Informatics Research Proposal
Learning to Segment Web-Pages

———— ————

s———

Abstract

The extraction of the main content sections of web-pages is an essential step in the indexing procedure of any information retrieval system. A challenging task arises when we are dealing with multi-content web-pages. The list-like visual representation of such web-pages transforms the content extraction problem into a web-page segmentation task. Various web-page segmentation approaches have been proposed, mainly focusing on interpreting the visual characteristics of each web-page element. Such techniques lack of both flexibility and ability to assess the significance of each extracted section. We propose a novel approach for the web-page segmentation task based on a sequential view of each web-page's source code. We expect that the visual uniformity and repetitiveness of the listed content sections should correspond to repetitive patterns in the underlying sequence derived from each web-page. By borrowing techniques from the field of computational biology and, particularly, from genome sequence analysis, we plan to develop, evaluate and compare two different segmentation methods: one based on dot-plots and one based on dynamic programming.

Contents

1 Introduction

The World Wide Web is undoubtedly the largest and richest source of information available nowadays. It contains enormous amounts of unstructured or semi-structured data related to any possible domain. The efficient use of such a vast information source would be impossible without the existence of *Information Retrieval* (IR) systems and, in particular, search engines.

Search engines, both general-purpose ones –such as Google, Bing, Yahoo etc.– as well as domain-specific ones, have been developed and have evolved into sophisticated and impressively useful tools for conveniently navigating through the WWW. In order to do so, such systems rely on systematically performing a certain indexing procedure. Figure 1 illustrates a simplified versions of this process: Firstly, the IR system obtains newly created or updated, html-formatted web-pages with the aid of a web-crawler. Secondly, the informative content of each fetched web-page is gathered. Since we are almost exclusively dealing with raw textual content, an important transformation step takes place in order to come up with a content representation suitable for IR purposes. Finally, the transformed content data is indexed in a way that will allow its efficient future use.

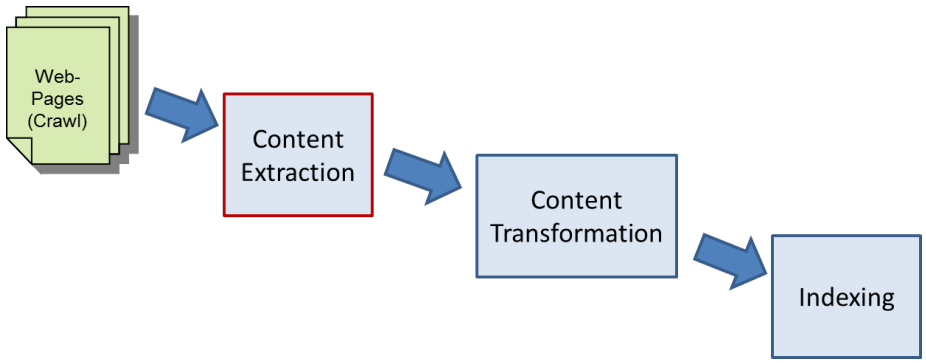


Figure 1: Overview of a Search Engine's Indexing Process

The aforementioned gathering of informative content available in a web-page is formally known as *Content Extraction*. Content extraction is the process of converting the unstructured or semi-structured information stored in web-pages into a purely textual sequence of words/tokens (no web-related code/tags), while separating its main content section(s) from any less informative web-page segments. Such "surrounding" sections are present in almost any web-page and may include navigational links, copyright notices, advertisements, multimedia etc. Although useful for the user of the web-page, these elements provide limited information about the actual content of the page and could mislead an information retrieval system if considered equally important as the main content elements, resulting in poor performance.

A variation of the above definition of content extraction arises when dealing with web-pages that consist of more than one content segment. In this case, we are still interested in discarding the surrounding, non-informative sections but, this time, the goal is to extract a list of *independent, non-overlapping content segments* instead of just a single one. This special case of content extraction will be the focus of this project.

The task of extracting non-overlapping content sections from multi-content web-pages can be viewed as a web-page segmentation problem. Given a multi-content web-page our goal is to identify the appropriate boundaries, such that the resulting sections between them will correspond to the actual content segments we wish to extract.

A number of solutions have been employed in order to address related web-page segmentation problems, some of which will be further discussed in a later section. Our approach will be based on the view of a web-page as a sequence. The textual, html-based source of web-pages lends itself to being viewed as a sequence of elements in an intuitively sensible manner.

