Scientific Bulletin of the „Petru Maior" University of Tîrgu Mureş
Vol. 9 (XXVI) no. 1, 2012, ISSN 2285 – 438X (Online), ISSN–L 1841 – 9267

APPLICATIONS OF STRING MINING TECHNIQUES IN TEXT ANALYSIS

Horaţiu MOCIAN

SociaLook

Belşugului Street, no. 25, 540037, Mureş County
Targu Mures, Romania
[1]horatiu@socialook.net

ABSTRACT

The focus of this project is on the algorithms and data structures used in string mining and their applications in bioinformatics, text mining and information retrieval. More specific, it studies the use of suffix trees and suffix arrays for biological sequence analysis, and the algorithms used for approximate string matching, both general ones and specialized ones used in bioinformatics, like the BLAST algorithm and PAM substitution matrix. Also, an attempt is made to apply these structures and algorithms for text mining and information retrieval.

Keywords: clustering, bioinformatics, string mining, text analysis, information retrieval

1.Introduction

The field of string mining covers string related topics like efficient structures for storing strings, algorithms for exact and approximate pattern matching, finding repeating patterns in a string, or methods for calculating distances between two strings. Before going further into this topic, we define a string (in computer science) as a contiguous list of characters. The characters of the string can be of various types and significance: letters of the English alphabet, the 4 unitary components of DNA strings (A, C, G, T), bits (0, 1) and so on. Therefore, string mining has a large number of applications in computer science: search functions in word processors, information retrieval on the web, text mining, bioinformatics, data compression, spam filtering, etc.

Although there are well-established structures and optimal algorithms for storing and searching strings, which have been around since the 1970s and 1980s [19], research in this area has become very active again in the last decade. Two of the reasons for this increase in popularity are identified here: an exponential growth in the number of documents available in digital format and on the internet and rapid advances in computational biology that lead to availability of larger data sets containing longer sequences of DNA, RNA or proteins. Algorithms that were once fast enough to be applied for these tasks are falling behind. Moreover, the fields of text mining and biological sequence analysis employ a common task: approximate string matching. A high throughput of research activity related to this task was seen in recent years.

This paper is focused on string mining applications in two areas: text analysis (information retrieval, text mining) and bioinformatics (biological sequence analysis). The specific string related features in these areas are discussed in this section.

Text analysis and biological sequence analysis share some common characteristics. First, both work with large data sets. While biological sequences are much larger than text documents, the latter are more numerous. The Protein Data Bank, one of the largest protein databases contains more than 40000 sequences [8], while the Google index contains more than 1 billion documents. Henceforth, algorithms in both areas need to be extremely efficient in time and space. A more interesting similarity between text documents and biological sequences is that they both exhibit a kind of semantics. While text documents, written in natural languages, reflect the semantic understanding of these languages, the ”semantics” of biological sequences are in fact mutations. Nucleic acids or protein strings can suffer modifications during replication. These mutations can be: insertion

and deletion of elements, repetition of elements, or substitution of one element with another one. Even if these mutations occur, the function of the sequence as a whole remains largely unchanged. Thus, modified sequences have to be considered when patterns similar to the original sequence are searched. For this reason, approximate string matching is essential to molecular biology. The last type of mutations, substitutions, are similar to synonyms in natural languages. This fact suggests that bioinformatics techniques for approximate string matching can be used for semantic search and semantic similarity measures between two documents. However, the semantics of natural languages are more complex that the mutations in biological sequences.

Biological sequences and text documents also have a number of differences. Choosing a unit of information in a text document (equivalent to a character in the general string definition) is much more difficult than in DNA strings, for example. The simplest approach would be to consider each letter of the alphabet of a natural language, space and punctuation mark as string units. A more practical way is to consider each word as a unit. Alternatively, ngrams of a specific length can be used. If we want to take the semantic significance of the text into consideration, a coarser unit of information can be used: groups of words, or even entire phrases. Moreover, the fact that the chosen unit of information has variable length makes the algorithms more complex. The semantic model of text documents is much more complicated than the mutations of biological sequences. In addition to synonyms, other constructs like homonyms and hyponyms exist. Named entities may have multiple identical identifiers: the financial district of London can be identified by ”City” or ”Square Mile”, ”New York” by ”Big Apple”, etc. Last, but not least, different languages can be used to express the same concepts. Thus, two words from different languages can have the exact same meaning, making them similar from a semantic point of view. All these aspects make need to be taken into consideration when dealing with text documents.

Although both text documents and biological sequences are composed of strings, and exhibit some common properties, it is somewhat surprising to observe that, in general, totally different structures and algorithms are used to store and process these two categories of strings. Obviously, there are enough differences between the two to justify using different approaches, but their core concepts are similar. Moreover, no study has been made to compare the performance and accuracy of text analysis tasks using their traditional structures and the ones used in bioinformatics. Thus, there is no basis to dismiss any string mining algorithms when performing IR or Text Mining tasks. However, the classical document representation model cannot be aplied to biological sequence mining. In the next paragraphs, the dominating structures in each of these two fields will be overviewed and compared. There are two important structures used to represent strings in biological sequence analysis: suffix trees and suffix arrays. Suffix trees [44] represent each possible suffix of a string using a directed tree, whose leaf nodes correspond to a suffix each. Edges are labeled with substrings, and the concatenated text of all the edges from the root of the tree to one of the leaf nodes is a suffix of the represented string. Without going into any more detail (Section 2 contains a thorough description of this structure), we can state that the suffix tree allows fast search of any strings against the string representing by the tree. Of course, it has a lot more applications than search (see Section 2). Suffix arrays have been derived from suffix trees to reduce memory consumption, and they can be built either from suffix trees or directly. However, for the purpose of the introduction, we consider these two structures equivalent.

The widely accepted model for representing documents in information retrieval and text mining is the vector space model (VSM), proposed by Salton [40]. In this model, documents are represented as a n-dimensional vector that contains a score for each word that it contains(n represents the number of words). The score of a term is influenced positively by its frequency in the document (TF term frequency), and negatively by the frequency of the term in the entire data set (IDF inverse document frequency). Although this technique gives fairly good results, it doesn’t maintain the ordering of the words and it doesn’t make any semantic analysis of the document.

