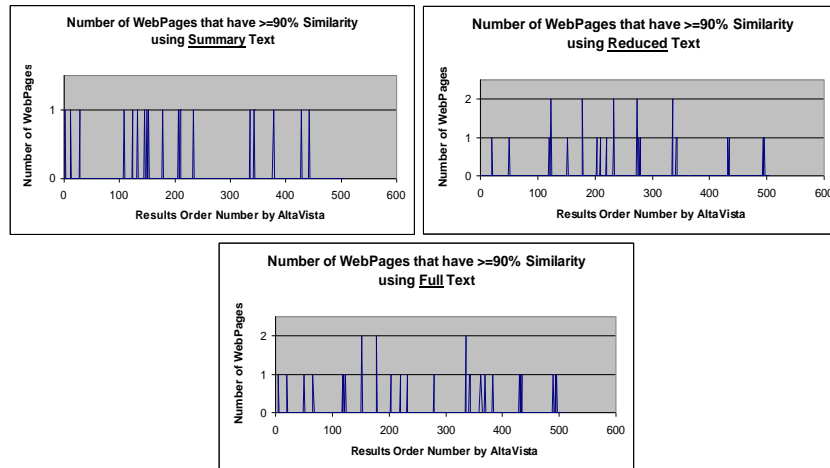


Figure 6. No. of duplicate with similarity $\geq 90\%$

The results indicate that most of the duplicate documents are ranked far from each other by AltaVista.com. They also show that some of these duplicates are highly ranked within the top 50 hits.

Figure 6 plots the amount of duplicate documents with similarity above 90% using different texts of Summary, Reduced and Full.

Reduced and Full sets were very similar to each other and with duplication whereas, summary results fail to represent real duplicates.

The above procedure was applied to (1) The returned webpage's summary provided by the search engine, (2) The full extracted text of the webpage and (3) The keyword context reduced text. From Figure 7 it can be seen that AltaVista has ranked some pages that had duplicates within the top 500 returned results. It is expected that the higher the duplication range considered, the fewer the number of duplicates. It is still, however, possible to detect duplicates in top 20 results even 90% is considered.

These are promising results in using text syntactical structure in combination with LCS to filter out duplication in the returned WebPages by a search engine.

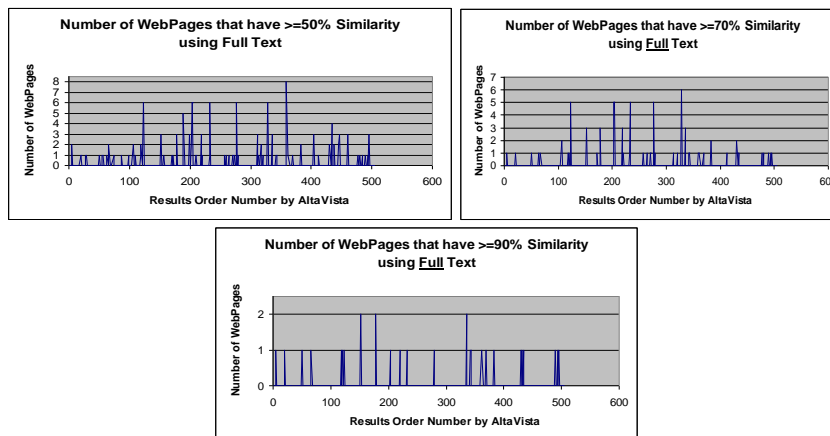


Figure 7. Duplicates using different reduced texts

5. Conclusion

A procedure that takes advantage of text syntactical content to represent documents' was developed. The documents representative strings generated using POS tagging were further

Detection of Duplication in Documents and WebPages Based Documents Syntactical Structures
through an Improved Longest Common Subsequence
Mohamed Elhadi, Amjad Al-Tobi

analyzed using an improved LCS, where a ranking of documents similarity was produced for the detection of duplication and the investigation of its utility.

Experiments were performed in order to validate our assumption and utility of the approach. One experiment used well known Reuters collection and the other used a set of documents that were collected from the web.