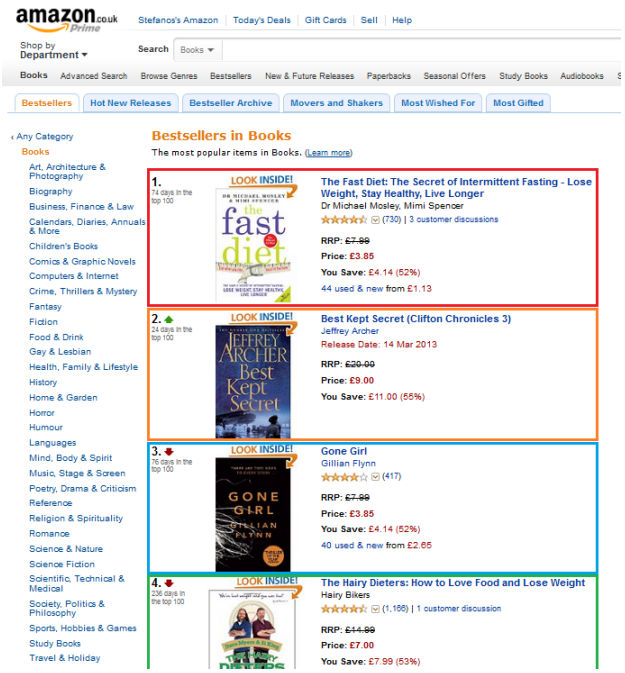


Figure 2: Example of a web-page containing multiple content elements

An illustrative example can be seen in Figure 2. An e-shop's products list is an example of a multi-content web-page. It is obvious that the visual representation of the listed content segments is almost identical. Since such visual repetitiveness is caused by the underlying html source code, we expect that a similar repetitive pattern should be evident in the source's sequence of tokens/characters.

Our aim is to utilize the sequential nature of web-pages as a way of identifying logical units/sub-sequences in a multi-content page in order to extract the list of content elements that occur in it.

The rest of this report is organized as follows: Section 2 will focus on previous related work on the domain of web-page segmentation, as well as on a number of techniques from different areas of study that might be useful for this project. In section 3, some of the methods that are likely to be used in our approach will be explained. Section 4 contains a description of how we are going to evaluate our work. A work-plan of the main goals and sub-goals based on which the project is going to be organized is provided in section 5. We conclude this research proposal in section 6.

2 Background

Content extraction from multi-content web-pages has not been explicitly addressed for *Information Retrieval* purposes. However, as stated in the introductory section, the close relation of this task to web-page segmentation is apparent. A number of techniques for automatically segmenting web-pages with or without the aid of human annotation have been proposed and evaluated.

In this section we will provide a brief literature review on a variety of web-page segmentation and content extraction methods. In addition, we will cover a number of approaches on the problem of identifying repeated patterns in sequences. Such techniques have been utilized profusely in the bioinformatics research area and are usually referred to as *sequence alignment* methods.

2.1 Web-Page Segmentation

One of the most commonly used methods for content extraction is Finn's Algorithm [Finn et al., 2001]. The aim of this technique is to extract a single content segment from a web-page, while disregarding any other non-informative sections. In this method, the web-page is viewed as a sequence of two types of tokens; html tags and non-tag words. It is expected that the main content section of the web-page will mostly consist of non-tag tokens. The idea is to find the portion of the sequence that contains the fewest number of html tags –also known as "tag-plateau". Although this method performs rather successfully for single-content web-pages, it cannot be directly utilized for our task, where the goal is to extract multiple content segments.

A number of approaches have been proposed for segmenting web-pages in the related literature. Web-page segmentation, although being a task closely related to content extraction, has been frequently used in numerous other applications too. Duplicate detection for web-mining purposes, information extraction and identification of important page segments for displaying web-pages in screen-space constrained devices (PDA's, smartphones, etc.) are some of them.

Some segmentation methods have focused on analysing the appearance of a web-page, using specific visual cues, in order to identify coherent blocks that clearly differ from the rest of the page. The VIPS algorithm [Cai et al., 2003] was developed using such an approach. Heuristic-based rules for analysing the DOM-tree structure of a web-page were used. This technique, in spite of resulting in satisfactory segmentation performance in their experiments, is not flexible enough and may be hard to adapt in the always changing visual structure of web-pages due to its heuristic nature. Moreover there is no clear way of assessing the significance of each extracted segment in order to distinguish between informative and non-informative sections of the page, which is a core aim of content extraction.