From these two descriptions, it is obvious that VSM cannot be used for comparing biological sequences, where the order of elements is crucial. However, there is nothing that prevents suffix trees from representing document. They have been adapted to hold only words. Moreover, algorithms for calculating TF and IDF have been developed. But, except for a few papers [45] [46], suffix trees and arrays have been hardly applied in text mining or information retrieval. This project aims to study the comparative performance of suffix trees and arrays, on the one hand, and the vector space model on the other hand, in the context of clustering, a basic text mining task.

The traditional distance measures used in text mining and IR, like the Euclidean distance, cosine similarity, or Jaccard similarity, cannot be used effectively with suffix trees. Although there is no problem in employing them, they are based on the VSM model, and don’t take advantage of the word ordering maintained by the suffix trees or their efficient comparison time. As an answer to this problem we will look at distance measures used in bioinformatics, or in general for strings, and evaluate their applicability for document similarity.

A secondary direction of research results by

studying the substitution matrices used for protein sequence searches. They represent the probability of a mutation of an amino acid into an other. There is a striking similarity between the amino acid substitutions and words that can have a similar senses. Accordingly, the possibility of extending the concept of substitution matrix to documents, in order to allow for minimal semantic understanding of the text. The possibility of applying this concept to multilingual text mining makes it even more interesting.

The primary objectives of this thesis is to analyze the feasibility of using suffix trees and/or suffix arrays to represent documents in text mining and information retrieval tasks. The secondary objective is to study the possibility of adding a semantic layer in the document representation models related to primary objective. A third possible objective is extending the semantic layer mentioned before into a transparent layer for multilingual text mining.

Two applications will be developed during the course of this project to evaluate the proposed objectives:

1. An application containing at least three clustering implementations, one based on suffix arrays, one bused on

suffix arrays with a semantic layer, and another one based on the vector space model

2. A knowledge extraction application that detects entities and extracts relationships between them. The output of this application will be added to the semantic layer of the clustering application

The motivation for this project stems from a combination of factors:

- Bioinformatics is a growing field, and research in this area is very active
- Mainstream Information Retrieval has remained unchanged in the last decade. Google, the leading web search engine, has been created in 1998. Although it suffered numerous changes since then, its concept remained untouched
- Biological sequences and documents have several features in common (e.g. large data sets, ”semantic” interpretation) but no study has been made to evaluate the feasibility of applying data structures used in bioinformatics to document representation

After the introduction, the paper will start with introducing the data structures that will be studied in this project,

in Section 2. The first part will cover structures for representing general strings, but which have been often applied

to computational biology: tries, suffix trees and suffix arrays. The second part will cover the vector space model,

and the inverted index, which are omnipresent in text mining and information retrieval tasks. Section 3 will look

at string comparison methods. The structure of this section is similar to the previous one, but bioinformatics

algorithms (BLAST) and structures (PAM) will be presented separately, because they cannot be applied to general

string mining tasks. The cosine and Jaccard similarity features will also be presented in this section.

Sections 4 and 5 will cover the two applications that are part of this project: clustering and relationship extraction.

They follow a similar structure: first, an overview of the task and the related issues is given, followed by a

brief presentation of related work, before detailing our approach, the corpora that will be used and the evaluation

methodology. The conclusions and future work will be presented in Section 6.

2. Structures for text representation

Tries, or suffix tries, were proposed by Aho and Corasick in 1975 [3] to improve bibliographic search. Keywords are represented using a finite state automaton which takes a document as input, and determines whether it contains any of the keywords. Figure 1 shows the trie representation of the strings ”A”, ”to”, ”tea”, ”ted”, ”ten”, ”i”, ”in”, and ”inn”.

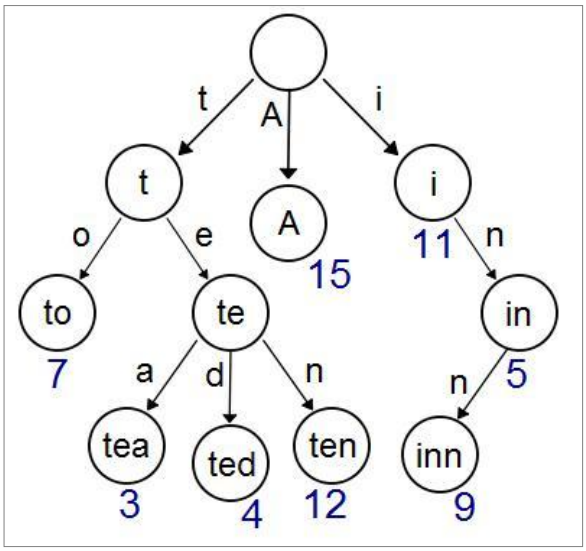

Fig. 1 – Suffix tree

The length of the suffix trie can be limited to reduce its space consmuption. A trie can be constructed in O(nk) time, where k represents the maximum length, using Aho and Corasick’s original algorithm, but this can be improved to O(n) if a suffix tree is built first, and then pruned to a maximum depth of k characters.

Suffix Trees were initially introduced by Weiner in 1973 [44], and they represent the single most important data

structure for string representation. Their first big advantage is a construction time O(m), linear in the size m of the processed string S. After the suffix tree is constructed, searching for a substring in it takes O(n) time, where n is the length of the substring (pattern). The fact that search time is independent of the length of the string makes suffix trees extremely useful. They can be used for any number of string-related tasks, constituting a bridge between exact string matching and approximate string matching algorithms [19].

This paragraph describes a suffix tree. Given a string S[0::n], a suffix is defined as a substring of form S[i::n]. The suffix tree represents all the possible suffixes of S in a rooted, directed tree. Edges are labeled with substrings of S. Each leaf represents a possible ending of the string, and reading the characters from edges on the path from the root to a leaf will give one suffix. Each suffix is represented by one and only one leaf node. Each internal node has at least two outgoing edges, and each edge begins with a different character. Suffix trees are also called compact tries, because an edge has more than one character. Figure 2 represents the suffix tree representation of string ”BANANA$”.