6. References

- [1] R. Steinberger, B. Pouliquen and J. Hagman, "Cross-lingual Document Similarity Calculation Using the Multilingual Thesaurus EUROVOC", in Proceedings of the 3rd Inter. Conf. of Computational Linguistics and Intelligent Text Processing, Mexico City, Mexico, pp. 415-424, 2002.
- [2] S. Brin, J. Davis and H. Garcia-Molina, "Copy Detection Mechanisms for Digital Documents", in the ACM SIGMOD International Conference on Management of Data, San Jose, California, USA, pp. 398-409, May 1995.
- [3] A. Al-Tobi, "Combined Syntactical Structures and Sequence Alignment Approach to Document Similarity Calculation for Copy Detection", M.Sc. thesis, Department of Computer Science, Collage of Science, Sultan Qaboos University, Muscat, Oman, 2008.
- [4] A. Zaslavsky, A. Bia, and K. Monostori, "Using copy-detection and text comparison algorithms for cross-referencing multiple editions of literary works", in Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries, Darmstadt, Germany, pp 103-114, September , 2001.
- [5] A. Si, H.V. Leong and R.W.H. Lau, "CHECK: A document plagiarism detection system", in Proceedings of ACM Symposium for Applied Computing, ACM, San Jose, California, USA, pp. 70-77, February 1997.
- [6] N. Shivakumar and H. Garcia-Molina, "SCAM: A Copy Detection Mechanism for Digital Documents", in Proceedings of 2nd International Conf. in Theory and Practice of Digital Libraries, Austin, Texas, USA, June, 1995.
- [7] REUTERS, Reuters Corpus (Volume 1: English Language, 1996-08-20 to 1997-08-19), Released date: Nov 2000.
- [8] Y. Liu and L. Liang, "A Dual-method Model for Copy Detection", in Proceedings of the IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent, Hong Kong Convention and Exhibition Centre, Hong Kong, pp. 634-637, December, 2006.
- [9] G. Forman, K. Eshghi and S. Chiochetti, "Finding Similar Files in Large Document Repositories", in Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, pp 394-400, August 2005.
- [10] M. Elhadi and A. Al-Tobi, "Use of Text Syntactical Structures in Detection of Document Duplicates", in Third IEEE International Conference on Digital Information Management, University of East London, London. UK, Nov2008.
- [11] A. Singhal, "Modern Information Retrieval: A Brief Overview", IEEE Data Engin. Bulletin, Vol. 24, No. 4, pp. 35-43, March 2001.
- [12] S.F. Altschul, W. Gish, W. Miller, E.W. Myers and D.J. Lipman, "Basic Local Alignment Search Tool", Journal of Molecular Biology, Vol. 215, No. 3, pp. 403-410, October 1990.
- [13] "Local Alignment: Smith-Waterman algorithm", class notes for CSE 591: Computational Molecular Biology, Depart. Of Computer Science & Engineering, Arizona State University, spring 2003.
- [14] A. M. Lesk, "Introduction to Bioinformatics", Second Edition, Oxford University Press Inc., New York, USA, 2005.
- [15] S. Schleimer, D. S. Wilkerson and A. Aiken, "Winnowing: Local Algorithms for Document Fingerprinting", in Proceedings of the ACM SIGMOD Inter. Conf. on Management of Data, San Diego, California, USA, pp. 76-85, June 2003.
- [16] Lexicon and Textcorpora Group, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart, Germany, "TreeTagger - a language independent part-of-speech tagger", 2003, <http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger>.