An interesting graph-theoretic approach was proposed by Chakrabarti et al. [2008]. In their work, the task of segmenting web-pages into semantically cohesive sections was formulated as a weighted-graph optimization problem. A graph representation of the relations between each DOM-tree element was created, where graph nodes corresponded to different DOM entities. Given manually labelled training data, their goal was to learn how to automatically distinguish which DOM-tree elements are more likely to be semantically related and which not. This novel approach produced adequate segmentation results and is definitely more flexible than any heuristic alternatives due to its ability to learn given manually pre-segmented web-pages. However, gathering labelled data is not easy and, additionally, it was reported that time-efficiency was a big issue in their algorithmic implementation. Once again, deriving the level of content-wise significance of each segment is not an obvious task.

Although the previously mentioned segmentation approaches fail to capture the significance of each extracted segment, they could be used as a preparatory step in order to narrow down the portion of the web-page that will be examined.

Finally, a particular *Information Extraction* method, which has a lot in common with the sequence-based approach we are willing to take, should be mentioned. Information extraction is the task of automatically extracting structured information from unstructured or semi-structured data. When used for extracting structured data from web-pages, this task significantly resembles the extraction of multiple content sections we are willing to address.

In their approach, Chang et al. [2003], used a sequence pattern discovery method in order to build information extractors to extract structured data from web-pages. Their software was a semi-automated application –some user interaction was needed for selecting the most appropriate pattern to be used– which used a technique for identifying repeated sub-sequences in the sequence of html-tags of each web-page. Although their goal was slightly different, the success of their method works as supporting evidence that analysing the sequential nature of web-pages is a promising approach for extracting information about its content. The technique used was borrowed from the *sequence alignment* literature, which will be the focus of the following subsection.

2.2 Sequence Alignment

As stated in the introduction, our approach will be based on a sequential representation of web-pages. Given arbitrary sequences constructed using a pre-defined vocabulary, any task related to the comparison of sequence pairs or the analysis of the intra-sequence similarities/repetitions form a group of problems labelled as *sequence alignment tasks*. Such tasks range from finding exact or approximate sub-sequence duplicates between different sequences to identifying tandem or dispersed repeats of arbitrary length and similarity inside a single sequence. Although most of these tasks are of no interest in regard to our content extraction problem, the techniques stated below can be modified to address more than a few of them.

One of the tools that is commonly used in the biological sequence alignment literature is a special type of recurrence plot, called *dot-plot* [Gibbs and Mcintyre, 1970]. Dot-plots can be utilized for both inter-sequence similarity searching as well as for identifying any type of repeats in a single sequence. They are a powerful, yet simple to implement tool for visually analysing sequences in order to identify sections of significance. Barring their visual interpretation use, they are also the basis of various algorithms for extracting duplicate sub-sequences between sequence pairs or inside single sequences. Apart from their widespread use in computational biology they have also been employed in other domains. Church and Helfman [1993] used dot-plots to explore self-similarities in text documents and programming code consisting of millions of lines. They have also been used for analysing continuous time-series [Yankov et al., 2005].

With the exception of dot-plots which can be used –with small variations– for both exact and approximate sequence alignment/repeats matching, most algorithms are focused on only one of these task families.

Firstly, a number of useful data structures have been extensively used for discovering exact repeats in single sequences. String look-up tables, automata and, mainly, suffix trees –and their more space-efficient counterpart, suffix arrays– have been utilized for such purposes [Abouelhoda and Ghanem, 2010].

In addition, a significant number of algorithms have been developed and used extensively for approximate sequence alignment. The vast majority of these techniques are based on dynamic programming. The first such approach was proposed by Needleman and Wunsch [1970]. The Wunsch-Needleman algorithm is used to extract the optimal global sequence alignment, i.e. the alignment of two whole sequences that minimizes the edit distance. An extension of this algorithm, called the Smith-Waterman algorithm, has been widely used for identifying approximate repeats in a single sequence [Waterman and Smith, 1981].

Undoubtedly, dynamic programming algorithms for approximate sequence alignment are essential tools for real-life sequence analysis application. This is due to the fact that, in both computational biology as well as other domains, exact matches of long sub-sequences are either very rare or statistically insignificant. However, dynamic programming algorithms, as the ones mentioned above, tend to be unsuitable for large-scale alignment purposes, due to their –at least– quadratic time complexity. A number of heuristics based on combining exact and approximate matching techniques have been proposed and used in the past decades. Abouelhoda [2005] provides a detailed survey on these methods which, in the field of comparative genome analysis, are also known as *filtering techniques*.