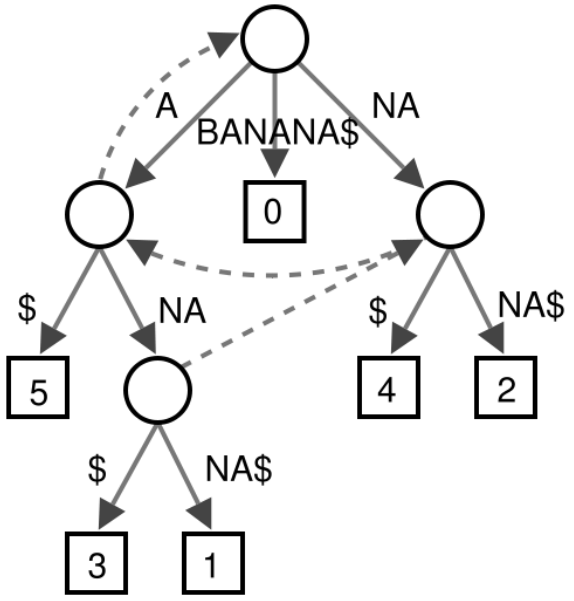

Fig. 2 – Suffix tree

To fulfil the condition that each suffix should be represented by a leaf node, no suffix must be a prefix of another suffix. In order to prevent this, a character uncontained in the string is appended at the end. This is called the sentinel character, and is generally denoted by $. The first linear time construction algorithm was proposed by Weiner in his initial paper about suffix trees [44]. It was followed several years later by another linear-time algorithm with less space consumption, by McCreight [31]. More recently, Ukonnen [43], has devised another linear time construction algorithm that works better in practice and is easier to understand.

Edge labels can be represented in two different ways. If the substring itself is used as a label, then the space complexity of the algorithm is $O(n^2)$. However, this can be easily reduced to O(n) if the start and end indices of a substring in S are stored.

Although very fast, the high memory consumption of suffix trees make them infeasible for large scale applications. In this scenario, suffix arrays are more efficient.

Suffix arrays were introduced in 1990 by Manber and Myers [29], as a compact representation of suffix trees. They report a space consumption reduction by an order of magnitude. Research in suffix arrays is currently active, with newer compact variants being developed in the recent years. Because they are similar in functionality to suffix trees, they can be used for many diverse string mining tasks.

As the tree variant, a suffix array represents all the possible suffixes $S[i..n]$ of a string $S[0..n]$. The suffixes, together with their corresponding indices, are stored in an array, in alphabetical order. Again, the sentinel character $ is appended so that a suffix cannot be the prefix of another one. Figure 3 represents the suffix array for the string ”abracadabra”.

Sorted Suffix	Index
a	10
a b r a	7
a b r a c a d a b r a	0
a c a d a b r a	3
a d a b r a	5
b r a	8
b r a c a d a b r a	1
c a d a b r a	4
d a b r a	6
r a	9
r a c a d a b r a	2

Fig. 3 – Suffix array

Search in suffix arrays is based on binary search and it has logarithmic complexity in the size n of the represented string. Manber and Myers [29] provide two basic algorithms for searching, one with complexity $O(mlogn)$, and another one with complexity $O(m+logn)$. They show that in practice these algorithms are competitive with search in suffix trees.

Suffix arrays can be constructed either directly, or from suffix trees. The initial direct construction algorithm by Manber and Myers had $O(nlogn)$ complexity. This has been improved to $O(n)$ since then [25]. The additional working space during construction time has a $O(nlogn)$ complexity. To improve this, a range of succint data structures have been developed. A succint data structure has an additional construction space requirement which is at

least close to linear to the size of input. [36] has a good overview of structures like compact suffix arrays, compressed suffix arrays or succinct suffix arrays.

In order to improve the computation efficiency of different tasks on suffix arrays, additional tables may be used. In this case, the suffix arrays and the additional tables are called collectively enhanced suffix arrays. The array LCP[0..n], where LCP[i] is the longest common prefix of suffixes s[i] and s[i-1] is noted as the lcp-array and is often used as a helper structure for suffix arrays. In [2] it is shown that a suffix tree can be replaced with an enhanced suffix array for every algorithm, maintaining the same time complexity.

In [16] suffix arrays are adapted to hold only suffixes that start with a word, in the situation where a string is in

fact a document. The new structure is called word-suffix array. An optimal algorithm is proposed, that constructs the suffix array in O(n) time and O(k) space, where k represents the number of words in the string.

Suffix arrays have been applied for a wide range of tasks, like frequency pattern mining, emerging substrings mining, range minimum queries (RMQ), or clustering.

3. String Comparison Methods

Classical algorithms for local and global approximate string matching are covered in this section. We have also included two algorithms from bioinformatics, that may be used for text mining, as well as the similarity measures currently used in IR and text mining.

There are three types of approximate string matching problems (they can also be applied to exact string matching) [1]:

- Global matching (compare entire strings with roughly the same size)
- Semi-global matching, or pattern matching (search for appearances of a pattern P in a larger string S)
- Local matching (find common substrings of the compared strings)

In text mining, global matching is equivalent to computing the similarity between two documents, while semiglobal matching can be used for computing the frequency of a word or phrase. Although local matching doesn't appear to have a direct use, it constitues the basis for global matching. These problems have the same uses in bioinformatics, but the document is replaced by a biological sequence. Also, global and local approximate matching are called global sequence alignment and local sequence alignment, respectively.

The edit distance is often used for computing the difference between two strings. It counts the number of insertions of a character into the first string, the number of deletions of characters from the first string, and the number of replacements of one character from the first string to a character from the second string. A match is also considered an operation, although its associated cost is 0.

The edit transcript is a string over the alphabet I, D, R, M (corresponding to the 4 operations) that describes the transformation from one string to another. The edit distance (and edit transcript) can be calculated using dynamic programming. The algorithm is described in detail in [19] by Gusfield. The time complexity for calculating the edit distance is $O(nm)$, while the complexity for creating the edit transcript is $O(n+m)$.

Global string alignment is similar in concept to the edit distance. It is often called simply string alignment. Therefore, if not mentioned otherwise in this paper, string alignment will refer to global string alignment. The global alignment of two strings S1 and S2 is obtained by inserting spaces into the strings until they have equal length and each character or space from one string coresponds to a character or space in the other strings. Insertions and deletions are collectively called indels, because an insertion in one string corresponds to a deletion in the other one. Although mathematically edit distance and string alignment are equivalent, they model different things. Edit distance shows the transformation steps from one word to another, while string alignment shows the final result, disregarding how it was reached. In his book, Gusfield explains that "the distinction is that of process versus product".