- [17] X.R. Hu and E. Atwell, "A survey of machine learning approaches to analysis of large corpora", in Proceedings of the Workshop on Shallow Processing of Large Corpora, Lancaster University, UK, pp. 45-52, March 28 - 31 2003.
- [18] M.S. Waterman, "General Methods of Sequence Comparison", Bulletin of Math. Biology, Vol. 46, No. 4, pp. 473-500, 1984.
- [19] ELC Courses, English Language Centre, University of Victoria, "Parts of Speech", 1997, <http://web2.uvcs.uvic.ca/elc/StudyZone/330/grammar/parts.htm>.
- [20] H. Schmid, "Probabilistic Part-of-Speech Tagging Using Decision Trees", in Proceedings of International Conference on New Methods in Language Processing, Manchester, UK, pp. 44-49, September, 1994.
- [21] H. Schmid, "First refinements in Part-of-Speech Tagging With an Application To German", in Proceedings of the 14th International Conference on Computational Linguistics, Kyoto, Japan, pp. 172-176, March 1995.
- [22] A.G. Maguitman, F. Menczer, H. Roinestad and A. Vespignani, "Algorithmic Detection of Semantic Similarity", in Proceedings of the 14th international conference on World Wide Web, Chiba, Japan, pp. 107-116, May 2005.
- [23] R. Mihalcea, C. Corley and C. Strapparava, "Corpus-based and Knowledge-based Measures of Text Semantic Similarity", in Proceedings of The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, Boston, Massachusetts, USA, July, 2006.
- [24] K. Monostori, R. Finkel, A. Zaslavsky, G. Hodasz and M. Pataki, "Comparison of Overlap Detection Techniques", International Conference on Computational Science, (Amsterdam, The Netherlands, pp 51-60, April, 2002.
- [25] N. Kang, A. Gelbukh and S. Han, "PPChecker: Plagiarism Pattern Checker in Document Copy Detection", in Proceedings of Text, Speech and Dialogue 9th International Conference, Brno, Czech Republic, pp 661-667, 2006.
- [26] N. Heintze, "Scalable Document Fingerprinting", In Proceedings of the Second USENIX Workshop on Electronic Commerce, Oakland, California, November, 1996.
- [27] J. Cussens, "Part-of-Speech Tagging Using Progol", in Proceedings of the 7th International Workshop (Inductive Logic Programming), (Prague, Czech Republic, pp. 93-108, September, 1997.
- [28] H. MacFadyen, University of Ottawa, "The Parts of Speech", 2007, <http://www.arts.uottawa.ca/writcent/hypergrammar/partsp.html>,
- [29] Wikipedia®, Wikimedia Foundation, Inc., "Viterbi algorithm", 8th Sept 2008, http://en.wikipedia.org/wiki/Viterbi_algorithm.
- [30] W. Daelemans, J. Zavrel, P. Berck and S. Gillis, "MBT: A Memory-Based Part of Speech Tagger Generator", in Proceedings of Fourth Workshop on Very Large Corpora, University of Copenhagen, Copenhagen, Denmark, pp. 14-27, August 1996.
- [31] C. J. van RIJSBERGEN, "INFORMATION RETRIEVAL", Department of Computing Science, University of Glasgow, London: Butterworths, 1979.
- [32] P. Ingwersen, Information Retrieval Interaction, Taylor Graham Publishing, London, United Kingdom, 1992.
- [33] G. Salton, "Automatic Content Analysis in Information Retrieval", Department of Computing Science, Cornell University, Ithaca, N.Y., 1968.
- [34] Yonghong Mao, "Natural Language Processing Module (Part of Speech Tagging and Sentence Parsing)", Cognitive Science in Context Laboratory, Cornell University, New York, U.S., 1997.
- [35] Wikipedia®, Wikimedia Foundation, Inc., "diff", 25th Sep 2007, <http://en.wikipedia.org/wiki/Diff>.
- [36] Algorithmist, GNU Free Documentation License, "Longest Common Subsequence", 23rd Oct 2006, http://www.algorithmist.com/index.php/Longest_Common_Subsequence.
- [37] C. Charras and T. Lecroq, "Longest common subsequences", February 1998, Wikipedia®, Wikimedia Foundation, Inc., "Part-of-speech tagging", 8th Sept 2008, http://en.wikipedia.org/wiki/Part-of-speech_tagging
- [38] M. Elhadi and A. Al-Tobi, "Refinements of Longest Common Subsequence Algorithm", ACS/IEEE International Conference on Computer Systems and Applications HAMMAMET TUNISIA, May 2010.