In the next section we will firstly formally define the problem of web-page segmentation in terms of input, expected output and hypothesis. Then, a description of the main methods to be used, based on the aforementioned related work, will be given.

3 Approach

The view of a web-page as a sequence of elements from a pre-defined alphabet makes intuitive sense. Since each web-page is stored as a textual html-formatted file, one can easily comprehend that a number of sequential representations could be derived, e.g. a sequence of `ASCII` characters, a sequence of tokens etc. According to this, it should be expected that an underlying structure of logical units exists in such a sequence. These logical units should correspond to different sections of the web-page. The accurate extraction of these units results in a segmentation and, consequently, their interpretation should make it possible to identify independent content sections. In particular we expect that a number of web-page content elements presented in a list-like manner, will create a repetitive pattern in the underlying sequence related to the web-page. This is due to the visually uniform structure of such content segments.

3.1 Problem Definition

In order to formally state the hypothesis of our proposed research, we should first clearly define the input and expected output of our segmentation algorithm.

Given the source code of a web-page, the first task should be to tokenize it based on the type of alphabet we wish to use. This will result in either a character-based or a word-based tokenization. Since both alternatives could be assessed, let Σ be the ordered alphabet derived and let $|\Sigma|$ be its size. Let S be the tokenized web-page's sequence over Σ and $|S| = \mathrm{n}$ be its length. We denote $S[i]$ the i-th element of the sequence, for $0 \leq i < n$. We denote (i, j) a pair of positions in S, for $0 \leq i < j < n$ and $S[i..j]$ a sub-sequence of S starting at position i and ending at position j. A sequence S constructed from a web-page tokenization will be the input of our algorithm. Our algorithm's output will be a set of pairs $\{(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)\}$, for $k > 1$, $i_p < j_p < i_q < j_q$ and $p < q$, which will correspond to a set of sub-sequences $\{S[i_1..j_2], S[i_2..j_2], \ldots, S[i_k..j_k]\}$. These sub-sequences will in turn correspond to k independent and non-overlapping segments of the web-page.

The hypothesis of our research proposal is that sequence alignment techniques for approximate intra-sequence repeats recognition could be well-suited solutions for the task of extracting multiple content segments from multi-content web-pages. This is due to the sequential nature of html-formatted web-pages which is expected to reflect their visual repetitiveness. In order to assess this claim, our aim is to develop, evaluate and compare two methods borrowed from the computational biology literature similar to those referenced in section 2.2: one based on dot-plots and one on dynamic programming with the aid of suffix-trees (or suffix-arrays).

Utilizing such methods in the domain of web-page segmentation is a novel idea and should require algorithmic variations of the original techniques. In section 3.2 two core approaches will be described. Varied versions of these should end up being used in our project.

3.2 Methodology

In this section we will briefly describe two possible sequence alignment approaches for our task: One based on dot-plots and one based dynamic programming.

3.2.1 Dot-plots

As stated in section 2.2, dot-plots are a simple, yet powerful visualization tool used –among others– for identifying repeated sub-sequence patterns within a sequence over a pre-defined alphabet. The process of constructing a dot-plot to identify such patterns within a sequence S is straightforward. The sequence S is plotted against itself in its original order.
This produces a *2D* plot/matrix, whose starting point is the upper-left corner. For each point/cell (x, y) of the dot-plot a dot is placed if $S[x] = S[y]$ or, otherwise, the cell is left blank. When whole sub-sequences $S[i_1..j_1]$, $S[i_2..j_2]$ are partially or completely identical, distinguishable patterns are created. In order to illustrate these dot-plot patterns, a number of trivial, yet representative sequences have been used in order to construct simple dot-plots and are shown in Figure 3.