If we assign a score to each operation, then a score for the alignment can be defined as the sum of all the operations it contains. Usually, a match has a score of 1, a substitution is neutral, while an indel has a negative score of -1. Furthermore, we can define an optimal string alignment as being the alignment with the highest score.

The algorithm for determining the optimal alignment, proposed by Needleman and Wunsch [37] has a space and time complexity of $O(n^2)$. However, space complexity can be reduced to $O(n)$ if only the last row (or column) is stored.

The task of local aligment of two strings S1 and S2 is to find two substrings a and b, which have an optimal global alignment score greater than any other pair of substrings from S1 and S2. Smith and Waterman [41] have adapted the algorithm for global alignment for this task. To achieve this, they use negative scores for both substitions and indels, while each value of the dynamic matrix must be at least 0.

The algorithm has the same $O(n^2)$ complexity in space and time to the one for global alignment. The entire matrix needs to be stored, because there can be multiple candidates for substrings with optimal alignment score. Choosing the best substring when two or more have an equal score constitutes a topic of research.

Approximate string matching is heavily used in bioinformatics, to discover similarities between

genes, or protein sequences. The problem was that protein sequences can suffer mutations. Nucleotides, represented by four letters of the alphabet: A, C, G, T for DNA and A, C, G, U for RNA, or amino acids (represented by 22 four letters of the alphabet) can be deleted or inserted. Moreover, one nucleotide can transform into another. At the same time, entire portions of biological sequences may have no importance when studying the function of the entire sequence. All these possibilities need to be considered when computing the similarity between two biological sequences.

The same concept of ”mutations” can be applied for text documents. A document may have a high similarity with another, but it may contain several extra words or extra sentences (insertions), some text may have been cut out (deletions). Nucleotide transformation is equivalent to words with same meaning (synonyms) in a text. This is why approximate string matching algorithms are of direct interest in text mining and information retrieval. For this fields, it is actually more useful than exact string matching algorithms, because it is infeasible to discover similar documents by comparing them character by character.

BLAST was introduced in 1990 by Altschul et al. [4], and it became the preferred tool for searching biological sequence databases. One of its objectives was to improve the efficiency of the FASTA algorithm [28], which was used at the time for searching through biological sequences. When introduced, BLAST was reported to be an order of magnitude faster than FASTA, but newer versions of the latter reduced the gap significantly. Both algorithms use heuristics to reduce the number of possible searches. As BLAST is applied to DNA string, there is a version of it that runs on protein sequences, called BLASTP. In addition to BLASTP, there are other applications in the BLAST family.

In order to explain the BLAST algorithm, its fundamental objects need to be introduced first. A segment pair of two strings S1 and S2 is a pair of equal length substrings aligned without spaces. A locally maximal segment pair is a segment pair whose alignment score cannot be improved or maintained when reducing or extending the strings in either side. A maximal segment pair (MSP) is a pair with the maximum score over all possible segment pairs.

For a pattern P that is searched, BLAST finds all the sequences that have a higher MSP than a certain threshold. Moreover, sequences that contain a MSP below the threshold, but have segment pairs of statistical significance are also returned. The algorithm for finding sequences with high MSP is based on the concept of hot spots, taken from FASTA. BLAST calculates all substrings of the pattern P having a fixed length, and then searches for possible matches between these substrings, and any substrings of S. For DNA strings, the fixed length is 12. Once a hit is located, it is checked, by extension, if the sequence is contained in a locally maaximal segment pair. If the alignment score during extension drops far below the best score found for a smaller substring, the extension is truncated. However, because of this optimization, it is not guaranteed that BLAST finds all sequences having a MSP above the threshold.

It is difficult to derive a theoretical efectiveness for BLAST. Although less effective than optimal local alignment, and a bit less effective than FASTA in some cases, it is much more faster than the former, and slightly faster than the latter, and in general is competitive with both of them. It is recommended to use both FASTA and BLAST for searching biological sequences.

PAM matrices were the first important amino acid substitution matrices. Evolutionary mutations make it possible for one amino acid to transform into another one over time. This phenomenon is represented by the substitution matrix, which stores the probability of substituting any amino acid by another one. This matrix is used for protein database searches. The acronym PAM stands for either ”point accepted mutation” or ”percent accepted mutation”. An important issue is calculating the scores of substitutions. Some suggest that a proper algorithm for calculating substitution scores is the most important element of successful protein search.

PAM matrices, proposed by Dayhoff et al.[15], use PAM units to measure the ”evolutionary divergence”, or distance, between two sequences. Two sequences, S1 and S2 are defined as being one PAM unit divergent if a series of accepted point mutations, but no insertions or deletions, has transformed S1 into S2, with an average of one accepted point-mutation per one-hundred amino acids. A mutation is considered accepted if it didn’t change the function of a protein, or the change was either beneficial or unharmful. Note that a mutation can be applied to a position multiple times, so two strings having 1 PAM distance don’t necessarily have a sequence difference of 1%.

PAM matrices are a series of substition matrices that represent the expected evolutionary changes between 2 amino acids. Each PAM matrix represents the substitution scores for sequences that differ in a fixed number of PAM unit. Thus, PAM 1 is used to compare sequences that are 1 PAM units diverged, PAM 2 for sequences that are 2 PAM units diverged, and so on. Theoretically, the score for each pair Ai, Aj of amino acids in the PAM n matrix can be calculated by gathering many pairs of sequences that are n PAMs distant from each other, aligning them, and counting the number of times Ai and Aj appear in the same position in two different sequences. This result is then divided by the total number of pairs. However, in practice is impossible to align sequences in a way that reflects their evolutionary changes. In order to construct the matrices, Dayhoff gathered similar sequences, but

only for low PAM numbers, where the changes are easier to locate, and then applied the theoretical method. Higher PAM matrices were obtained by creating a matrix M that represents for each pair of amino acids the frequency of one being substituted by another for 1 PAM divergent sequences, and multiplying matrix M by itself n times (if PAM n is calculated).

Another substitution matrix is BLOSUM [20], which is more successful in capturing distant relationship between sequences. The concept of substitution matrices, can be readily applied to text documents, where a similar matrix can store either synonymity relationships, or co-occurrences, or any other kind of relationship. However, a performance issue is raised by the high number of words, which can easily exceed one hundred thousand, compared to 20 amino acids.