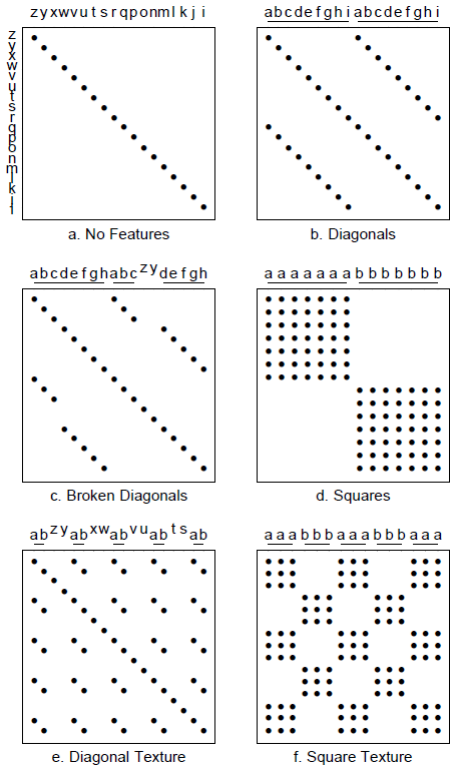
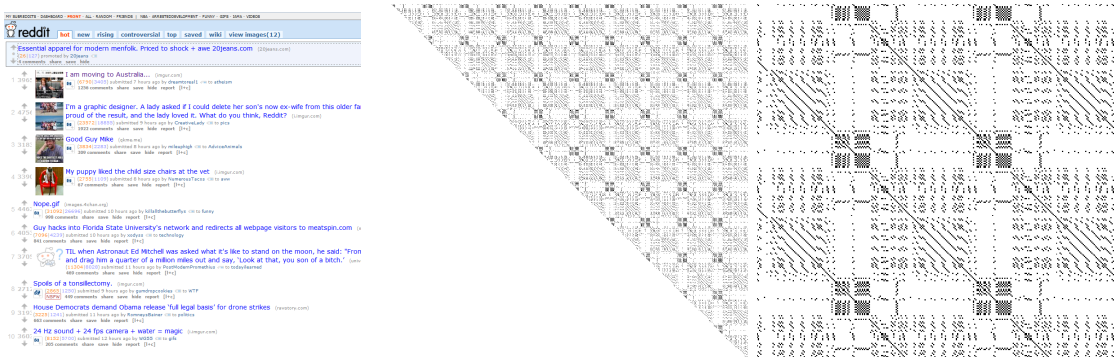


Figure 3: Example of dot-plots constructed from simple sequences

Obviously, the main diagonal of such a dot-plot will always be an uninterrupted line, since any subsequence is perfectly matched with itself. However, any line off the main diagonal may be indicative of an existing repeat. For example, plot (a) has no line off the main diagonal, as the plotted sequence "zyxwvutsrqponmlkji" includes no repeats. Plot (b) has two lines of the main diagonal (one can be ignored since they're obviously symmetrical) which indicate an exact sub-sequence repeat. This is actually the case, since the plotted sequence is a result of the concatenation of two exact copies of the sequence "abcdefghi". On the other hand the broken diagonals of plot (c) indicate an approximate repeat, which includes a two-character insertion ("zy") in the second repeat of the sub-sequence "abcdefgh". Plots (d), (e) and (f) illustrate some more complex repetitive patterns.

We expect that similar but far more complex patterns should be evident if we construct dot-plots from sequences derived from multi-content web-pages. As in these trivial examples, approximate repeats indicating structural source code patterns should exist. In order to illustrate this, we constructed a dot-plot from a word-based tokenized web-page consisting of user-posts. In such a page, each user-post could be considered as an individual content segment. The actual web-page, as well as its corresponding dot-plot is shown in Figure 4.



(a) Actual web-page *(b)* Full dot-plot *(c)* Zoomed-in section

Figure 4: Real web-page example

Interestingly, we can clearly see repeated patterns in the constructed dot-plot. This confirms our assumption that the underlying token sequence of web-page source code carries the repetitive sub-sequences that correspond to visually interpretable patterns.

A variety of methods for sub-sequence significance scoring have been proposed over the years, most of them evolving from the initial approach of Gibbs and Mcintyre [1970]. Their techniques were based on the idea of determining how probable it would be to get a diagonal line –indicating– a repeat by chance. Two simple methods were used in their study:

a) The length of all diagonal runs longer than a threshold is recorded. The frequencies of run lengths are calculated. These values are then compared to the run frequencies that were expected to appear in the dot-plot by chance.

b) The total number of matches that occur in every diagonal is determined. Once again these values are compared to the expected number of matches for each diagonal if the sequences compared were random.

Obviously these methods are far to simplistic and will be varied according to the needs of our task. Additionally, they tend to be quite time-wasting so a number of efficiency-driven heuristics should be utilized in order to avoid examining sections of the dot-plot which clearly include no apparent repeats. An interesting idea could be to employ techniques from the computer vision literature in order to identify sections of the dot-plot in which repetitive patterns are more evident and, then, examine those for a more detailed extraction of repeats.

3.2.2 Suffix-Array Aided Dynamic-Programming Methods

Dynamic programming algorithms have always been the backbone of the genome analysis research. They have been used extensively for multiple sequence alignment as well as single-sequence approximate repeat identification. The majority of such algorithms are variations of the initial method introduced by Needleman and Wunsch [1970].

As stated before, the quadratic time complexity of this family of algorithms makes them inconvenient for large-scale sequence alignment tasks. The most common solution for addressing this issue is the use of an exact sub-sequence matching algorithm (often the suffix-array supermaximal repeat algorithm [Abouelhoda et al., 2004]) as a preparatory step in order to extract a set of identical fragments which are then extended using a dynamic programming approximate alignment algorithm (most commonly the one proposed by Waterman and Smith [1981]). This way, large regions of the compared sequences, which show no significant similarity, are ignored.

Detailed descriptions of both the exact and the approximate matching algorithms are beyond the scope of this proposal. However, an overview of one common method for combining these algorithms is provided below.

The *seed-and-extend* strategy is used in order to efficiently extract approximate repeats from large sequences. Given a sequence S, a set of identical repeated subsequences is extracted using an exact matching algorithm based on suffix-arrays. These exact repeats are fixed similarity regions also known as *anchors*. We expect that any approximately repeated region should be identified by extending these anchors to the left and to the right by applying a dynamic programming algorithm until the similarity score of the repeated section falls under a certain threshold.

A slightly more conservative approach, known as *double seed extension*, requires the existence of two anchors close to each other in order to start their extension. An illustrative example of this technique is shown in Figure 5: Two exact matches have been located close to each other (thick lines) using the suffix-array algorithm. They are extended to both directions using the Smith-Waterman algorithm as long as the similarity score of the approximate repeat is adequate.

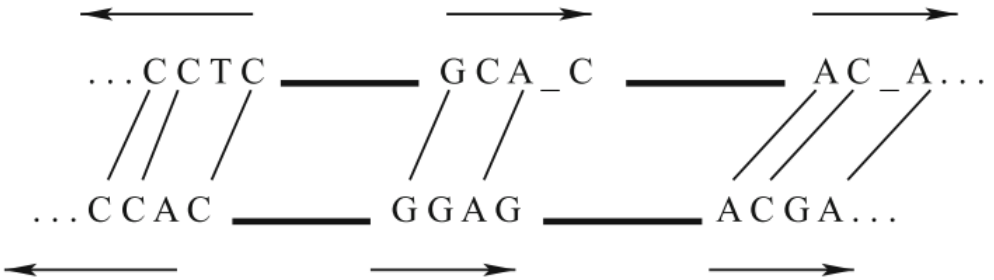


Figure 5: Seed-and-extend: Exact repeats (thick lines) are extended resulting in larger homologous sub-sequences (approximate repeats)

4 Evaluation

As formally defined in section 3.1, given an input web-page sequence S our goal is to compare the segmentation performance of two novel segmentation techniques based on identifying repeated sub-sequence patterns in the sequential representation of each web-page. The outputted segmentation in terms of boundary position pairs of each algorithm should be evaluated and compared.

4.1 Measures

Recall the segmentation output of our algorithm as defined previously. Each algorithm will output –for a given web-page– a set of position pairs $\{(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)\}$, which correspond to predicted web-page segments. Assuming that we have constructed a *ground truth* dataset of the true web-page segmentations, our goal is to compare each algorithm's segmentation to the actual one. Both the predicted and the ground truth segmentations should be on a html-tag level. This means that no segmentation boundaries should be placed mid-tag regardless of the chosen tokenization policy.

A simple and intuitively sensible way of comparing our prediction with the ground truth segmentation is to derive *True Positive*, *True Negative*, *False Positive* and *False Negative* counts which could then be used to come up with *Accuracy*, *Recall* and *Precision* measures. For each sequence element we can check whether it is correctly classified as being within a content sub-sequence or not. Figure 5 illustrates this using a trivial example. The green brackets correspond to the position pairs of our ground truth dataset, while the red brackets to the position pairs outputted by one of the evaluated methods.

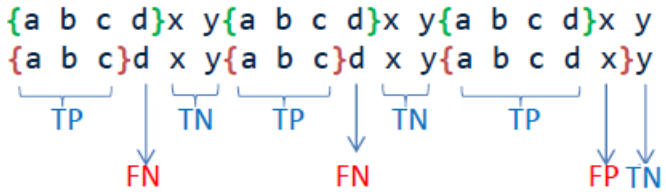


Figure 6: Comparison of the true (upper) and predicted (lower) segmentations

We can then compute the evaluation measures mentioned above as follows:

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

Tuning the various parameters/thresholds of each method would result in different performance measurements which could then be illustrated using *Recall-Precision Plots/ROC Curves*.