4. Application: Document Clustering

The first application that will be developed as part of this project is a clustering system. We chose clustering because it is a representative task in text mining, and data mining in general. Additionally, We have experience with clustering systems, having developed one as a part of a larger text mining project [14]. Also, the topic of the author's first Individual Study Option was surveying distributing clustering techniques.

Document clustering is one of the most suitable task for evaluating document representation models and similarity measures. In this paragraph, I am comparing it against the classification and document retrieval tasks. On one hand, not all classification techniques require direct comparison between documents, so it might be argued that computing document similarity is not a representative sub-task of classification. On the other hand, in document retrieval the query contains only several words. Consequently, the comparison between each document and the query will become a semi-global approximate string matching problem, instead of a global one. Another issue with the query is that it is difficult to define its semantic context, because of the small number of words it contains. In these conditions, adding a semantic layer would not be able to increase accuracy. Therefore, clustering is the most appropriate task in regard to our requirements.

Several clustering techniques will be developed as part of this application. All of them will use the same clustering algorithm. The difference between the techniques will be the document representation structure they use, as well as the similarity measure they employ. The first implementation, using the vector space model and cosine similarity measure, will constitute the baseline of our measurements. It will be followed by another implementation, that uses suffix arrays to represent strings, and a different similarity measure, based on a global approximate string matching algorithm. The third implementation will also use suffix arrays, but will add the semantic layer, based on the substitution matrix concept used for protein sequence comparisons in computational biology. The semantic layer will represent synonyms, polysemantic words, and relationships between named entities (see Section 5 for details).

Clustering, or unsupervised learning, is the task of grouping together related data objects [22]. Unlike supervised learning, there isn't a predefined set of discrete classes to assign the objects to. Instead, new classes, in this case called clusters, have to be found. There are a lot of possible definitions for what a cluster is, but most of them are based on two properties: objects in the same cluster should be related to each other, while objects in different clusters should be different.

Clustering has applications in many fields of computer science: data mining, statistics, pattern recognition, bioinformatics or image processing. While, in general, the same clustering algorithm can be applied in any of these fields with only slight modifications, some of the algorithms work better for certain clustering tasks. Also, in addition to the algorithm itself, there are other factors that influence the accuracy of a specific technique: the type of data that is clustered, the methods used for preprocessing, or the parameters of the algorithm. The focus of the application is not on algorithms, but on the preprocessing of items, and on the similarity measures used throughout the process.

When applied on a data set composed of text documents, this learning task is called document clustering. It has several particularities compared with the general technique. First, the feature space is high-dimensional, reaching easily orders of ten thousand or hundred thousand. In most cases, the features of documents consist of their words. Almost always, feature selection is applied: only the highest ranking words (according to same scoring method) are used further in the process. Second, the feature distribution of each document is sparse: only up to several hundred words are contained in a single document, which represent roughly around 1% of the total number of documents. After feature selection is applied, this value drops even further. Third, documents have a deep semantic context which should be taken into account (but often it is not) when clustering is performed. While two documents may have a small number of words in common, they can have the same meaning, provided that many of the words are synonyms. Another example is the use of named entities. If two documents contain the same named entities (companies, people, organizations), there is a strong relatedness between then, even if the majority of words don't match. However, these situations will be poorly dealt with by a clustering algorithm if it doesn't apply some basic semantic rules.

The algorithms for clustering are numerous

and diverse. However, most algorithms fall into two categories: hierarchical and partitional clustering.

The clustering algorithm chosen for implementation is called Quality Threshold clustering [21]. It was originally developed for clustering genes. In a previous paper [14], the algorithm was successfully implemented for document clustering. Thus, it provides another example that algorithms from bioinformatics can be applied in text mining.

QT clustering exhibits a number of advantages over the majority of other algorithms:

- It satisfies a quality criteria: the diameter of a cluster is guaranteed to be over a certain threshold;
- The number of resulting clusters doesn't have to be specified a priori;
- There is no randomness in the algorithm: each run will have the same results.

Instead of the widely accepted Vector Space Model, we will use Suffix Arrays [29] for representing documents. Suffix Arrays have been applied to text documents by Yamamoto and Church in [45], where they propose methods for computing TF and IDF for strings using suffix arrays. Fischer [16] introduces word suffix arrays, where only words are stored. Using this structure will allow to incorporate word ordering into any kind of similarity computation.

A discussion is worth regarding the chosen representation unit of suffix arrays. We will probably try several versions. One of them is to store all the suffixes, considering the character as the unit of the string. However, this doesn't seem to be the best choice. A more appropriate one is to consider words as the unit of strings, and store only suffixes consisting of one or more. Finally, we will investigate using an even coarser units, like phrases or even entire paragraphs.

The similarity measure will be in fact a global sequence aligning algorithm, which will be used to compare two documents. If two strings don't appear in the similarity matrix, they are considered completely different, from a semantic point of view. Since their actual spelling presents no interest, they will be considered completely different (i.e. they have a similarity of 0.0). However, edit distance with a small threshold can be used to spot misspellings.

Adding the semantic layer can be done using a matrix similar in concept to the one representing possible mutations in biological sequence alignment (Section 3.2). However, a sparse matrix implementation will be used, because only a small percentage of the total number of words will have an established semantical relatedness. The same matrix can be used for modelling any kind of relationship. If two words are synonyms they will receive a score of 1.0. The relatedness score between two entities will vary from 0.0 to 1.0, where 1.0 are different words or expressions for the same entities (e.g. "International Business Machines", "IBM" or "Big Blue").

The introduction of the semantic layer brings an increase in complexity and size of computation. Note that not only words need to be included in a semantic similarity matrix, but also different senses of the words, and groups of words that form expressions, increase the size of the matrix considerably. Moreover, a word sense disambiguation (WSD) classifier will need to be employed in order to find the sense of ambiguous words (or at least some of them). In order to help with this task, a part-of-speech (POS) tagger might also be employed.

There will be two categories of semantic relationships taken into consideration. First, there are synonyms and polysemantic word: different words that have the same sense. These will be replaced by their WordNet synsets (a group of words having the same sense). Thus, words with the same sense will be replaced with a common synset, even if they are different. Second, relationships between named entities will be introduced. For this, the output of the second application (Section 5) will be used.

We have identified at least 3 resources that can be used for semantic analysis of documents: WordNet, Wikipedia and SemCor. We will give a short overview on each of them in this section.

WordNet [33] is a lexical database for English, developed from scratch. It is the most complete lexical resource publicly available in a digital format. It has 3 databases, one for words, one for nouns, and one for adjectives and verbs, respectively. A completeWordNet entry consists of a set of synonyms (called synsets), with a dictionary-like definition, or gloss, and example uses. In addition to the synonymy relation described by the synsets, WordNet has the advantage of more complex relations, like hyponymy.

Wikipedia is an user-powered encyclopaedia. Articles can be added and edited by everyone, so they become a collaborative effort. Wikipedia has several features that makes it attractive for semantic analysis. First, it is an actively update corpus in the public domain. Moreover, it is available in more than 200 languages. Although many of them have few articles, there are over 2.8 million articles in English, with another 10 languages having between 300,000 and 1 million articles. However, articles in different languages about the same entity are not aligned, and

they can contain an entirely different content, making Wikipedia impossible to use as a parallel corpus. Words with multiple senses have a disambiguation page, with short definitions for each sense, that can be used directly by WSD classifiers for disambiguation. Another interesting thing is that articles usually provide links to other articles which are related in one way or another. As most of the articles are about entities, semantic relationships between them can be inferred.

SemCor [34] is a corpus developed by Princeton University, containing 352 texts in English. Out of these, 186 have all the words fully annotated (POS tagging, sense tagging), while only the verbs

are tagged in the others. It has more than 700,000 words in total. It is a subset of the Brown corpus.

There are two corporas that can be used for evaluation: RCV1 from Reuters[27] and OHSUMED.

RCV1 was released in 2000, and it contains about 810,000 Reuters, English Language News stories, published over the period of one year. The size of the uncompressed corpus is 2.5 GB. Representing the document in the vector space model reduced its size to about 600 MB. The OHSUMED corpus is 400 megabytes in size containing 348,566 clinically-oriented MEDLINE abstracts covering all references from 270 medical journals over a five-year period (1987-1991).

Because these two corpora are untagged, only internal evaluation measures can be used, like overall intra-cluster similarity. We are planning to use a smaller, annotated corpus, in order to use the external valuation measures: precision, recall and F-measures.

An interesting feature that can be added to the semantic layer is multilingualism. This can be done easily if we imagine that words express the same concepts, even if they are in different languages. Thus, using a parallel dictionary, relationships between words from different languages can be established and added to the semantic similarity matrix described in Section 2.2. Named entities usually retain their labels across languages (a notable exception are geographic names), so they should be easier to integrate. Thus, the two resources needed for multilingual clustering are parallel dictionaries and a WSD Classifier (which might itself need additional resources). An example of multilingual clustering can be found in [42]. An example of parallel corpora is MultiSemCor [6] (English/Italian).

5. Application: Knowledge Extraction

According to the Message Understanding Conferences (MUC), which pioneered named entity recognition at the end of the 1980s, named entities are words, or groups of words denoting names of persons, companies, organizations, geographical locations, calendar dates and numerals. The task of a Named Entity Recognition (NER) system is to identify and annotate each of the entities of a text with their files.

Advanced techniques for extracting named entities are out of the scope of this project. We have two alternatives for implementing this task:

1. Use existing lists of named entities, and make comparisons against them for each word in the document

2. Use an existing package for NER

The first technique is straightforward. The difficult part is populating the lists. However, data sets of named entities are readily available. A good examples is DBPedia [5], which extracts all the named entities fromWikipedia and organizes them in a taxonomy. Another database, GeoNames stores all the geographic locations on the Globe in a multi-tier taxonomy (e.g. continent includes country, which includes region, which includes city, etc.). Although a simplistic technique, it can become very powerful if comprehensive lists are available. Another advantage is that any word can be considered a named entity if it appears on a list. This allows us to extended the concept to domain-related entity types like: foods, plants, animals, illnesses, etc. This extension has a great overlap with another subtask of IE, terminology extraction.

There are many NER packages providing a very good accuracy. The best of them have an accuracy of over 96% [13]. Since human annotators don't achieve a 100% accuracy, because of inter-annotator disagreements, we can say that NER systems function at human performance levels. Because of this level of perfection, contributions in this field are hard to achieve. That is why developing a separate NER system was not considered necessary. Two of the most popular NER systems, ANNIE and Stanford NER, are overviewed in this section.

ANNIE (A Nearly-New IE system) [12] is a component of the GATE Natural Language Processing framework, developed at the University of Sheffield. ANNIE is a "portable" NER system, that can be applied in many different scenarios, from XML structured files to old court documents. We have previously integrated ANNIE in a text mining system [14].

Stanford NER [17] is a package that performs named entity recognition using Gibbs sampling. It assigns one of three possible classes to an entity: person, organization or location.

For this section, a broad definition of relationship extraction is used: finding relationships between entities. In this context, the term "relationship" is very loose. Some examples of relationships are: two cities located in the same country (or county), two companies competing in a certain industry, two persons working at the same company, or a person living in a certain city. Any kind of interaction between two entities can be considered relationship. Two entities appearing in the same document, or a hyperlink in Wikipedia from one to another also indicate that they might be related. We will consider several techniques for determining relationships in this chapter. However, not all of them can be considered extraction techniques in the sense of Information Extraction.

The most comprehensive type of Relationship Extraction, also called Relationship Discovery, or extracting Scenario Templates, uses NLP techniques to identify related entities. These techniques also identify the actual relationship, like merger, succession event or narcotics smuggling [13]. This task has been included in information extraction conferences lik MUC-6 (Message Understanding Conference) or SensEval-3. However, these algorithms are tailored for specific domains: for MUC-6, the set of documents was about changes in

executive management personnel. Consequently, they perform poor on the general domain. Others [39] have tried to solve this problem.

Another way of determining relationships between documents is a probabilistic approach. Calculating the co-occurrences of two entities in a corpus gives us the probability of appearing together in a document. If this probability is high enough, then a relationship between the entities can be inferred. Although less accurate than the NLP approach, this technique has the advantage that it doesn't need any linguistic-based algorithms or resources (except for the NER task itself). Additionally, it is language independent. On the downside, it cannot determine what kind of relationship exists between entities. A special case of co-occurrence can be considered Wikipedia hyperlinks. Each Wikipedia entry has links to other entries. If the originating and the linked entries are named entities, we can infer a strong relationship between entities. This kind of inference has been use by Mihalcea for word sense disambiguation [32].