4.2 Ground Truth Dataset

In order to evaluate the segmentation performance of our algorithms we have to create a benchmark dataset that will serve as the ground truth. The first step towards doing so will be to gather a list of web domains which will be used for our testing and evaluating purposes. Such web-site types could include search engine results pages (Google, Bing, Youtube etc.), online stores (Amazon, Ebay etc.), blogs, user communities (forums, reddit etc.) and more.

We will then perform a domain-specific web-crawl in each of these websites in order to gather enough web-pages for our evaluation. It is expected that each web domain should have a very specific html-code structure that is responsible for the listed presentation of its content segments. By manually examining the source code of these web-sites, coming up with custom built regular expressions in order to extract the true segments of each web-page should be straightforward. By doing so we'll have a large enough corpus of correctly segmented web-pages which we could then use in order to evaluate the performance of our algorithms.

4.3 Baseline

The absence of previously established research on multiple content segments extraction gives rise to a number of problems when trying to define the baseline for our evaluation. Most importantly, a benchmark dataset as the one described in the section above does not already exist. As a result, our proposed algorithms cannot be tested and evaluated on existing datasets. The focal point of our project will be the comparison of the aforementioned techniques in order to assess their suitability for the task of segmenting multi-content web-pages. Comparisons with previously established web-page segmentation algorithms will not be the focus of this project.

5 Work Plan

A work plan was created based on the proposed method and evaluation techniques described in the two previous sections. The following Gantt chart illustrates the main goals that have been set for this project as well as a duration approximation for each one.

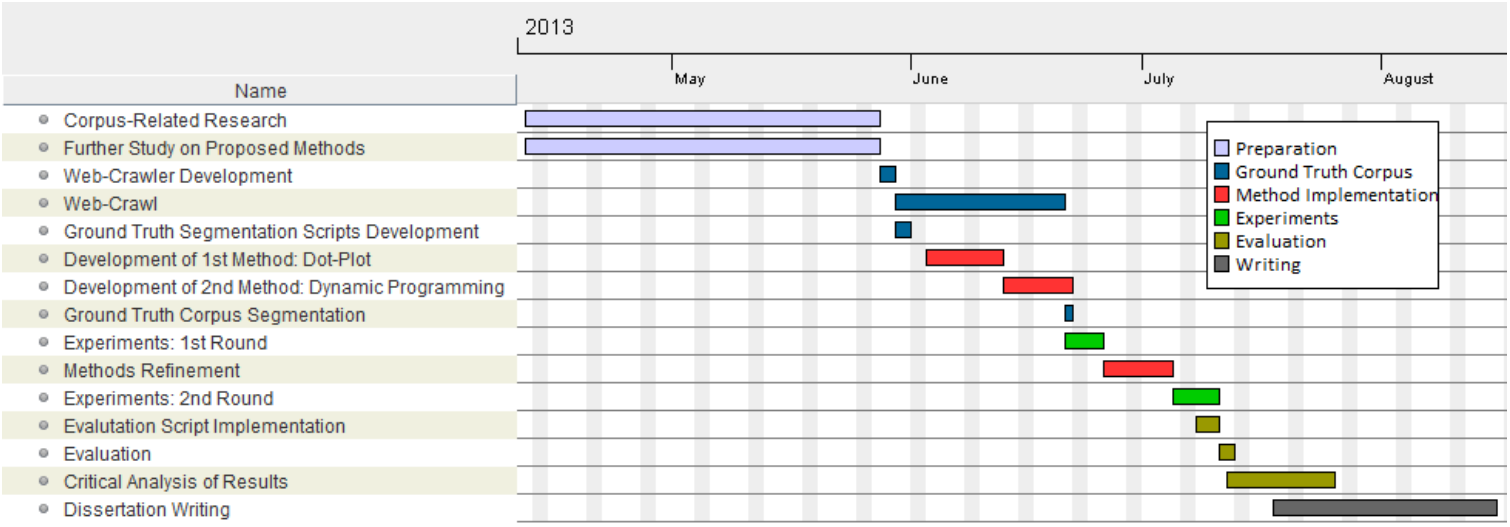


Figure 7: Project's work plan

A brief description of each goal, its sub-goals and some potential risk factors paired with the corresponding back-up plan is provided below:

- Corpus Related Research
  - *Description:* Identify suitable web-sources for construction of ground truth corpus.
  - *Sub-goals (for each candidate web-source):*
    Check whether listed content elements' structure is consistent enough for our purposes.
    Check web-source's crawling policy.