Domain knowledge can also be used to infer relationships between entities. Comprehensive databases of geographic locations and relationships between them are available. Also, databases with companies contain data like the industries they activate in, headquarters locations, key personnel or aliases. Movie databases can be used to establish between actors, and so on. These kinds of relationships are straightforward to determine, but their success is highly dependant on the availability and quality of the external sources.

The last two techniques will be used in this project for relationship extraction. Although the NLP-based techniques for this task, are an active topic of research and present a lot of opportunities, they are out of the scope of this project. Developing successful algorithms in this area can be itself the topic of a paper. Even though the probabilistic and domain knowledge approaches cannot be considered extraction techniques in the sense of Information Extraction (the information is readily available), we believe that it is appropriate to list them under this category.

The implementation of the knowledge extraction application is discussed separately for each of its subtasks, as they can use different structures and algorithms.

The role of the named entity recognition task is to provide a starting point for the relationship extraction task. The two alternatives for doing this, listed in Section 5.3, are to use an external system or to use lists with entities. If an external system is used, then no special structure will be required for representing strings. On the other hand, when using lists of entities classical string matching algorithms can be employed on suffix arrays. Lists of named entities will be compiled from existing databases or domain terminologies, if specialization is needed.

No matter which technique is used for relationship extraction (probabilistic or domain knowledge) suffix arrays can be used for the implementation. Entity co-occurrences can be calculated using the algorithm for computing MI (Mutual Information) scores on suffix arrays proposed by Yamamoto and Church in [45]. Identifying relationships between entities using domain knowledge, will involve heavy use of string matching algorithms.

Two of the corpora that can be used for information extraction are: DBpedia and GeoNames. Wikipedia, another important corpus, was presented for the clustering application.

DBpedia[5] is a knowledge base developed at Freie Universitat, Berlin. It extracts information about entities from Wikipedia. It contains more than 213,000 persons, 328,000 places, 36,000 films and 20,000 companies. Labels of the entities and their attributes are available in 30 languages. It also provides a multi-domain ontology based on Wikipedia. It has the advantage as evolving witk Wikipedia's growth. DBpedia is based on user collaboration, so it does not have a strict formal structure as hand-crafted corpora like WordNet, and is more prone to inconsistencies. However, its breadth (covers multiple domains, not one in particular) makes it a valuable resource.

GeoNames is a database containing geographic locations from around the world. It has more than 600 hundred features (each representing a type of geographical landmark), grouped into 9 categories, including administrative divisions, land features, water features, roads and railroads, etc. It also provides an ontology for semantic integration. The data is accessible through one of its numerous web services. Each record in the database has more than a dozen attributes, including its exact positioning (longitude and latitude), the country in which it is situated, timezone, elevation, population, etc. The database is available under the Creative Commons attributive license.

The external NER systems that will be used have already been evaluated, so they won't be evaluated during this

project. For the list-based technique, annotated corpora, taken from MUC or SensEval conferences will be used for

assessing their accuracy, together with the standard precision, recall and F-measures.

Evaluating the accuracy of relationship extraction is quite difficult for several reasons. First, there are few corpora annotated for this task. The only ones available are the ones from the MUC and SensEval Conferences. However, these are specialized for a certain domain, so they cannot be used to test general systems, or those specialized in other domains. The lack of properly annotated corpora is also signalled in [39]. Moreover, it is impossible to have an objective quality metric, if the

concept of relationship is not clearly defined. However, this is difficult to do when using the probabilistic approach. Finally, the domain knowledge approach doesn't need to be evaluated, since the relationships are already given and there is nothing to extract.

A better way to evaluate the knowledge extraction application is to integrate it with the clustering application, and measure the improvement of the latter.

6. Conclusions

The research that was conducted for preparing this paper have shown that both suffix trees and suffix arrays have been used for representing documents, with good results. As examples, suffix trees have been used for clustering [46], while suffix arrays have been used for calculating term frequencies and document frequencies in a corpus [45]. Additionally, the author's experience with applying a gene expression clustering algorithm on documents, shows that algorithms in bioinformatics can be successfully applied for text mining and information retrieval.

Moreover, there is no study to contradict this claim. In fact, there are no comparative studies on the applicability of computational biology techniques to text analysis. The main objective of this project is to conduct a study of this kind.

More conclusions will be drawn after the applications presented in this paper will be implemented and experiments reported. If they succeed, this may open a new direction of research in document representation models and similarity measures that will be more appropriate for the semantic web paradigm than the existing ones. Even if the experiments will not be successful, the study will be useful for assessing the strengths and weaknesses of suffix trees and arrays when representing documents.

The possible outcomes of this project are one or more of the following:

1) A thorough evaluation of the advantages and disadvantages of adapting structures and algorithms from bioinformatics to text analysis. This objective will be successfully fulfiled once the application introduced in Section 4 will be implemented

2) Development of a better way for representing documents that takes into account semantic aspects. The clustering application from Section 4 will evaluate the fulfilment of this objective. If two implementations use the same algorithm, and differ only by the document model that they use, we can infer that if one of them has a higher accuracy (F-measure and/or overall similarity) than the other, it is "better".

3) Development of a new semantic similarity measure to be applied with the new document representation model. This will be evaluated in the same way as the previous objective, because the new similarity measure and the document model will be packaged together.

4) Development of a language-neutral document similarity measure. To evaluate this possible outcome, a separate clustering application needs to be build, that performs multilingual clustering. By language-neutral, we understand that the subset of clusters in one language returned by the multilingual algorithm would have a similar quality to the clusters obtained by running a monolingual algorithm on that language only.

5) Finding a way of integrating named entity relationships into the semantic document model and similarity measure. This outcome will be evaluated in a similar way as the second and the third outcome.

Acknowledgement

This paper has been written as part of Horatiu's work for the MSc degree in Advanced Computing at Imperial College London (2008 – 2009). The contents of this paper is based on the MSc Project Background Paper.

References

[1] Mohamed Abouelhoda and Moustafa Ghanem. *To Appear in Data Mining and Knowledge Discovery for Scientific Applications*, chapter String Mining in Bioinformatics. Springer Berlin / Heidelberg, 2009.

[2] Mohamed Ibrahim Abouelhoda, Stefan Kurtz, and Enno Ohlebusch. *Replacing suffix trees with enhanced suffix arrays*. J. of Discrete Algorithms, 2(1):53–86, 2004.