- Further Study on Proposed Methods
  - *Description:* More research on the details of each method. Narrow down possible variations.

- Web-Crawler Development
  - *Description:* Implement web-crawling script capable of crawling thousands of web-pages from various web-sources.
- Web-Crawl
  - *Description:* Employ web-crawler.
  - *Risk factors:* Might take more time than expected. Also, blacklisting from web-sources could be a possibility. In such a case we could substitute some web-sources with static corpora (available from supervisor).
- Ground Truth Segmentation Scripts Development
  - *Description:* Implement a unique custom segmentation script for each web-source in order to come up with our ground truth corpus.
  - *Sub-goals (for each candidate web-source)*:
    Examine source code characteristics and patterns
    Identify segmentation boundaries in source code
    Implement segmentation script based on regular expressions
- Development of 1st and 2nd Methods
  - *Description:* Implement the two main algorithms based on the methodology description of section 3.2.
  - *Sub-goals*:
    Dot-plot-based method
    Dynamic programming method
  - *Risk factors:* Implementation issues might cause delays. In such a case the experiment parts of the work plan should be shortened time-wise.
- Ground Truth Corpus Segmentation
  - *Description:* Employ segmentation scripts on the gathered corpus.
- Experiments: 1st Round
  - *Description:* Run first round of experiments mainly focusing on evaluating the correctness of the implemented methods.
- Methods Refinement
  - *Description:* Change/Debug possible problematic parts of implemented methods based on 1st round of experiments
- Experiments: 2nd Round
  - *Description:* Actual segmentation experiments. Predicted segmentation of web-pages should be obtained.

- Evaluation Script Implementation
  - *Description:* Implement script to compare predicted and ground truth segmentations. Evaluation measures as the ones described in section 5.2 should be obtained.
- Evaluation
  - *Description:* Employ aforementioned evaluation scripts.
- Critical Analysis of Results
  - *Description:* Analyse obtained results. Come up with comparisons/-conclusions regarding the effectiveness of proposed methods.
- Dissertation Writing

6 Conclusion

In this report we proposed a novel approach for the problem of extracting content segments from multi-content web-pages. Although the web-page segmentation task has been addressed numerous times in the related literature, no adequate solution exists that fits the content extraction problem from an information retrieval point of view.

The intuitively sensible view of a web-page as a sequence has never been directly taken advantage of for content extraction purposes, despite the fact that it has been successfully used in other domains for decades. The sequence analysis literature provides a rich background consisting of a vast number of different techniques and algorithms that could be employed for such a task. By applying some of the most commonly used methods, we are willing to get an idea of how well does this family of algorithms fit the web-page segmentation problem. Any positive results should be enough evidence that more variations of sequence alignment techniques could be utilized in a similar way and provide a new direction for many problems related to web-page segmentation.

References

Abouelhoda, M. (2005). *Algorithms and a software system for comparative genome analysis*. PhD thesis.

Abouelhoda, M. and Ghanem, M. (2010). *String Mining in Bioinformatics*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Abouelhoda, M. I., Kurtz, S., and Ohlebusch, E. (2004). Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2(1):53–86.

Cai, D., Yu, S., Wen, J., and Ma, W. (2003). Extracting content structure for web pages based on visual representation. *Web Technologies and Applications*.

Chakrabarti, D., Kumar, R., and Punera, K. (2008). A graph-theoretic approach to webpage segmentation. *Proceeding of the 17th international conference on World Wide Web - WWW '08*, page 377.

Chang, C.-H., Hsu, C.-N., and Lui, S.-C. (2003). Automatic information extraction from semi-structured Web pages by pattern discovery. *Decision Support Systems*, 35(1):129–147.

Church, K. and Helfman, J. (1993). Dotplot: A program for exploring self-similarity in millions of lines of text and code. *Journal of Computational and Graphical ...*.

Finn, A., Kushmerick, N., and Smyth, B. (2001). Fact or fiction: Content classification for digital libraries.

Gibbs, A. and Mcintyre, G. (1970). The diagram, a method for comparing sequences. *European Journal of Biochemistry*.

Needleman, S. and Wunsch, C. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*.

Waterman, M. and Smith, T. (1981). Identification of Common Molecular Subsequences. *Journal of molecular biology*, 147(1):195–197.

Yankov, D., Keogh, E., and Lonardi, S. (2005). Dot plots for time series analysis. *Tools with Artificial Intelligence*.