[3] Alfred V. Aho and Margaret J. Corasick. *Efficient string matching: an aid to bibliographic search*. Commun. ACM, 18(6):333–340, 1975.

[4] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipmanl. Basic local alignment search tool. *Journal of Molecular Biology*, 215(2):403–410, 1990.

[5] S¨oren Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. *Dbpedia: A nucleus for a web of open data*. pages 722–735. 2008.

[6] L. Bentivogli and E. Pianta. *Exploiting parallel texts in the creation of multilingual semantically annotated resources: the multisemcor corpus*. Nat. Lang. Eng., 11(3):247–261, 2005.

[7] Pavel Berkhin. *Survey of clustering data mining techniques. Technical report*, 2002.

[8] Helen M. Berman. The Protein Data Bank: a historical perspective. *Acta Crystallographica* Section A, 64(1):88–95, Jan 2008.

[9] Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien. *Using linear algebra for intelligent information retrieval*. SIAM Rev., 37(4):573–595, 1995.

[10] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine.

Computer Networks and ISDN Systems, 30(1-7):107 – 117, 1998. *Proceedings of the Seventh International World Wide Web Conference*.

[11] Krzysztof Cios, Witold Pedrycz, and Roman W. Swiniarski. *Data mining methods for knowledge discovery*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[12] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.

[13] Hamish Cunningham. *Information extraction - a user guide.* CoRR, cmp-lg/9702006, 1997.

[14] Ovidiu Dan and Horatiu Mocian. Scalable web mining with newistic. In *Proceedings of PAKDD Conference 2009*, 2009.

[15] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. In M. O. Dayhoff, editor, *Atlas of Protein Sequence and Structure*, pages 345–352+. 1978.

[16] Paolo Ferrag ina and Johannes Fischer. *Combinatorial Pattern Matching, chapter Suffix arrays on words*, pages 328–339.

[17] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In ACL '05: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[18] E. Forgy. *Cluster analysis of multivariate data: efficiency versus interpretability of classifications*. Biometrics, 21:768–780, 1965.

[19] D. Gusfield. *Algorithms on strings, trees and sequences.* Cambridge University Press, Cambridge, United Kingdom, 1997.

[20] S. Henikoff and J. G. Henikoff. Amino Acid Substitution Matrices from Protein Blocks. *Proceedings of the National Academy of Science*, 89:10915–10919, November 1992.

[21] L. Heyer, S. Kruglyak, and S. Yooseph. *Exploring expression data: Identification and analysis of coexpressed genes.* Genome Research 9, pp. 1106-1115, 1999.

[22] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[23] I. T. Jolliffe. *Principal component analysis*. Springer Series in Statistics, Berlin: Springer, 1986, 1986.

[24] Rasha Kashef. *Cooperative Clustering Model and Its Applications*. PhD thesis, University of Waterloo, Department of Electrical and Computer Enginnering, 2008.

[25] Dong Kyue Kim, Jeong Seop Sim, Heejin Park, and Kunsoo Park. *Constructing suffix arrays in linear time. Journal of Discrete Algorithms*, 3(2-4):126 – 142, 2005. Combinatorial Pattern Matching (CPM) Special Issue.

[26] R. Krishnapuram, A. Joshi, and Liyu Yi. A fuzzy relative of the k-medoids algorithm with application to web document and snippet clustering. *Fuzzy Systems Conference Proceedings*, 1999. FUZZ-IEEE '99. 1999 IEEE International, 3:1281–1286 vol.3, 1999.

[27] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: *A new benchmark collection for text categorization research*. J. Mach. Learn. Res., 5:361–397, 2004.

[28] DJ Lipman and WR Pearson. *Rapid and sensitive protein similarity searches*. Science, 227(4693):1435–1441, 1985.

[29] Udi Manber and Gene Myers. Suffix arrays: a new method for on-line string searches. In SODA '90: *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 319–327, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics.

[30] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[31] Edward M. McCreight. *A space-economical suffix tree construction algorithm*. J. ACM, 23(2):262–272, 1976.

[32] Rada Mihalcea. Using Wikipedia for automatic word sense disambiguation. In Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; *Proceedings of the Main Conference*, pages 196–203, Rochester, New York, April 2007. Association for Computational Linguistics.

[33] George A. Miller. *Wordnet: a lexical database for english. Commun*. ACM, 38(11):39–41, 1995.

[34] George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. A semantic concordance. In HLT '93: *Proceedings of the workshop on Human Language Technology*, pages 303–308, Morristown, NJ, USA, 1993. Association for Computational Linguistics.

[35] F. Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4):354–359, November 1983.

[36] Gonzalo Navarro and Veli M¨akinen. *Compressed full-text indexes*. ACM Comput. Surv., 39(1):2, 2007.

[37] S B Needleman and C D Wunsch. A. General method applicable to the search for similarities in the aminoacid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[38] Clark F. Olson. *Parallel algorithms for hierarchical clustering*. Parallel Comput., 21(8):1313–1325, 1995.

[39] Cartic Ramakrishnan, Krys J Kochut, and Amit P Sheth. *A framework for schema-driven relationship discovery from unstructured text. Knowledge Creation Diffusion Utilization*, pages 583 – 596, 2006.

[40] G. Salton, A. Wong, and C. S. Yang. *A vector space model for automatic indexing*. Commun. ACM, 18(11):613–620, 1975.

[41] W Smith and M S Waterman. Identiffication of common molecular subsequences. *Journal of Computational Biology*, 147:195–197, 1981.

[42] R. Steinberger, B. Pouliquen, and C. Ignat. *Navigating multilingual news collections using automatically extracted information*. pages 25–32, 20-23, 2005.

[43] Esko Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

[44] Peter Weiner. Linear pattern matching algorithms. In SWAT '73: *Proceedings of the 14th Annual Symposium on Switching and Automata Theory (swat 1973*), pages 1–11, Washington, DC, USA, 1973. IEEE Computer Society.

[45] Mikio Yamamoto and Kenneth W. Church. *Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus*. Comput. Linguist., 27(1):1–30, 2001.

[46] Oren Zamir and Oren Etzioni. Web document clustering: a feasibility demonstration. In SIGIR '98: *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 46–54, New York, NY, USA, 1998. ACM.